GREEKDB

An E-Ship Thesis written by

AARON SIMMONS

and submitted to

KETTERING UNIVERSITY

in partial fulfillment
of the requirements for the
degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

2016

_____
Student

_____
Faculty Thesis Advisor

_____
Committee Member

DISCLAIMER

This thesis is submitted as partial fulfillment of the graduation requirements of Kettering University needed to obtain a Bachelor of Science in Computer Science Degree.

The conclusions and opinions expressed in this thesis are those of the student author and do not necessarily represent the position of Kettering University or anyone else affiliated with this culminating undergraduate experience.

PREFACE

This thesis represents the capstone of my five years combined academic work at Kettering University and past work experiences. My Culminating Undergraduate Experience provided the opportunity for me to use the knowledge and skillset learned while at Kettering to manage a project of this magnitude.

Although this thesis represents the compilation of my own efforts, I would like to acknowledge and extend my sincere gratitude to those individuals who both helped in testing of the application and, of providing insights into their needs as officers of the Michigan Delta chapter of Phi Delta Theta, to help improve the application and its functionality. In addition, I would like to thank Ryan Allen for his contribution of assets and Dr. John Geske for his feedback and time devoted to reviewing this thesis.

TABLE OF CONTENTS

DISCLAIMER 2

PREFACE 3

LIST OF ILLUSTRATIONS 12

I. INTRODUCTION 14
- Problem Topic 14
- Background 15
- Criteria and Parameter Restrictions 15
- Methodology 16
- Primary Purpose 17
- Overview 17

II. CONCLUSIONS AND RECOMMENDATIONS 21
- User Activity Logging 22
- Designing Permissions Systems 22
- Designing Data Structures 22
- Organizing Code 23
- User-Defined Groups 23
- Tying Together Different Components 23
- Use Prepared Statements 24

III. ENUMERATED REQUIREMENTS 26
- Overview 26
- Requirements 26
  - R1 Assist organizations in maintaining information about their membership 26
  - R2 Assist organizations in tracking information about their community involvement 27
  - R3 Assist organizations in maintaining good record of their judicial proceedings 28
  - R4 Allow organizations to manage permissions and access for officers and members 29
  - R5 Assist officers in communicating with the organization's members and officers 30
  - R6 Help organizations improve financial tracking and reporting of members' financial standing 30
  - R7 Assist organizations with reporting for internal and external uses 30
  - R8 Provide a simple overview for members and officers of important information 31
  - R9 Be clean 31

IV. APPLICATION MARKET AND REQUIREMENT DRIVERS 32
Introduction 32
Market 32
Revenue Model 33
Free model 33
Mobile application 33
Paid subscription 34
Competition 35
ChapterSpot® 35
Tendenci® 36

V. APPLICATION ENVIRONMENT AND ACTORS 38
Introduction to the Environment Model 38
The System 39
User Interaction with the System 40
The Database 41
Reports 42
Encryption 42

VI. USE CASE MODELS 43
Introduction to the Use Case Models 43

VII. USE CASES 47
Introduction to the Use Case Documents 47
2-Panel Render 48
Main flow 1 48
Main flow 2 48
3-Panel Render 50
Main flow 1 50
Main flow 2 51
Add Judicial Case 52
Main flow 52
Alternate flow 1: the user doesn't have permission to submit new events 53
Alternate flow 2: there are more messages in-queue 53
Alternate flow 3: the text alert checkbox is not checked 53
Add Member 54
Main flow 54
Alternate flow 1: member ID not entered 54
Add Transaction Record 55
Main flow 55
Alternate flow 1: the user does not have permission to access the page 55
Add Service Event 56
Main flow 56
Alternate flow: the user doesn't have permission to submit new events 56

Report Bug 58
Main flow 58
Download Contact Card (vCard) 59
Main flow 59
Load Accounts Receivable Report 60
Main flow 60
Alternate flow 1: the user does not have rights to access the page 61
Load Active Roster Report 62
Main flow 62
Alternate flow 1: the user does not have permission to access the page 63
Alternate flow 2: the next record will overrun the page and is part of the first column 63
Alternate flow 3: the next record will overrun the page and is part of the second column 64
Load Chapter Directory 65
Main flow 65
Alternate flow 1: user has special officer permissions for the chapter directory 65
Alternate flow 2: user has restricted rights to view only him/herself 66
Load Chapter Service List 67
Main flow 67
Alternate flow 1: the user does not have permission to access the page 68
Alternate flow 2: the user only has access to edit his/her own events, but can view all 68
Alternate flow 3: the user only has access to edit his/her own events, and view no others 68
Alternate flow 4: the user only has access to view all 68
Alternate flow 5: the user only has access to view his/her own events 69
Load Community Service Report 70
Main flow 70
Alternate flow 1: the user does not have permission to access the page. 71
Load Easy-Biller 72
Main flow 72
Alternate flow 1: the user does not have permission to access the page. 72
Load Full Roster Report 73
Main flow 73
Alternate flow 1: the user does not have permission to access the page 74

Alternate flow 2: the next record will overrun the page and is part of the first column 74
Alternate flow 3: the next record will overrun the page and is part of the second column 75
Load Judicial Case Creator 76
Main flow 76
Alternate flow 1: the user does not have access to the judicial case creator 77
Alternate flow 2: the user does not have permission to create cases for all members 77
Load Judicial Case Editor 78
Main flow 78
Alternate flow 1: the user has only view access to the case 79
Alternate flow 2: the user has no access to the case 80
Load Judicial Case List 81
Main flow 81
Alternate flow 1: the user does not have permission to access the page 81
Alternate flow 2: the user only has access to view all 82
Alternate flow 3: the user only has access to view his/her own cases 82
Alternate flow 4: the user only has access to edit others' cases, but can view all 82
Load Landing Page 83
Main flow 83
Alternate flow 1: the user is not logged in 86
Alternate flow 2: the user has no access to view his/her community service record 86
Alternate flow 3: the user has no access to view his/her judicial record 86
Alternate flow 4: the user does not have admin access to any of the community service, judicial, and financial tools 87
Alternate flow 5: the user does not have admin access to the community service tools 87
Alternate flow 6: the user does not have admin access to the judicial tools 87
Alternate flow 7: the user does not have admin access to the financial tools 87
Alternate flow 8: the user is on a mobile browser 88
Load Mass-Announcer 89
Main flow 89
Alternate flow 1: the user does not have permission to access the page 89
Load Member Information Page 91
Main flow 91
Alternate flow 1: admin opens a link to a member profile. 92

Alternate flow 2: user opens a link to his/her own profile..... 92
Alternate flow 3.1: user opens a link to a member profile which does not exist..... 92
Alternate flow 3.2: admin opens a link to a member profile which does not exist..... 93
Alternate flow 4.1: admin opens a link to a new profile: “?localid=(new)”..... 93
Alternate flow 4.2: user opens a link to a new profile: “?localid=(new)”..... 93
Load Navigation Banner..... 94
Main flow..... 94
Alternate flow 1: user is using GreekDB from a mobile device... 94
Alternate flow 2: user has administrative rights to a linked tool.. 95
Alternate flow 3: user has no rights to the linked tool..... 95
Load Officer Permissions Overview..... 96
Main flow..... 96
Alternate flow 1: the user does not have rights to access the page 96
Load Open Accounts..... 97
Main flow..... 97
Alternate flow 1: the user does not have permission to access the Page..... 98
Load Permissions Editor..... 99
Main Flow..... 99
Alternate flow 1: the user does not have rights to access this page 99
Load Punishment Totals Report..... 101
Main flow..... 101
Alternate flow 1: the user does not have permission to access the page..... 102
Load Recent Judicial Cases Report..... 103
Main flow..... 103
Alternate flow 1: the user does not have permission to access this page..... 104
Load Service Event Creator..... 105
Main flow..... 105
Alternate flow 1: the user does not have access to the service event creator..... 106
Alternate flow 2: the user does not have permission to create events for all members..... 106
Load Service Event Editor..... 107
Main flow..... 107
Alternate flow 1: the user has only view access to the record... 108
Alternate flow 2: the user has no access to the record..... 108
Load Session Bar..... 109
Main flow..... 109

Alternate flow 1: user is not logged in 109
Print Member-Selector Box 110
Main flow 110
Send Balance Reminder 111
Main flow 111
Alternate flow 1: the user does not have access to the page 112
Alternate flow 2: the phone number does not include the corresponding country code 112
Alternate flow 3: the phone number is too short 112
Send Mass-Announcement 113
Main flow 113
Alternate flow 1: the user does not have access to the page 114
Alternate flow 2: there are more messages in-queue 114
Send Text Notification 115
Main flow 115
Alternate flow 2: the phone number is too short 115
Alternate flow 3: the phone number does not include the corresponding country code 116
Toggle All Members 117
Main flow 117
Alternate flow 1: the toggle all selector is changed to unchecked 117
Update Judicial Case 118
Main flow 118
Alternate flow: the user doesn't have permission to update cases 118
Update Member 119
Main flow 119
Alternate flow 1: member ID not entered 119
Update Officer Permissions 120
Main flow 120
Alternate flow 1: the user does not have permission to use the tool 120
Update Service Event 121
Main flow 121
Alternate flow: the user doesn't have permission to update events 121
Update User Information 122
Main flow 122
Alternate flow 1: email address not changed 122
Alternate flow 2: password left blank 123
Alternate flow 3: email left blank 123
User Clicks Logo 124
Main flow 124
Alternate flow 1: user is not logged in 124
User Login 125

Main flow 125
Alternate flow 1: missing information 126
Alternate flow 2: incorrect password 126
User Logout 127
Main flow 127
Register User 128
Main flow 128
Alternate flow 1: invalid information entered 129

VIII. APPLICATION DATA MODEL 130
About the Data Model Diagram 130
Relationships 131
Keys 131
Members vs. Users 131
Officers 132
Organizational Heirarchy 132

IX. APPLICATION FRONT-END 134
2-Panel Render 134
3-Panel Render 135
Navigation Banner 136
Standard Footer 137
Session Bar 137
Member-Selector 138
Login Page 139
Landing Page 140
Chapter Directory 142
Chapter Service List 144
Judicial Case List 146
Officer Permissions Table 147
Mass-Announcer 149
Edit Account 150
Open Accounts 151
Bug Reporter 152
Member Information Page 153
Service Event Creator 155
Service Event Editor 156
Judicial Case Creator 158
Judicial Case Editor 159
Officer Permissions Editor 160
Easy-Biller 161
Sample SMS Notification 162
Community Service Report 163
Judicial Punishments Report 165
Recent Cases Report 167
Accounts Receivable Report 168
Chapter Roster(s) 170

X. DETAILED SCENARIOS 172
Explanation 172
Determination 172
User Login 174
Brief Description 174
Detailed Description 174
Load Accounts Receivable Report 177
Brief Description 177
Detailed Description 177
Submit Judicial Case w/ Text Notification 179
Brief Description 180
Detailed Description 180
Load the Navigation Banner 182
Brief Description 183
Detailed Description 183
Load the Landing Page 186
Brief Description 186
Detailed Description 186

XI. APPLICATION DESIGN DECISIONS 194
GreekDB Permissions Handler 194
Permission Structure 194
Permission Granularity 195
Nationally-Mandated Permissions 195
Lessons Learned 196
Database Member Heirarchy (Organization, Chapter, Member) 197
Member-User Distinction 198

GLOSSARY 199

LIST OF ILLUSTRATIONS

Figures **Page**

1. Environment model 39

2. Use case model 1 44

3. Use case model 2 45

4. Use case model 3 46

5. Data model 130

6. 2-panel render 134

7. 3-panel render 135

8. Navigation banner 136

9. Standard footer 137

10. Session bar (not logged in) 137

11. Session bar (logged in) 137

12. Member selector 138

13. User login page 139

14. Landing page 140

15. Chapter directory 142

16. Chapter service list 144

17. Judicial case list 146

18. Officer permissions table 147

19. Bulk announcer utility 149

20. Edit account page 150

21. Open accounts page 151

22. Bug reporting tool 152
23. Member information page (member profiles) 153
24. Service event creation tool 155
25. Service event editing tool 156
26. Judicial case creation tool 158
27. Judicial case editing tool 159
28. Officer permissions editor 160
29. Chapter billing tool 161
30. Sample SMS notification 162
31. Community service contribution report 163
32. Judicial case results report 165
33. Recent judicial case summary report 167
34. Accounts receivable report 168
35. Chapter roster print-out 170
36. User login collaboration diagram 176
37. Load accounts receivable report collaboration diagram 179
38. Submit judicial case w/ text notification collaboration diagram 182
39. Load navigation banner collaboration diagram 185
40. Load landing page collaboration diagram part 1 190
41. Load landing page collaboration diagram part 2 191
42. Load landing page collaboration diagram part 3 192
43. Load landing page collaboration diagram part 4 193

I. INTRODUCTION

This project focuses on the development of a web application for managing greek-letter organizations. The application is designed to assist these organizations in tracking information such as their members' and alumni's contact information, community service contributions, maintain adequate records on judicial hearings, automate the reporting process for their national bodies, and to provide a front-end for members to track their own information, including community service contributions, donations, judicial case involvement, and account balances. As an additional note, the application seeks to maintain alumni involvement post-graduation, so some tools remain enabled for members to continue using after graduation.

Problem Topic

The intention for this application is to serve as a model for the eventual production of a service to help manage membership-based organizations of any kind, in particular, those with large overseeing bodies.

For many organizations, in particular those who experience a high rate of turnover (about every 3 years for greek-letter organizations on most campuses), maintaining records is vital to the organizations survival and ability to thrive. Further, in particular for greek-letter organizations, the management of the organization serves as a learning experience for the members/officers. Combining these two factors with the limited time individuals have to devote to the organization, greek-letter organizations often lose a lot of information with each turnover, and do not have the experience or expertise to perform their operations at their highest efficiency.

Various software solutions attempt to solve individual problems, including billing, finances, etc. Rarely however, do they address all (or even many) of these issues and often are outside financial viability for the organization. For national organizations looking for a global solution, they may be forced to contract out work to a third-party to build custom solutions, which is costly and only provide a solution for that single organization.

Background

Many of the problems noted above can be resolved simply by retaining information about the organization's operations. The solution to this is a database for storing information about an organization's operations, in a manner that is tailored to what is important to the organization. There are couple benefits to software managing this: First, the information may now be manipulated to provide a better experience for the end-user, providing simpler management for the organization. Second, where a larger entity is involved, we can provide reporting services on this data, providing value for that organization. Third, best practices can be enforced in the software, educating the users and complementing their operations.

Criteria and Parameter Restrictions

The following are some accommodations that were made in the process of writing the software/service:

- Light-weight: The software was being written to work on budget hardware to avoid needing to purchase new components.
- No Content Management Framework (CMF): Using a CMF would have taken away from the project to include others work for designing the user interface. It would additionally have caused issues on the afore-mentioned hardware.

- Low-cost: During testing the service would be utilized by a chapter for no cost. Because of this, renting or buying expensive infrastructure would have been infeasible.

- Time: A version of the software with enough perks to encourage its own use, needed to be completed within 3 months, working only weekends and some evenings.

- Flexibility: The service needed to work for any greek-letter organization, not just the one to which I am affiliated.

- Engaging: A number of software applications on the market to handle these kinds of organizations are exclusively top-down, relying on administrators of the organization, exclusively to update information in the system. A need exists for a system that is managed from the top-down, but allowed interaction from the bottom to engage members, reduce administrator workload, and provide more data and a better workflow.

This thesis presents the results of my implementation and testing of a web services for aiding in the management of multiple, distinct organizations.

Methodology

The stages were spread out chronologically, but there were essentially four main stages to the production of the website (GreekDB), with a fifth planned phase.

1. Prototype Phase: An initial model was produced using Microsoft Access® to show the kinds of information that the application would work with. The project went on hiatus for a while, but the Access® model proved invaluable in planning and research.

   The prototyping phase was one full rotation of the standard five-stage Software Development Life Cycle (SDLC). The initial model was based on an early understanding of the kinds of information which the system needed to track, and Microsoft Access was chosen based on my own understanding of the platform. The application was designed around that information and which information as relevant to each house officer. The application was broken out into different packages for each officer, which allowed them access to different tools and data. Officers tested the software and provided feedback on the system.
   In testing, some issues were uncovered with the platform, including the cost associated with application licenses, network requirements, manageability, and lack of security configurability. It became apparent that a new platform was needed, and to make the application accessible from anywhere, that would mean a web-based application.

2. Learning Phase: This phase represents the learning period for each of the respective languages for the application: HTML, PHP, MySQL, JS, and CSS for use in this thesis. This fits in the Requirements Analysis, Design, and Implementation sections of the standard five-part SDLC model. Implementations of individual tools in the application were driven both by user need and by my own abilities in the respective languages. Here is where the time constraint of 3 months was relevant to the creation of GreekDB. During this time, a number of sacrifices were required either because of experience with web development, or because of time constraints.

3. Testing Phase: This represents a 3-month period of real world testing by end-users of the application. The testing phase overlaps with the next phase, as new features are requested, bugs were found, and usability changes were required.

4. Maintenance Phase: This phase represents the ongoing changes and updates to the service. Having met the testing deadline and gathered valuable information, a number of modifications to code to improve the user experience were made. Some of these include corrections to changes or "shortcuts" that were required for the testing phase to complete on time.

5. Version 2: Version 2 is the afore-mentioned planned fifth phase. It represents the next iteration in the software. Some changes were too large to be realized in the maintenance phase, or were too general for GreekDB. Version 2 marks the end of a completed loop of the standard SDLC, and the start of a new loop for the next iteration of the application.

Primary Purpose

This thesis presents the results of my implementation and testing of a web service to aid in the management of multiple, distinct organizations.

Overview

The following chapters contain information about the requirements for the software, explanations of process flow for the different parts of the application, and how the different parts of the application come together and interact. Chapter III explains the different requirements that were uncovered through conversation with other officers, and through my own experience. These requirements drove creation of GreekDB and were necessary components of the application for it to be effective. The requirements section

can also be thought of as the results of the data-gathering phase of the application's development, or part one of the SDLC (Requirement Analysis).

Chapter IV explains the market for the software, the business strategy, and the competition, and explains how those considerations have influenced the software requirements and design decisions.

Chapter V talks about the actors, or the "moving" components of the GreekDB system. It describes visually, and in text, the different acting components, and the persons who interact with the GreekDB system, and how those communications take place. Chapter V is part of the design phase of the SDLC, and is representative of the final product.

Chapter VI describes diagramatically, the communication between different components of GreekDB and 3rd parties, users, and other components. It also gives a high level overview of the effects of what kinds of operations are secured by the user-configurable security system built into the application.

Chapter VII describes in detail the different communications that take place between different components of the system, users, actors, and 3rd party services. They can be thought of as communication transcripts, and they describe the different possible outcomes based on user inputs and system states.

Chapter VIII shows the data model of the database that GreekDB utilizes for long-term storage of data. It also describes how different tables are related, and what kinds of relationships the different tables have to eachother. These relationships can help to understand how the system extrapolates information from one table based on criteria for another.

Chapter IX shows the GUI (Graphical User Interface) for the application as it is implemented. It also includes annotations of what different kinds of information are captured in the the GUI, and descriptions of the purpose of the interface, and any nuances to the interface and its design or behavior.

Chapters VI, VII, VIII, and IX all represent a combination of the Design and Implementation phases of the Software Development Life Cycle.

Chapter X describes the communications between the system, external components, and uniquely, the individual tables in the database. Where complex data manipulations occur, you can also see how that data is collected, and the operations that are performed on it to make it useable in the application. This chapter represents the Implementation phase of the SDLC, as information is provided on the specific interactions of the different parts of the application.

Chapter XI explains some of the initial design decisions, changes to be made to the application, and the benefits and problems associated with some of these design decisions, decisions made out of inexperience, or decisions which posed unintended benefits or consequences.

Chapter X goes through many of the lessons learned over the course of developing the software, and recommendations for others who are looking to construct a software suite for a similar type of audience, similar kinds of information, facing similar challenges, or ultimately, asking similar questions.

Chapters IX and X represent a combination of the Testing and Evolution phases of the Software Development Life Cycle, as some of the issues with the design are brought forth, with recommendations and planned changes for the future.

Chapter XII, finally, goes through some of the terms used through out the document, and how they are to be interpreted in the document, and their significance in understanding the GreekDB system.

II. CONCLUSIONS AND RECOMMENDATIONS

Going through the development process showed that it was possible to develop a software necessary to address the issues of these organizations. GreekDB was able to provide resolution to a number of communicative issues that existed in the chapter through its messaging utilities. The software also proved useful in logging contact information for members and alumni as the chapter prepared for its 50th Anniversary celebration, and many of the reports proved invaluable in preparation of the chapter's annual awards packet. For the awards packet tracking alone, the software eliminated all administrative work required to generate a community service log, collection of an active roster, and generation of an accounts-receivable report. There are many other problems to be addressed for these organizations, where more value can be added for the software, but the software did prove its value in easing productive efforts.

As far as the problem of retaining information across officers, I was able to produce some online tutorials explaining how the officer roles function and how to manage the software for the officers. This eliminated the turnover problems for many of the offices. There were still some tasks that were left to be resolved, but they were outside the scope of the software's initial design.

The following sections describe some of the lessons learned in development of the software. These are design decisions that would have made development easier, or changes that would have made the software more valuable had they been implemented at the start of the development process.

User Activity Logging

One of the biggest things that I wish I had done from the beginning was to implement change logging on each field from the beginning. When starting off, it did not seem like it was very important, but it turned out that it is far more difficult to implement the feature (which is rather important for ensuring data integrity) when I have to go back and add it (versus implementing it as I go. Having this functionality is important not only for security tracking, but also for analytics on the application, as it gives you a better view of how the application is being used, and how the data changes, as opposed to just being stateful.

Designing Permissions Systems

I learned over the course of this project that it is better to take a granular approach to permissions, even when it seems unnecessary. It is much easier to have extremely granular permissions, and to hide the complexity from the user, than to start out with permissions that seem good enough, then to have to write in a new permissions structure later.

Designing Data Structures

Many things need to go into the design of the structure of your databases and their tables. Over the course of the project, I realized that sometimes it does not actually make sense to organize the data structure in a way that makes immediate intuitive sense, like having a member table that relies on a chapter table and an organization table in the way I had it. A number of considerations need to go into the design, including security, and how cascading updates will behave.

Unfortunately, by the time I realized the shortcomings of my design, I was too far into the project and had to keep the structure.

Organizing Code

Over the course of the project, the way I organized code changed dramatically. It started from single PHP files with Style headers and tons of in-line php and css. It then transitioned to reduced in-line code with all functions in the same block and style sheets placed at the bottom of the page. Finally, it transitioned to completely removing any styles and using a standard css file that was stored on a separate server, a template html file, and separate functions files.

Over the course of the project, I had to learn how to organize my code as each page required more and more code and better organization. By the end of it, things turned out well organized since I had a better grasp of what I could do with the language, but some of the older components had an older style to the code, which was not as well organized. For this reason, I highly recommend doing consistent style audits, especially when you are first learning a language.

User-Defined Groups

The feature I received requests for most often was for users to be able to define custom groups of members. Because of some poor design decisions early on, this was not feasible for me to implement. I highly recommend that for anyone developing an application that is even remotely similar to GreekDB, that he/she consider that this may be a possibility. Being able to include this would have added immensely to the capability of the application, but I was simply unable to include the functionality because of my earlier design choices.

Tying Together Different Components

In the application, I needed a quick way to cause judicial case results that resulted in a fine to show the change in balance on the user's landing page. I also needed the

application to be able to handle charging a user, but then having the charge lifted. My solution was to capture the amount as a quantity, as I was already doing, and then having each of the finance-related tools pull in the information from the judicial cases for the calculation steps.

This was a bad idea. It was easy, but it is not a clean solution. The way I should have handled it was to add an extra column to the transaction data I was storing, that had a referrer id that linked the charge to the judicial record, and then had the judicial case create the financial record when the fine was added, and removed it if the fine was taken away. The solution would have required more work, but would have resulted in much cleaner database queries. It also would have allowed me to start storing a lot more information about transactions that are created, and to generate reports and detailed bills.

I highly recommend keeping this in mind when handling utilities that can modify the data for another or multiple utilities. It is better to perform 3 or 4 transactions that are simple, than to set up a complex transaction that you have to work with every time you need the data. In my case, each time I wanted to know what a member's balance was, I had to pull their financial information, and I had to pull the judicial cases where they were found guilty and the punishment was set as monetary. I could have just had to pull their financial records instead.

Use Prepared Statements

At the beginning of the project, I probably learned seven different ways to handle adding and removing data to/from a database in PHP. Some protected against SQL Injection, some did not, some partially did, and some did, but were lossy (they were too aggressive with removing unsafe characters). By the end, I was working exclusively with

prepared statements, and they made the task of performing safe queries, and auto-generating SQL statements much easier than doing them without.

III. ENUMERATED REQUIREMENTS

Overview

The Enumerated Requirements are a list of required features for the software, as well as the individual steps and components that are necessary to meet these requirements.

Requirements

The system will:

R1 Assist Organizations in Maintaining Information about Their Membership

R1.1 Organizations need to track:

R1.1.1 Full Name

R1.1.2 Whether or not the member is an alumnus/alumna

R1.1.3 Whether or not the member is deceased

R1.1.4 Email Address

R1.1.5 The last time the email address was updated

R1.1.6 Website Link

R1.1.7 Facebook Link

R1.1.8 LinkedIn Link

R1.1.9 Company Name

R1.1.10 Phone Number

R1.1.11 The last time the phone number was updated

R1.1.12 Post-Secondary Education Major

R1.1.13 Graduation Year

R1.1.14 Whether or not the member has prior-initiated members in the organization

R1.1.15 Address

R1.1.15.1 Street Address

R1.1.15.2 Street Address (Line 2)

R1.1.15.3 City

R1.1.15.4 State

R1.1.15.5 Zip Code

R1.1.16 Bid Writer

R1.1.17 The date the member accepted his/her offer to join the organization

R1.1.18 The date the member was officially joined the organization

R1.1.19 The member who serves as this member's mentor in the organization

R1.1.20 Miscellaneous, additional information

R1.2 Allow addition of members at any time

R1.3 Organizations need to be able to limit/prevent members from performing certain actions:

R1.3.1 Add new members

R1.3.2 Remove members

R1.3.3 Edit other members' information

R1.3.4 View other members' information

R2 Assist Organizations in Tracking Information about Their Community Involvement

R2.1 Organizaitons need to track:

R2.1.1 Event Title

R2.1.2 Date Performed

R2.1.3 Participating Member

R2.1.4 Quantity

R2.1.5 Donation Type:

R2.1.5.1 Service Hours

R2.1.5.2 $USD

R2.1.5.3 Blood Donation

R2.1.6 Event Contact

R2.1.7 Event Contact Email

R2.1.8 Event Contact Phone Number

R2.2 Allow addition of records at any time

R2.2.1 Allow addition of records for multiple members at a time

R2.3 Allow deletion of records at any time

R2.4 Allow modifying of records at any time

R2.5 Allow organizations to limit/prevent members from performing certain actions:

R2.5.1 Add new events

R2.5.2 Delete events

R2.5.3 Edit events

R2.5.4 View other members' events

R2.5.5 View their own events

R3 <u>Assist Organizations in Maintaining Good Record of Their Judicial Proceedings</u>

R3.1 Organizations need to track judicial information, including:

R3.1.1 Offense Title

R3.1.2 The date the case was opened

R3.1.3 Member Involved

R3.1.4 Case Verdict/Status Information

R3.1.4.1 Pending

R3.1.4.2 Guilty

R3.1.4.3 Innocent

R3.1.4.4 Dismissed

R3.1.5 Case Details

R3.1.6 Punishment Quantity

R3.1.7 Punishment Type

R3.1.7.1 Demerits

R3.1.7.2 Additional Required Service Hours

R3.1.7.3 Monetary Fine

R3.1.7.4 Additional Designated Drive Shifts

R3.1.7.5 Additional Cleaning Details

R3.1.8 Last Date the Case was Updated

R3.1.9 Details About the Decision

R3.2 Organizations need to be able to add cases at any time

R3.3 Guilty Verdicts with monetary punishments need to affect the member's financial records

R3.4 Organizations need to be able to limit/prevent members from performing certain actions:

R3.4.1 Add New Cases

R3.4.2 Edit Cases

R3.4.3 View Other Members' Cases

R4 Allow Organizations to Manage Permissions and Access for Officers and Members

R4.1 Organizations need to be able to edit the permissions for officers at any time

R4.2 Organizations need to be able to edit who has each officer role at any time

R4.3 Organizations need to be able to create their own officers

R4.4 Organizations need to be able to create/use their own permissions structure

R4.5 Organizations need to be able to limit/prevent members from performing certain actions:

R4.5.1 Change the member holding an office

R4.5.2 Change the title of an officer

R4.5.3 Change the permissions for an existing officer role

R4.6 Any changes to officer permissions, member permissions, member-held offices, and any other access permissions must affect the members within a reasonable amount of time

R5 Assist Officers in Communicating with the Organization's Members and Officers

R5.1 Allow officers to send SMS announcements to members

R5.2 Organizations need to be able to send messages to multiple members at a time

R5.2.1 Organizations need to be able to choose the members individually to send to, or to select the entire active membership

R5.3 Organizations need to be able to able to customize the messages they send to members

R5.4 Organizations need to be able to limit/prevent members from sending mass-announcements

R5.5 Organizations need certain tools to allow pre-defined messages to be sent:

R5.5.1 Creating judicial cases needs to be able to notify members that they were involved in a case

R5.5.2 Creating financial transaction records need to be able to notify members that their balance has changed

R6 Help Organizations Improve Financial Tracking and Reporting of Members' Financial Standing

R6.1 Allow officers to track payments by members

R6.2 Allow officers to bill members

R6.3 Allow officers to send balance reminders to members with a non-zero balance

R6.4 Organizations need to be able to limit/prevent members from creating financial transactions

R7 Assist Organizations with Reporting for Internal and External Uses

R7.1 Chapters need to be able to generate the following reports:

R7.1.1 Total number of service hours each member and the organization has contributed

R7.1.2 Total number of punishments and types assigned to each member

R7.1.3 A recent summary of the judicial cases that have been created in the system

R7.1.4 A list of the current outstanding balances of members, and the accounts receivable and payable

R7.1.5 A list of the active members, and a list of all members regardless of membership state

R7.2 Service hours reports, and judicial cases reports need to allow the user to configure the date range

R7.3 The reports need to be printer-friendly

R7.4 The reports should not contain any complicated or excessively colorful graphics.

R8 Provide a Simple Overview for Members and Officers of Important Information

R8.1 Users need information about:

R8.1.1 Their community service totals

R8.1.2 A summary of the results of any judicial cases with a guilty verdict

R8.1.3 The user's current balance

R8.2 Officers need information about:

R8.2.1 The chapter's current community service totals

R8.2.2 A list of the members who have been involved in the most judicial hearings

R8.2.3 The number of outstanding judicial cases

R8.2.4 A list of the members who currently still owe money to the organization

R9 Be Clean

R9.1 The system should respond to the potentially low-resolution devices

R9.1.1 The interface may need to be made more compact for mobile devices

R9.1.2 Any tools/links/information that members don't have access to should not be displayed to those members

IV. APPLICATION MARKET AND REQUIREMENT DRIVERS

Introduction

At first glance, the market for a membership-management/data tracking system for membership-based organizations might seem like a saturated market, or that that kind of platform for fraternities might seem to be a little too niche. In reality, it is a wide-open space, with sparse-scatterings of topical solutions to individual problems such as finances alone, community service-alone, and the occasional tumbleweed of one-size-fits-all-but-does-not-actually-fit-all solutions.

Market

As might have been discerned from the title of the thesis, the target audience for GreekDB is greek-letter organizations, both on the national and local levels.

On the local level, the application seeks to provide a system which individual chapters can improve documentation and ease turnover. The application also seeks to improve member engagement and provide additional, streamlined communication mechanisms for the chapter's officers.

On the national level, the software provides standardized reporting for all chapters using the system, as well as on-demand insight into the chapter's operational information, such community engagement, chapter and member financial status (as they relate to the organization), and more as the application grows to handle more information. Organizations that subscribe are also able to drive behavior in the system as it relates to the chapters.

As the application grows, its focus may expand to other membership-based organizations which are not strictly greek-letter. As it exists currently, nothing in the

system prevents a non-greek organization from using the software; there may be some verbiage, which is foreign or features which are not as useful to these other organizations. For the purpose of this thesis however, the target market should be considered to be greek-letter organizations on the national, regional (where applicable), and local levels.

Revenue Model

The revenue model is broken up into three parts: Free, Mobile, and Paid Subscription. Under the free model, users can spin up their own organization and chapter.

Free Model

The interface will feature targeted ads based on common organization purchases (for example: customized apparel and accessories), and based on other information that is known about the organization/chapter.

An organization cannot manage its chapters under the free model. The chapters behave individually without any governance or reporting to the parent. As a result, an organization could choose to require all chapters to adopt GreekDB on a free subscription, but the organization would have no access to the data apart from reports sent by the individual chapters to them. This helps reinforce the value of a paid subscription for an organization.

Mobile Application

A mobile application is planned that will make interacting with the system much easier for things like community service logging, and other planned features. It will also allow the user a copy of the directory local to their phone, which automatically updates from the application. The mobile application will have features and applications that are

optimized for mobile, and can be obtained by purchasing the app and using it with a free subscription, or is given to the user free via a paid subscription. Regardless of the user's paid or free subscription status, the mobile application will feature no ads.

Paid Subscription

A paid subscription can be purchased by any level of the hierarchy, including the member, chapter, and organization. By default, if a subscription is owned by one level of the organization, and a higher level purchases a subscription, the lower level of the organization will have the remainder of their subscription value refunded, and the highest level will take over.

A paid subscription contains no ads on the website. Additionally, for an organization entering a paid subscription, GreekDB would work with them to consolidate existing chapters that are already using the system under their own banner (but that are a part of the subscribing organization). Organizations will be able to manage information about themselves, such as the descriptions, default color scheme, and other content that is delivered to the end-user or users that are signing up under the organization. Organizations will be able to customize and view reports about chapter operations and finances through their own portal, and generate reports based on information they are interested in.

For chapters adopting a paid subscription, the main difference is the lack of advertisements for end-users, and the ability for members to download the mobile application free. Much of the subscription's value exists on the organizational level, and not the chapter level. Some features may be added in the future to drive up the value of a

chapter subscription, but an organization subscription will always provide the most assets.

For members adopting a paid subscription, the website will feature no ads, and the member will be able to download the mobile application for free.

Competition

Ignoring disjointed tools that are designed to manage a single purpose, such as strictly financing, or strictly membership, there are a couple of outstanding competitors.

ChapterSpot®

The first, a company called ChapterSpot® provides fraternity management utilities to chapters and organizations. They provide both a free tier and a paid tier, where the paid tier costs $30 per member per year, and 2.5% of payment if dues are collected through them. The software provides dues collection, a managed message board, chapter and organization calendars, member tracking, and community service tracking. In addition, chapters can purchase one-time apps that integrate with the platform to handle tasks like recruitment, branding, budgetting, and more. These can be free, all the way up to $500 per purchase.

One of the first distinctions to be drawn between the two platforms is that GreekDB adopts a completely different management model. ChapterSpot® adopts a top-down management model, where specific roles are pre-defined and hard-coded, such as president, vice-president, and treasurer must be represented by an individual within the platform. The tasks that these officers handle are static, and not open to configuration.

GreekDB takes a wholly different approach, and allows an organization to create various roles, assign then to whomever, and configure the role's rights to whatever they

would like. Users can also have multiple roles in the organization and the application will sort out inconsistencies.

In addition, as-of May, 2016, their website is void of any mention of community service tracking. GreekDB not only supports service tracking, but allows multiple configurations for how they are managed:

- A single user can manage all records.
- Multiple users can manage all records.
- All members can submit their own records, but not update the records without going through a manager.
- All members can add or update their own records.

All of those configurations can also be set to allow or disallow other members visibility to other members' records, and the entire GreekDB platform is set up this way. The platform is designed to be administered either from the top-down, or from the bottom-up, according to how the chapter/organization wants it to be managed. This is a far cry from the approach many platforms take.

Tendenci®

Tendenci® is an open-source platform that seeks to provide a more complete management platform for membership management and non-profits. Tendenci® has a much broader feature set than ChapterSpot®, and even GreekDB does currently. The software's features can, in many ways, serve as a model for what features GreekDB should begin to integrate. The software can be downloaded free and hosted on an organization or chapter's own servers. This is its own barrier to entry, as many organization I have met, including other chapters of my own fraternity neither have the

hardware, nor the expertise to manage their own server. My chapter, being located at a technical school, was an exception.

Tendenci® is also more single-chapter-driven, than hierarchly--driven. Meaning the server setup would be required for each instance of the organization (Think Troop 533, not the Boy Scouts of America®). This incurs even more overhead for managing the service, which falls on the shoulders of the organization.

Tendenci® does offer hosted pricing, meaning the organization does not need to manage their own servers. This still causes issues with providing chapter-specific services however, because the software is designed to service the lowest autonomous organization, and not the hierarchy as a whole. This would mean a new subscription for each chapter as well. Currently, their hosted services begin in $169/mo. Their hosted services are also based on hardware requirements, meaning an organization could experience performance degradation if user-driven load is too high.

Finally, upon reviewing the platforms GitHub® page, it appears that out of the software's 29 registered contributors, only 3 are still active. Most of the rest have not made any changes since 2014, and only one person makes consistent contributions to the application's code.

V. APPLICATION ENVIRONMENT AND ACTORS

Introduction to the Environment Model

The environment model is designed to give a simplified view of the communication mechanisms within the system, the communication paths and directions, and the affected/affecting parties of the system.

A unique component to the diagram below is the inclusion of the *Educational Institution Staff*, *National Organization Staff*, and the *Additional 3rd Parties* actors. They do not yet interact directly with the system, but reporting features are included in the system to generate documents for submission to these parties. For parties without a standardized mechanism in place, they can benefit from the reporting system in GreekDB.

Additionally, because of some changes to the system that are performed by/because of the browser that a user is using, it is necessary to include the browser as an intermediate actor between the system and the users/officers, as this effect can be profound in some cases.

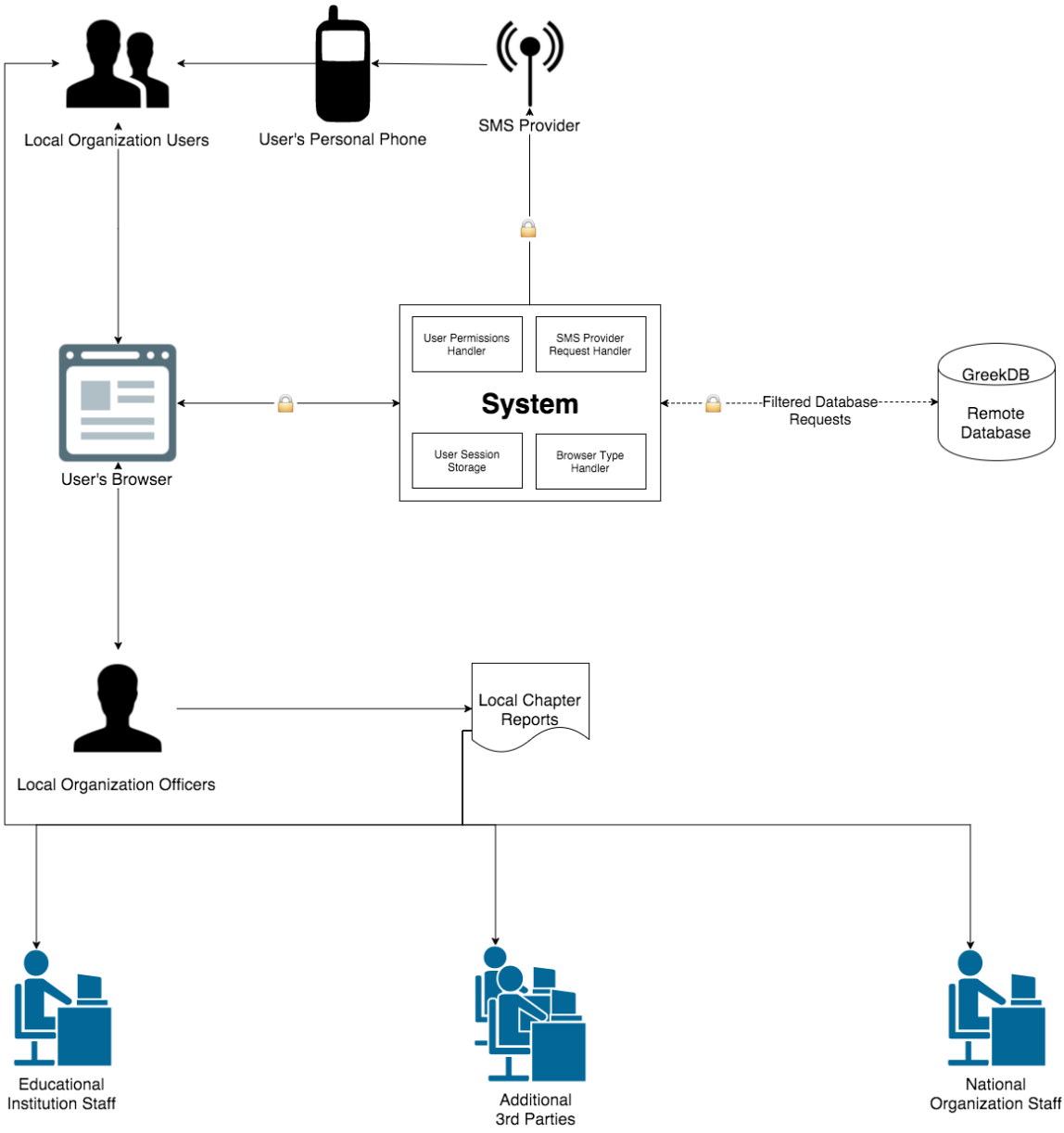


Figure 1. Environment model.

The System

Starting with the System we can see four smaller sub-parts: 3 handlers and 1 storage. These handlers essentially amount to major shared services within the application.

GreekDB needs to be able to communicate with an SMS provider so it can send SMS notifications to members of the organization. This interaction is handled by the

SMS Provider Handler. The connection between the system and the SMS provider (denoted by the antenna icon) for all intents and purposes, one-way (outbound from the system).

For every request the user makes, the system is designed to check that he/she has the appropriate permissions to perform the operation. The User Permissions Handler takes care of this, and ensures that members may only perform actions for which they have been allowed.

The Browser Type Handler checks the users browser to determine if it is a mobile browser or a desktop one. This is necessary as some mobile devices have issues displaying all of the content on a page at once. If it detects a mobile browser, it adjusts the rendering settings to accommodate this. It has the additional benefit of removing some Javascript content, which reduces the page load time and reduces the amount of data sent. This can remove the calendar from the Navigation Banner, reduce the width of the Navigation Banner, and remove the Twitter feeds.

Finally, there is the local session storage, which contains information about the user that is logged in, his/her permissions, his/her organization, and any other data that is accessed too frequently for solely database storage to be reasonable.

User Interaction with the System

The user's browser can affect the user's experience when using GreekDB in a number of ways. Different web browsers can affect how pages appear to the user, and even some features available on forms will work in some browsers, while not in others. A specific example is the **<date>** input type, which is part of the HTML 5 standard. In

some browsers, the date tag is not recognized. In others, it is treated as a text box, and in others still, it renders a full date-selector dropdown when the box is selected.

The communication between the web browser is bidirectional between the System and the user's browser. At this time, there is also no method for the users or officers of an organization to interact directly with the system or its data. They must interact through the web browser. In addition to this, a number of tools allow the delivery of text messages to members' phones. At this time, there is no mobile application, and the SMS provider does not allow reception or processing of text messages, which is why the path from the SMS Provider to the User's Personal Phone is one-way, and the path from the User's Personal Phone is one-way to the Local Organization Users.

The Database

The database is hosted on a server separate from the application, and only allows interaction from the application server and a few specific, white-listed IP addresses, which are necessary for management. In addition, the user that is used by the system to connect to the database has limited permissions to prevent unwanted modification of the database structure. Input sanitization is also performed by the system to protect against SQL Injection attacks. Because requests are made by the system to the database, and results then returned by the database, the communication path is shown as bi-directional. In addition, because of the protections that have been added to the communication between the database and the system, the communication path is dotted, as not all requests may be answered if they are attempting actions that could cause damage.

Reports

Reports are important because they provide a standardized format to distribute information to members of the organization, or any organization or business that may request the information. At this time, there is no method for other organizations to interact directly with the system.

Encryption

There is some information such as email addresses, addresses, and phone numbers that are important and users may not want shared to for others to be able to see. As such, communication to and from The System is encrypted. The lock icon on a path indicated that the communication is encrypted.

VI. USE CASE MODELS

Introduction to the Use Case Models

The Use Case Model is designed to visually capture the "conversation" between actors and the system, and between the different actors through the system. This removes any actors who do not directly interact with the system.

It's important to note that while technically all actions performed by the user go through the web browser before reaching the system, this detail has been left out of the diagram intentionally, as it would make the diagram difficult to understand. Having noted this, the browser does appear as an actor in the 2-Panel Render and 3-Panel Render use cases because it is what is actually interpreting the Cascading Style Sheets (CSS) that are delivered by the system, and drives some of the rendering changes for mobile browsers. Any JavaScript (JS) is also operating on the user's browser (such as the Toggle All functionality), but would again make the diagrams difficult to understand, and are being delivered by the system. For those reasons I've decided to diagram them as though they are a part of the system.

Additionally, because of the important role that permissions play within the system, I've included the permissions handler within the diagram as the double-lined box with rounded corners. Any conversations on the left-hand side of the system that pass through that box depend on the permissions handler to allow them to complete the task.

The communication channels associated with these actions, which may be blocked, are drawn with a dotted line.

Finally, on the right-hand side there are some actions which result in a dotted line to the SMS Provider. This signifies that the user is presented an option to send a message to the SMS Provider, but this action is optional.

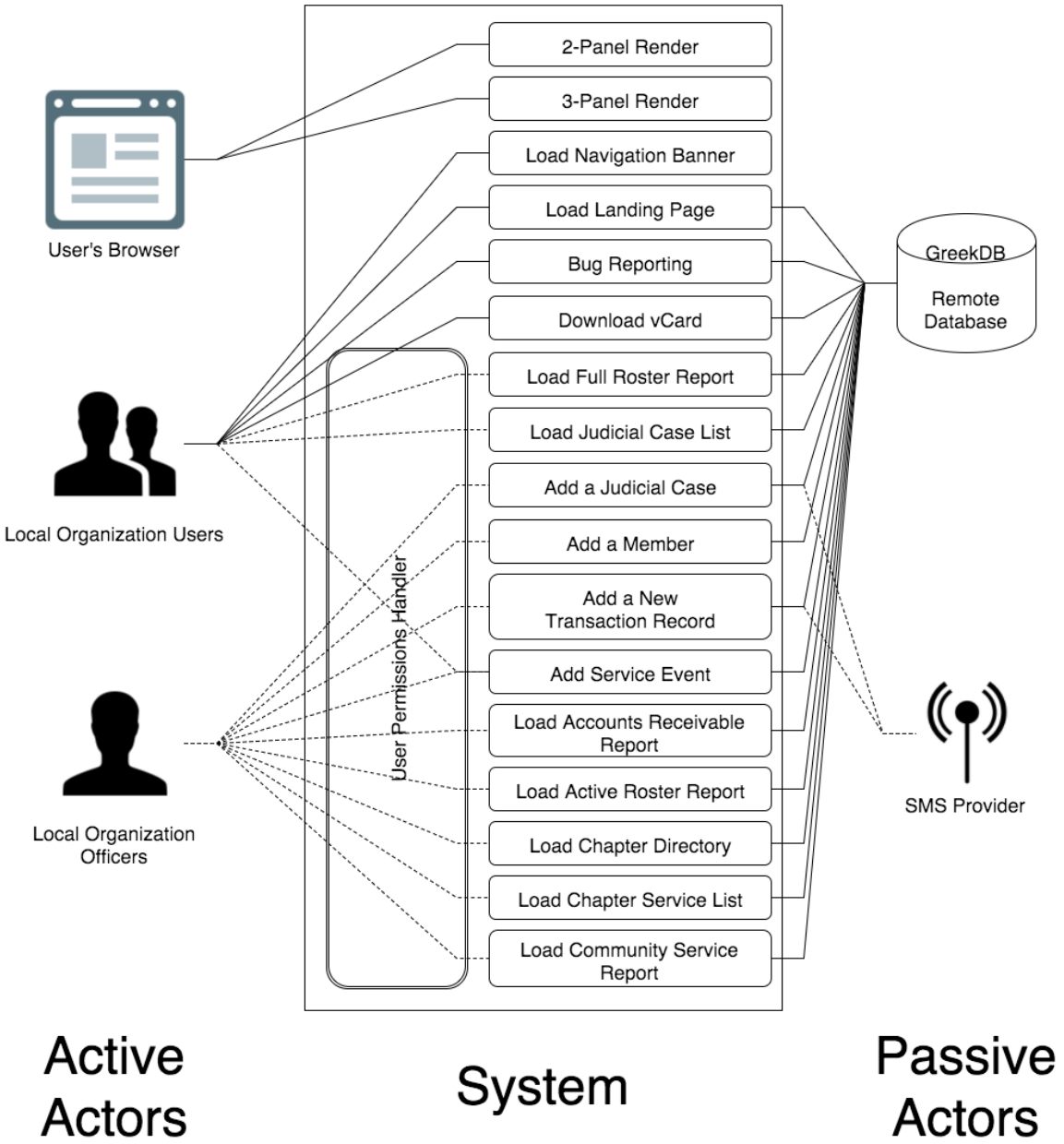


Figure 2. Use case model 1.

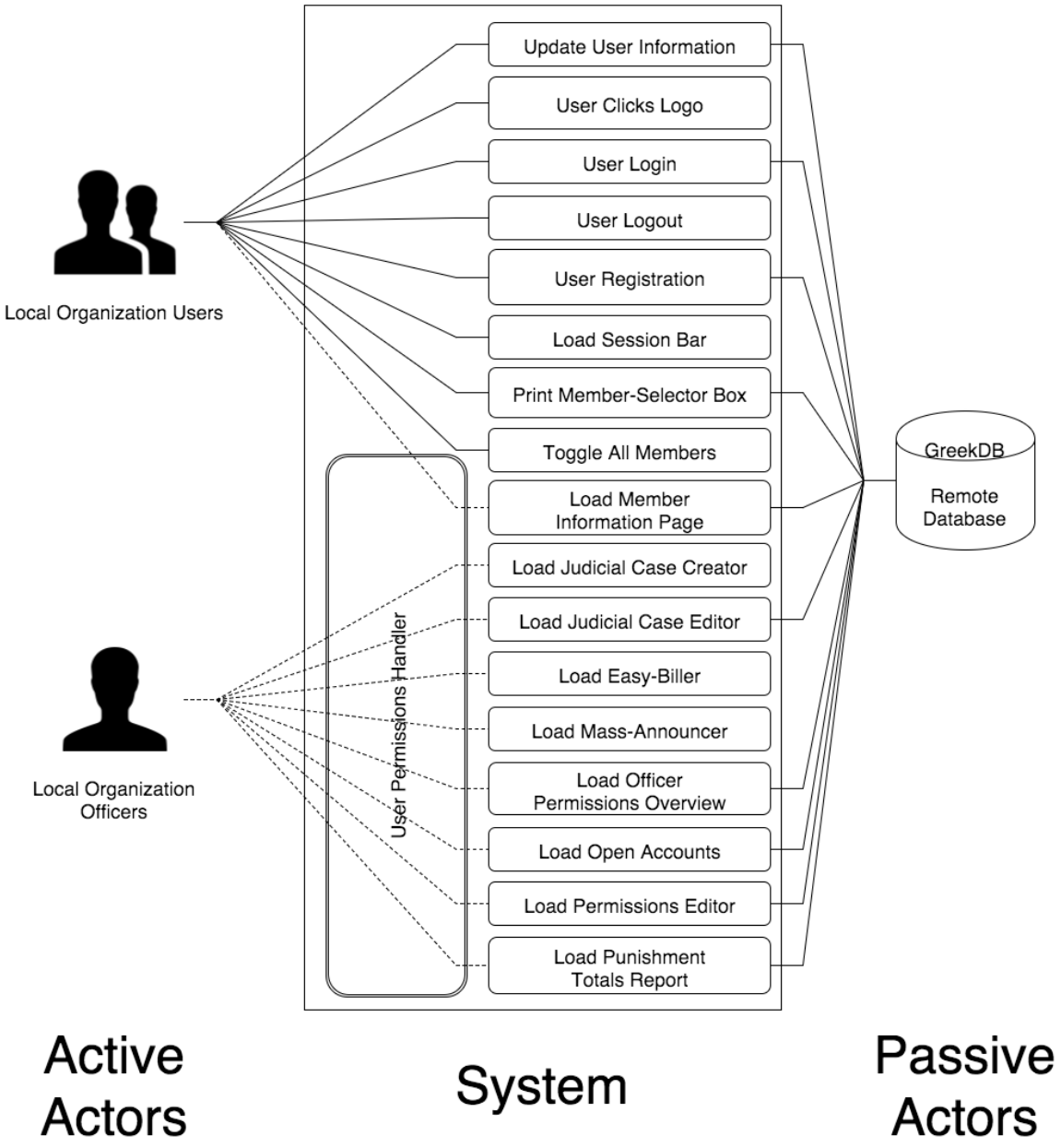


Figure 3. Use case model 2.

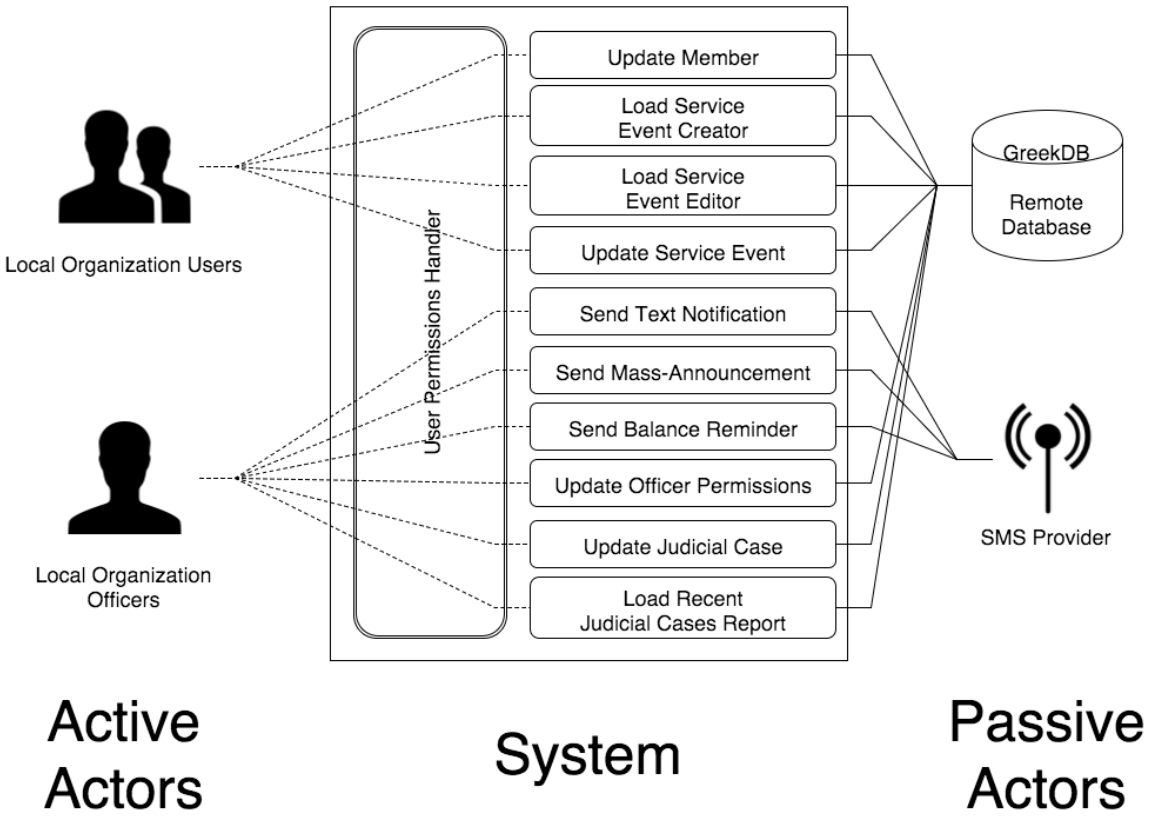


Figure 4. Use case model 3.

VII. USE CASES

Introduction to the Use Case Documents

The documents in this chapter are the Use Case Documents. They describe the different operations required of the system and the actors who interact with it. The documents describe the interactions shown in the figures shown in section 4. The documents contain basic information about the operations that the different actors perform, who is responsible, and what the operations provide to the end-user. The tables contain two columns, the left, which describes the actions of any of the actors (active and/or passive), and the right which describes the actions of the system. There may be more than one table. This corresponds to a break in the process, where either a decision is made or something caused the normal application flow to be no longer valid. There will always be at least one main flow, but there may be additional Alternate Flows to handle additional cases.

2-Panel Render

Name: 2-Panel Render
Summary: Web page renders in a responsive, clean, 2-panel rendering.
Actors: End-user, User's Web Browser
Creation Date: 9/26/2015 **Update Date:** 9/26/2015
Version: 3.0 **Person Responsible:** Aaron Simmons
Path: https://s3.amazonaws.com/greekdb/resources/css/2panel.css
Observed Value: Web page responds to screen resizing and different screen resolutions and aspect ratios in real time.
Precondition: Page is set to render in the 2-panel format.

Main flow 1

Actors	Browser
1. User opens a 2-panel-formatted page, with a horizontal resolution 800px < 1100px in a desktop browser.	
	2. Browser guarantees the middle application panel 500px, and the left-panel 300px. The right-panel renders with the remaining space.
3. User expands horizontal browser resolution beyond 1100px.	
	4. Browser grants left-panel 20% of screen resolution, and application is granted 80%. This fills the remainder of the window. 5. Some objects resize as their parent panel expands, such as the calendar in the left-panel. This is dependent on the page.
6. User shrinks the horizontal browser resolution to < 800px.	
	7. Browser guarantees the middle application panel 500px, and the left-panel 300px, so scroll bars are added to the page to accommodate the reduced resolution.

Main flow 2

Actors	Browser
1. User opens a 2-panel-formatted page, with a horizontal resolution 690px < 990px in a mobile browser.	

	2. Browser guarantees the middle application panel 490px, and the left-panel 200px. The right-panel renders with the remaining space.
3. User expands horizontal browser resolution beyond 990px.	
	4. Browser grants left-panel 20% of screen resolution, and application 60%. This grants the application panel the remainder of the page. 5. Some objects resize as their parent panel expands, such as the calendar in the left-panel. This is dependent on the page.
6. User shrinks the horizontal browser resolution to < 690px.	
	7. Browser guarantees the middle application panel 490px, and the left-panel 200px, so scroll bars are added to the page to accommodate the reduced resolution, and allow scrolling to view hidden content.

3-Panel Render

Name: 3-Panel Render
Summary: Web page renders in a responsive, clean, 3-panel rendering.
Actors: End-user, User's Web Browser
Creation Date: 9/26/2015 **Update Date:** 9/26/2015
Version: 3.0 **Person Responsible:** Aaron Simmons
Path: https://s3.amazonaws.com/greekdb/resources/css/3panel.css
Observed Value: Web page responds to screen resizing and different screen resolutions and aspect rations in real time.
Precondition: Page is set to render in the 3-panel format.

Main flow 1

Actors	Browser
1. User opens a 3-panel-formatted page, with a horizontal resolution 800px < 1100px in a desktop browser.	
	2. Browser guarantees the middle application panel 500px, and the left-panel 300px. The right-panel renders with the remaining space. *Note: Right-panel may be rendered with 0px width.
3. User expands horizontal browser resolution beyond 1100px.	
	4. Browser grants left-panel 20% of screen resolution, and application 60%. Right-panel receives 20%, and the application panel is centered. 5. Some objects resize as their parent panel expands, such as the calendar in the left-panel. This is dependent on the page.
6. User shrinks the horizontal browser resolution to < 800px.	
	7. Browser guarantees the middle application panel 500px, and the left-panel 300px, so scroll bars are added to the page to accommodate the reduced resolution, and allow scrolling to view hidden content. *Note: Right-panel may be rendered with 0px width

Main flow 2

Actors	Browser
1. User opens a 3-panel-formatted page, with a horizontal resolution 690px < 990px in a mobile browser.	
	2. Browser guarantees the middle application panel 490px, and the left-panel 200px. The right-panel renders with the remaining space. *Note: Right-panel may be rendered with 0px width
3. User expands horizontal browser resolution beyond 990px.	
	4. Browser grants left-panel 20% of screen resolution, and application 60%. Right-panel receives 20%, and the application panel is centered. 5. Some objects resize as their parent panel is expanded, such as the calendar in the left-panel. This is dependent on the page.
6. User shrinks the horizontal browser resolution to < 690px.	
	7. Browser guarantees the middle application panel 490px, and the left-panel 200px, so scroll bars are added to the page to accommodate the reduced resolution, and allow scrolling to view hidden content. *Note: Right-panel may be rendered with 0px width.

Add Judicial Case

Name: Add Judicial Case
Summary: From a blank judicial case creator form, a user or administrator can create a new judicial case for 1 or more users.
Actors: End-user, Database, Send Text Notification function
Creation Date: 11/3/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/judicial_case_creator.php
Observed Value: Administrators can log judicial cases for others (provided they have rights to do so).
Precondition: The user must be logged in. The user must also have permission to access the tool.

Main flow

Actors	System
1. The user fills in the offense title, checks the box next to any associated members, updates the verdict, updates the case details, fills in the punishment quantity, updates the punishment type, and the decision details, and clicks Add.	
	2. The system checks that the user has permission to submit a new judicial case. 3. The system sanitizes the user input, and sends the data to the database for storage.
4. The database inserts new records into the Judicial Case table containing the user-submitted data, 1 record for each member that the case was submitted for.	
	5. The system checks the value of the Text Alert checkbox. It is checked. 6. The system calls the "Send Text Notification" function, passing in the first un-sent phone number and a message notifying the user of the newly added case and its result.
7. Send Text Notification processes the supplied phone number and message. **8. Send Text Notification reports completion.**	

	9. The system checks that there are no more messages in-queue. It has completed its queue. 10. The system closes the pop-up window.

Alternate flow 1: the user doesn't have permission to submit new events

Actors	System
	2. The system checks that the user has permission to submit a judicial case. He/she is not. 3. The user is redirected to the No Access page on submission.

Alternate flow 2: there are more messages in-queue

Actors	System
	9. The system checks that there are no more messages in-queue. It finds an additional message. 10. The system halts for 1 second to meet the Nexmo post-frequency requirements. 11. The system returns to step 6 in the Main Flow, with the next phone number in-queue selected.

Alternate flow 3: the text alert checkbox is not checked

Actors	System
	5. The system checks the Text Alert checkbox. It is not checked. 6. The system skips to Step 10 in the Main Flow.

Add Member

Name: Add Member
Summary: A new member profile is added to the database, allowing a user to register a user to that profile.
Actors: End-user, Database
Creation Date: 9/27/15 **Update Date:** 6/20/16
Version: 0.9 **Person Responsible:** Aaron Simmons
Path: https://greek-db.com/memberinfo.php?localid=(new)
Observed Value: The user submits some preliminary information, allowing the application to tie information to that user, and allowing the respective member to register an account to their profile.
Precondition: The user attempting to add a new member must be an administrator to the membership tools. The user must also be logged in.

Main flow

Actors	System
1. The user clicks the Add Member link from the Chapter Directory.	
	2. The system prints a blank Member Information page with the Member ID blank in a new pop-up window.
3. The user fills in the Member ID at a minimum, and then clicks Add.	
	4. The system verifies that a Member ID was entered, and that it is unique. 5. The system sends the data to the database.
6. The Database adds the data to the member table for that organization and chapter.	
	7. The system redirects the user's pop-up window to the new member's profile page.

Alternate flow 1: member ID not entered

Actors	System
3. The user does not fill in a Member ID, and clicks Add.	
	4. The system notifies the user that the Member ID field is required and does not allow submission of the form.

Add Transaction Record

Name: Add Transaction Record
Summary: The system checks the charge/payment selector value. It sends the transaction value to the database with the member id. The database stores the transaction and closes the pop-up window.
Actors: End-user, Database
Creation Date: 11/15/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/easybiller.php
Observed Value: The Easy Biller allows a member, with rights to manage members' financial records, to assign charges and payments to a member's account. The user may also notify the member via text message about the account update.
Precondition: The user must be logged in. He/She must also have permission to modify members' financial records.

Main flow

Actors	System
1. The user chooses a transaction type, selects a quantity, selects members, and clicks submit.	
	2. The system checks that the user has permission to access the page. He/She does.
	3. The system submits a new transaction record for each selected member, to the database, containing the quantity and the member id.
4. The Database stores a transaction record for each member submitted and reports back to the system.	

Alternate flow 1: the user does not have permission to access the page

Actors	System
	5. The system checks that the user has permission to access the page. He/She does.
	6. The system redirects the user to the No Access page.

Add Service Event

Name: Add Service Event
Summary: From a blank service event form, a user or administrator can create a new service event for one or more users.
Actors: End-user, Database
Creation Date: 10/26/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/service_event_creator.php
Observed Value: Users can log community service events for themselves or others, as can administrators, depending on what permissions the organization has allotted for its members.
Precondition: The user must be logged in. The user must also have permission to access the tool.

Main flow

Actors	System
1. The user fills in the service event title, updates the date (if necessary), checks the box next to any participating members, updates the quantity, updates the event type, fills in the description, event contact, and the event contact information, and clicks Add.	
	2. The system checks that the user has permission to submit a new service event. 3. The system sanitizes the user input, and sends the data to the database for storage.
4. The database inserts new records into the Service Event table containing the user-submitted data, 1 record for each member that the event was submitted for.	
	5. The system closes the pop-up window.

Alternate flow: the user doesn't have permission to submit new events

Actors	System
	4. The system checks that the user has permission to submit a new service event. He/she is not.

	5. The user is redirected to the No Access page on submission.

Report Bug

Name: Bug Reporter
Summary: A user reports a bug in the system.
Actors: End-user
Creation Date: 9/19/2015 **Update Date:** 9/19/2015
Version: 0.9 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/bug_reporter.php
https://s3.amazonaws.com/greekdb/resources/sessionbar/icons/bug.png
Observed Value: User can notify developers of any issues they are experiencing with the software.
Precondition: None

Main flow

Actors	System
1. User clicks the little green bug icon on the session bar.	
	2. System prints the bug reporter pop-up.
3. User enters bug report details and an email for support to contact him/her back, then clicks submit.	
	4. System sanitizes user input to protect against SQL Injection. 5. System sends a copy of the bug report to the database.
6. Database stores a copy of the bug report.	
	7. System sends a request to the user's browser to close the bug reporter pop-up.

Download Contact Card (vCard)

Name: Downloadable Contact Card (vCard)
Summary: The system generates a standard vCard for the member currently being viewed, and allows the user to download it.
Actors: End-user, Database
Creation Date: 10/18/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/
Observed Value: The user can pull down contact information from GreekDB for a member and add it to a contact list using the standard contact card (vCard) format. A contact card may be added to a computer address book or transferred to a mobile device. A mobile device will usually download the file and ask if the user would like to download it and import it directly.
Precondition: The user must be logged in. The user must also be coming from a member info page for an existing user. The link will not appear in a blank member window.

Main flow

Actors	System
1. User opens a page for an existing member.	
2. The user clicks the "Download Contact" link at the bottom of the profile.	
	3. The system requests the original member information from the database.
4. The database returns the First Name, Last Name, Phone Number, Email Address, Employer, Street Address (up to 2 lines), City, State, and Zip Code.	
	5. The system creates a temporary .vcf file. 6. The system populates the file with the returned data from the database. 7. The file is closed and renamed to the downloaded member's name. 8. The system launches the download for the user.

Load Accounts Receivable Report

Name: Load Accounts Receivable Report
Summary: The system requests a list of all members with non-zero balances on their accounts from the database. The database returns a list of all members with non-zero balances and their balances. The system prints the list and the sum of all payable accounts, and the sum of all receivable accounts.
Actors: End-user, Database
Creation Date: 11/14/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/reports/financial/accounts_receivable.php
Observed Value: This allows members who have permission to view all members' financial information to get a view of current chapter finances, and to build a standardized report for submission to a national body as needed.
Precondition: The user must be logged in. He/she must have access to view all members' financial information.

Main flow

Actors	System
1. The user clicks the Accounts Receivable link under the reports section of the Navigation Banner.	
	2. The system opens a pop-up window to the Accounts Receivable report.
	3. The system checks that the user has permission to access the page. He/She does.
	4. The system prints the Accounts Receivable Title, followed by the Organization Name and Chapter Designation.
	5. The system requests from the Database, a list of all members with a non-zero balance, and their current balance.
6. The Database looks up all transactions for the organization, and sums all transactions according to the member.	
7. The Database returns a list of members with non-zero balances, their balance, and their Member ID, to the system.	
	8. The System prints a table of the returned members and their balances.

	9. The system prints the sum of the negative values as the Accounts Receivable, and prints the sum of the positive values as the Accounts Payable.

Alternate flow 1: the user does not have rights to access the page

Actors	System
	3. The system checks that the user has permission to access the page. He/she does not. 4. The system redirects the user to the No Access page.

Load Active Roster Report

Name: Generate Active Roster Report
Summary: The system requests a list of the all members who are currently involved in the organization (Active), and their basic contact information and profile image. The system then prints a table of the information in a printer-ready format.
Actors: End-user, Database
Creation Date: 11/14/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/reports/directory/roster.php
Observed Value: The user may retrieve a list of the active membership for sharing contact information, tracking information, reference, and reporting.
Precondition: The user must be logged in. He/she must have permission to read all members' contact information.

Main flow

Actors	System
1. The User clicks the Chapter Roster link in the Reports section of the of the Navigation Banner.	
	2. The system opens a new pop-up window for the Active Roster Report. 3. The system checks that the user has rights to access the page. He/She does. 4. The system prints the Organization Name and the Chapter Designation. 5. The system prints the Active Membership title, which also serves as a hyperlink to the Full Roster report. 6. The system requests a list of all active members and their basic contact information from the database.
7. The database returns a list of all active members in the chapter, including their First Name, Last Name, Major, Phone Number, Email Address, Member ID, and the Image ID for their Profile Picture, to the System.	
	8. The system prints a 3-column record: a. The first column containing the ordered index of the record (1st printed is 1, 2nd printed is

	2, etc.), the Member ID for the member's record, and an empty checkbox, printed. b. The second column contains the member's profile picture, which is retrieved from an image host, identified by an image id to represent the organization, chapter, and member. c. The third column contains the First Name, Major, Phone Number, and Email. 9. The system checks that the next record will not overrun the page. It will not. 10. The system returns to step 8 until each returned member's record has been printed. 11. The system returns to step 8, but prints blank records instead, until the next record would overrun the page and the second column is occupied.

Alternate flow 1: the user does not have permission to access the page

Actors	System
	3. The system checks that the user has rights to access the page. He/She does not. 4. The user is redirected to the No Access page.

Alternate flow 2: the next record will overrun the page and is part of the first column

Actors	System
	9. The system checks that the next record will not overrun the page. It will. 10. The system checks the current column. It is the first column. 11. The system inserts a new column.

	12. The system returns to step 8 until each returned member's record has been printed.

Alternate flow 3: the next record will overrun the page and is part of the second column

Actors	System
	9. The system checks that the next record will not overrun the page. It will. 10. The system checks the current column. It is the second column. 11. The system inserts a pagebreak. 12. The system returns to step 8 until each returned member's record has been printed.

Load Chapter Directory

Name: Load Chapter Directory
Summary: A user loads the chapter directory while logged in.
Actors: End-user
Creation Date: 9/20/2015 **Update Date:** 9/20/2015
Version: 1.2 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/chapter_directory.php
Observed Value: User can view contact and historical information about current and previous members of the organization.
Precondition: User must be logged in to view the chapter directory. If he/she is not, then the banner rendering is prevented by the redirect to the login page. He/she must also be allowed to view all members information. If he/she is granted no access, he/she is redirected to a No Access page before the processing and rendering can occur.

Main flow

Actors	System
1. User loads chapter directory page.	
	2. The system requests the list of current and past members from the database.
3. The database returns id number, first name, last name, email, phone number, graduation year, alumnus/alumna status, and deceased status.	
	4. The system prints this information in a user-legible table, in descending order by id number. 5. While printing the table, users are tagged based on alumni status and deceased status. They are colored blue if they are active members, and greyed out with their links removed if they are deceased.

Alternate flow 1: user has special officer permissions for the chapter directory

Actors	System
	6. The system prints a link to add new members to the organization.

Alternate flow 2: user has restricted rights to view only him/herself

Actors	System
	2. The system queries the database for only the information related to the requesting user.

Load Chapter Service List

Name: Load Chapter Service List
Summary: A logged in user loads the chapter service list so they can view all of the service events that have been logged for the organization.
Actors: End-user, Database
Creation Date: 10/24/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://greek-db.com/chapter_service_list.php
Observed Value: The user may view all logged community service events for the last 6 months. Depending on the permissions applied, the user may only be able to view his/her logged events. A user with admin rights may be able to view all events and open an edit link to modify a previous service record.
Precondition: The user must be logged in. The user must also have permission to view this page.

Main flow

Actors	System
1. The user clicks the Service List link on the Navigation Banner.	
	2. The system checks that the user has permission to access the page. He/she does, so the user is not redirected.
	3. The system requests the last 6 months of community service events from the database.
4. The database returns the last 6 months of events, including the Event Date, Member, Short Description, Quantity, and the Service Type that was logged.	
	5. The system checks what rights the user has to the page. He/she has administrative rights.
	6. The page is printed with the last 6 months events, and an edit link adjacent each item.
	7. The system also prints a link at the bottom of the page to add a new service event.

Alternate flow 1: the user does not have permission to access the page

Actors	System
	5. The system checks that the user has permission to access the page. He/she does not. 6. The system redirects the user to the No Access page.

Alternate flow 2: the user only has access to edit his/her own events, but can view all

Actors	System
	5. The system checks what rights the user has to the page. He/she has permission to view all events, but may only edit his/her own. 6. The system prints the page with all events, but with edit links next to only those that have been created for the user, from within the last 6 months. 7. The system also prints a link to add new events to the bottom of the page.

Alternate flow 3: the user only has access to edit his/her own events, and view no others

Actors	System
	5. The system checks what rights the user has to the page. He/she has permission to edit his/her own events, and cannot see any other members'. 6. The system prints the page with only this user's events, with edit links adjacent each item from the last 6 months. 7. The system also prints a link to add new events to the bottom of the page.

Alternate flow 4: the user only has access to view all

Actors	System
	5. The system checks what rights the user has to the page. He/she has permission to view all events.

	6. The system prints the page with all events from the last 6 months.

Alternate flow 5: the user only has access to view his/her own events

Actors	System
	5. The system checks what rights the user has to the page. He/she has permission to view only his/her events. 6. The system prints the page with the user's events from the last 6 months.

Load Community Service Report

Name: Load Community Service Report
Summary: The user launches the Community Service Report by clicking the Service Totals link under the Reports header on the Navigation Banner. This launches the tool with the start and end-date passed in (some defaults are set initially). The system prints a header for the report as well as information about the organization submitting it. It then prints a list of every active member and the sum of all of the events for which the member was involved, and the totals for the organization at the bottom of the table.
Actors: End-user, Database
Creation Date: 11/11/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/reports/communityservice/memberservicetotals.php?startdate=YYYY-MM-DD&enddate=YYYY-MM-DD
Observed Value: The system generates a list of each member's service totals from within the specified period for reporting and distribution in a concise, printer-friendly document.
Precondition: The user must be logged in. He/she must have permission to read all members' service information.

Main flow

Actors	System
1. The user clicks the Service Totals link.	
	2. The system opens a pop-up window, passing in the current date as the end-date and the date 1 year ago as the start-date to the Community Service Report.
	3. The system checks that the user has permission to access the page. He/she does.
	4. The system requests the Member ID, First and Last Name, service hour total, service event total, blood donation total, and monetary donation total for each active member, beginning with the start-date and stopping after the end-date, from the database.
5. The database pulls the list of all active members and sums each of the service event types for each of those members, from within the	

passed-in dates. It returns the list of members and sum of each of the service event types for each member.	
	6. The system prints the Chapter Service Report title. 7. The system prints the organization name and chapter designation. 8. The system prints a text box, pre-populated with the date 1-year-ago-today, and an end-date text box populated with today's date. 9. The system prints a table containing a row for each active member (even if he/she has no logged records within the time-period), and a column for each the Member ID, Member Name, Service Hours Total, Service Event Total, Blood Donations Total, and Monetary Donations Total.
10. The user changes the Start Date, the End Date, or both and clicks Reload.	
	11. The system continues to step 4 with the new date parameters.

Alternate flow 1: the user does not have permission to access the page

Actors	System
	3. The system checks that the user has permission to access the page. He/she does not. 4. The system redirects the pop-up window to the No Access page.

Load Easy-Biller

Name: Load Easy Biller
Summary: The system prints the title, a drop-down selector to choose a charge or credit, a quantity blank, and calls a function to print a member-selector box, a selector box for sending an SMS notification, and a submission button.
Actors: End-user
Creation Date: 11/15/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/easybiller.php
Observed Value: The Easy Biller allows a member with rights to manage members' financial records, to assign charges and payments to a member's account. The user may also notify the member via text message about the account update.
Precondition: The user must be logged in. He/She must also have permission to modify members' financial records.

Main flow

Actors	System
1. The user clicks the Easy Biller link under the special Treasurer Tools header.	
	2. The system opens a new pop-up window for the Easy-Biller page.
	3. The system checks that the user has permission to access this page. He/She does.
	4. The system prints the Easy Biller title.
	5. The system prints Charge/Payment selector.
	6. The system prints the Quantity blank.
	7. The system calls the function to print the member selector box.
	8. The system prints the Text Notify checkbox.
	9. The system prints the Submit button.

Alternate flow 1: the user does not have permission to access the page

Actors	System
	3. The system checks that the user has permission to access this page. He/She does not.
	4. The system redirects the user back to the No Access page.

Load Full Roster Report

Name: Generate Full Roster Report
Summary: The system requests a list of the all members who have been involved in the organization, and their basic contact information and profile image. The system then prints a table of the information in a printer-friendly format.
Actors: End-user, Database
Creation Date: 11/14/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/reports/directory/roster.php?includeAlumni=TRUE
Observed Value: The user may retrieve a list of the full membership for sharing contact information, tracking information, reference, and reporting.
Precondition: The user must be logged in. He/she must have permission to read all members' contact information.

Main flow

Actors	System
1. The User clicks the Active Membership Title link on the Active Roster.	
	2. The system redirects the Active Roster Report pop-up to the Full Roster Report. 3. The system checks that the user has rights to access the page. He/She does. 4. The system prints the Organization Name and the Chapter Designation. 5. The system prints the Full Roster title, which also serves as a hyperlink to the Active Members Roster report. 6. The system requests a list of all active members and their basic contact information from the database.
7. The database returns a list of all members in the chapter, including their First Name, Last Name, Major, Phone Number, Email Address, Member ID, and the Image ID for their Profile Picture, to the System.	
	8. The system prints a 3-column record: a. The first column containing the ordered

	index of the record (1st printed is 1, 2nd printed is 2, etc.), the Member ID for the member's record, and an empty checkbox, printed. b. The second column contains the member's profile picture, which is retrieved from an image host, identified by an image id to represent the organization, chapter, and member. c. The third column contains the First Name, Major, Phone Number, and Email. 9. The system checks that the next record will not overrun the page. It will not. 10. The system returns to step 8 until each returned member's record has been printed. 11. The system returns to step 8, but prints blank records instead, until the next record would overrun the page and the second column is occupied.

Alternate flow 1: the user does not have permission to access the page

Actors	System
	3. The system checks that the user has rights to access the page. He/she does not. 4. The user is redirected to the No Access page.

Alternate flow 2: the next record will overrun the page and is part of the first column

Actors	System
	9. The system checks that the next record will not overrun the page. It will.

	10. The system checks the current column. It is the first column. 11. The system inserts a new column. 12. The system returns to step 8 until each returned member's record has been printed.

Alternate flow 3: the next record will overrun the page and is part of the second column

Actors	System
	9. The system checks that the next record will not overrun the page. It will. 10. The system checks the current column. It is the second column. 11. The system inserts a pagebreak. 12. The system returns to step 8 until each returned member's record has been printed.

Load Judicial Case Creator

Name: Load Judicial Case Creator
Summary: The user loads the Judicial Case Creator. It is used to add judicial cases for multiple members, speeding up the process of adding case records.
Actors: End-user, Database, Print Member-Selector Function
Creation Date: 11/3/15 **Update Date:** 6/20/16
Version: 1.1 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/judicial_case_creator.php
Observed Value: The Judicial Case Creator allows the user to bulk-add service events. Originally, users had to add cases one-at-a-time through the Service Event Editor. This allows an administrator for the tool to bulk-add cases. Depending on the permission level, the tool will also reduce the multi-select tool to only show the user. An admin can see all active users.
Precondition: The user must be logged in. The user must have permission to access the tool.

Main flow

Actors	System
1. The user clicks the Add Case link.	
	2. The system checks that the user has permission to access the page. He/she does.
	3. The system launches the Judicial Case Creator.
	4. The system prints the page with a place-holder case id and case blank.
	5. The system checks that the user has permission to create cases for all members. He/she does.
	6. The system requests a list of all active members from the database.
7. The database retrieves a list of all active members, including their Member ID, First Name, and Last Name, and returns it to the System.	
	8. The system calls the Print Member-Selector Box function, passing in the list of all active members.
9. The print Member-Selector prints the selector box with the members that were passed in.	
	10. The system prints a verdict selector, offense details textbox, punishment total selector, punishment type

	selector, a decision details textbox, and an Add button.

Alternate flow 1: the user does not have access to the judicial case creator

Actors	System
	2. The system checks that the user has permission to access the page. He/she does not. 3. The user is redirected to the No Access page.

Alternate flow 2: the user does not have permission to create cases for all members

Actors	System
	5. The system checks that the user has permission to create cases for all members. He/she does not. 6. The system skips to Step 8, using the current user's Member ID, First Name, and Last Name as the only input to the function.

Load Judicial Case Editor

Name: Load Judicial Case Editor
Summary: The user loads the Judicial Case Editor. It can be used to view more details about cases when an id is supplied, as well as edit the details for a case depending on the user's permissions on the case record. Depending on the permissions, the page either redirects the user to No Access, renders the page in read-only, or renders it as editable.
Actors: End-user, Database
Creation Date: 11/3/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/judicial_case_editor.php?idjudicialcases=####
Observed Value: The user may view extended case details, or edit an existing case.
Precondition: The user must be logged in. The user must have rights to access the record supplied. The user must have permission to access the tool as well.

Main flow

Actors	System
1. The user clicks the edit link adjacent a service event record on the Judicial Case List.	
	2. The system checks user privileges on the record. He/she has edit rights to this record. 3. The system opens a new pop-up window. 4. The system requests details about the case linked, identified by the Case ID passed in via the URL, from the Database.
5. The database returns the Offense Title, the Date Opened, the Member ID, First and Last Name, Verdict, Offense Details, Punishment Quantity, Punishment Type, Last Updated Date, and Decision Details.	
	6. The system prints the following, all populated with the data from the record: • Case ID in a locked blank • Case Title • Date Opened in a locked blank • Member Selector with the case's member selected

	• A verdict selector with the verdict selected • Offense details • Punishment quantity • Punishment type selector with the case punishment type selected • The last updated date in a locked blank • Decision details in a text block • An Update Button

Alternate flow 1: the user has only view access to the case

Actors	System
1. The user clicks the view link adjacent a service event record on the Chapter Service List	
	2. The system checks user privileges on the record. He/she has read-only rights to this record. 3. The system opens a new pop-up window. 4. The system requests details about the event linked, identified by the Event ID passed in via the URL, from the Database.
5. The database returns the Offense Title, the Date Opened, the Member ID, First and Last Name, Verdict, Offense Details, Punishment Quantity, Punishment Type, Last Updated Date, and Decision Details.	
	6. The system prints the following, all populated with the data from the record, all printed in locked inputs: • Case ID in a locked blank • Case Title • Date Opened in a locked blank • Member Selector with the case's member selected

	• A verdict selector with the verdict selected • Offense details • Punishment quantity • Punishment type selector with the case punishment type selected • The last updated date in a locked blank • Decision details in a text block

Alternate flow 2: the user has no access to the case

Actors	System
	2. The system checks the user privileges on the record. He/she has no access to the record. 3. The system redirects the user to the No Access page.

Load Judicial Case List

Name: Load Judicial Case List
Summary: A logged in user loads the chapter's judicial case list so they can view all of the judicial cases that have been logged for the organization.
Actors: End-user, Database
Creation Date: 11/3/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/judicial_case_list.php
Observed Value: The user may view all logged judicial cases for the last 6 months. Depending on the permissions applied, the user may only be able to view his/her logged cases. A user with admin rights may be able to view all cases and open an edit link to modify a previous case record.
Precondition: The user must be logged in. The user must also have permission to view this page.

Main flow

Actors	System
1. The user clicks the Judicial Cases link on the Navigation Banner.	
	2. The system checks that the user has permission to access the page. He/she does, so the user is not redirected.
	3. The system requests the last 6 months of judicial cases from the database.
4. The database returns the last 6 months of cases, Creation Date, Member Name, Offense Title, Verdict, Punishment Type, and Punishment Quantity.	
	5. The system checks what rights the user has to the page. He/she has administrative rights.
	6. The page is printed with the last 6 months events, and an edit link adjacent each item.
	7. The system also prints a link at the bottom of the page to add a new judicial case.

Alternate flow 1: the user does not have permission to access the page

Actors	System

	2. The system checks that the user has permission to access the page. He/she does not. 3. The system redirects the user to the No Access page.

Alternate flow 2: the user only has access to view all

Actors	System
	5. The system checks what rights the user has to the page. He/she has permission to view all cases. 6. The system prints the page with all cases from the last 6 months, and view links adjacent each.

Alternate flow 3: the user only has access to view his/her own cases

Actors	System
	5. The system checks what rights the user has to the page. He/she has permission to only his/her cases. 6. The system prints the page with the user's cases from the last 6 months, with view links adjacent each.

Alternate flow 4: the user only has access to edit others' cases, but can view all

Actors	System
	5. The system checks what rights the user has to the page. He/she has permission to view all events, but may edit only those of others. 6. The system prints the page with all events, but with edit links next to only those that have been created other members, from within the last 6 months. 7. The system also prints a link to add new events to the bottom of the page.

Load Landing Page

Name: Load Landing Page
Summary: The page loads, presenting information on the member's judicial information, service record, financial status, and the twitter feeds associated with the chapter and its parent organization.
Actors: End-user, Database
Creation Date: 10/31/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/landing.php
Observed Value: This presents valuable information to the user for easier use. It also presents additional information to users who have administrative rights to. Finally, it also presents relevant information from the national organization and local chapter.

Main flow

Actors	System
1. The user clicks the GreekDB logo on the session bar while logged in or logs in initially into the application.	
	2. The system checks that the user is logged in. 3. The system prints a welcome message with the user's name. 4. The system prints the Member Information Title. 5. The system checks that the user can read his/her community service record. He/she can, so the system requests the sum of all community service events for each respective type from the database. 6. The system prints the Community Service header.
7. The Database sums all service records for the last 6 months for each respective type and returns the result to the system.	
	8. The system prints the sums of each service event type returned, with a label next to each. 9. The system checks that the user is able to read his/her judicial case records. He/she can, so the system requests the sum of all Judicial Case

	penalties, organized by the type of punishment levied from the database. 10. The system prints the Standards (judicial) header.
11. The database sums all judicial case records for the last 6 months for each respective type, where the records resulted in a "Guilty" state, organized by the punishment type.	
	12. The system prints the sums of each judicial case punishment returned, with a label next to each. 13. The system prints the Financial header. 14. The system requests the user's account balance from the database.
15. The database sums all charges and payments for the user, and returns the result to the user.	
	16. The system prints the user's balance. 17. The system checks the user's privileges to each the judicial, financial, and philanthropic applications. If he/she has management privileges on any of those applications, the system prints the Officer Information Title. 18. The system checks that the user has admin privileges to the community service tools. 19. The system prints the Community Service header. 20. The system requests the sum of all service event records for the chapter for the last 6 months from the database.
21. The system sums the quantity of each community service record type, according to the record type and returns the sums back to the system.	
	22. The system prints the service totals, with a label adjacent each total. 23. The system checks that the user has admin privileges to the judicial cases tools.

	24. The system prints the Standards (judicial) header. 25. The system requests the sum of all currently pending cases from the database.
26. The database sums all currently open judicial cases and sends the results back to the system.	
	27. The system prints the sum of the open judicial cases. 28. The system requests from the database the list of the 4 active members who have been involved in the most judicial cases within the last 6 months from the database.
29. The database sums the number of guilty cases for each active member within the last 6 months, and returns the 4 members first and last names with the sum to the system, in descending order of guilty cases.	
	30. The system prints the returned members in the returned order, adjacent their case total. 31. The system checks that the user has admin privileges to the financial tools. 32. The system prints the Financial header. 33. The system requests a random set of 4 user who have an outstanding balance or credit on their account from the database.
34. The database sums the financial transactions for each each member in the organization, and returns a list of a random 4 members who have non-zero balances on their account.	
	35. The system prints members' names adjacent their balances. 36. The system checks that the user is on a desktop browser. He/She is. 37. The system prints the Feeds title.

	38. The system prints the Organization header. 39. The system requests the Twitter Handle and the Twitter Widget ID for the organization from the database.
40. The database returns the Twitter Widget ID and the Twitter Handle for the organization.	
	41. The system calls twitter-provided javascript and prints a live twitter feed for the organization. 42. The system prints the Chapter header. 43. The system requests the Twitter Handle and the Twitter Widget ID for the chapter from the database.
44. The database returns the Twitter Widget ID and the Twitter Handle for the chapter.	
	45. The system calls twitter-provided javascript and prints a live twitter feed for the chapter.

Alternate flow 1: the user is not logged in

Actors	System
	2. The system checks that the user is logged in. 3. The user is not logged in. 4. The system redirects the user back to the login page.

Alternate flow 2: the user has no access to view his/her community service record

Actors	System
	5. The system checks that the user has access to view his/her community service record. He/she does not. 6. The system jumps to step 9.

Alternate flow 3: the user has no access to view his/her judicial record

Actors	System

	9. The system checks that the user has access to view his/her community service record. He/she does not. 10. The system jumps to step 13.

Alternate flow 4: the user does not have admin access to any of the community service, judicial, and financial tools

Actors	System
	17. The system checks the user's privileges to each the judicial, financial, and philanthropic applications. He/She has no management privileges on any of those applications. 18. The system jumps to step 36.

Alternate flow 5: the user does not have admin access to the community service tools

Actors	System
	18. The system checks that the user has admin privileges to the community service tools. He/She has no admin privileges to the community service tools. 19. The system jumps to step 23.

Alternate flow 6: the user does not have admin access to the judicial tools

Actors	System
	23. The system checks that the user has admin privileges to the judicial tools. He/She has no admin privileges to the judicial tools. 24. The system jumps to step 31.

Alternate flow 7: the user does not have admin access to the financial tools

Actors	System
	31. The system checks that the user has admin privileges to the financial

	tools. He/She has no admin privileges to the financial tools. 32. The system jumps to step 36.

Alternate flow 8: the user is on a mobile browser

Actors	System
	36. The system checks that the user is on a desktop browser. He/she is on a mobile browser. 37. The system skips rendering Twitter Feeds (Steps 37-45)

Load Mass-Announcer

Name: Load Mass-Announcer
Summary: The system prints an announcement block for the user to fill in his/her message and a multi-user selector block.
Actors: End-user, Database, Print Member-Selector Box function
Creation Date: 11/5/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/announcer.php
Observed Value: Members can send mass-SMS messages to the members of the chapter
Precondition: The user must be logged in. He/she must have permission to access the page.

Main flow

Actors	System
1. The user click the Mass-Announcer link in the navigation banner.	
	2. The system checks that the user has permission to access the page. He/she does. 3. The system prints the Bulk Announcer title. 4. The system prints the message block pre-populated with "Announcer:" text. 5. The system requests a list of all active members from the database.
6. The database retrieves a list of all active members, including their Member ID, First Name, and Last Name, and returns it to the System.	
	7. The system calls the Print Member-Selector Box function, passing in the list of all active members.
8. The print Member-Selector prints the selector box with the members that were passed in.	
	9. The system prints the Submit button.

Alternate flow 1: the user does not have permission to access the page

Actors	System
	2. The system checks that the user has permission to access the page. He/she does not.

	3. The system redirects the user to the No Access page.

Load Member Information Page

Name: Load Member Information Page
Summary: **The user launches the Member Information page, which is used for viewing/editing users and creating new users.**
Actors: End-user, Database
Creation Date: 9/27/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/memberinfo.php?localid=####
Observed Value: A blank user information page is launched if the user clicked Add User, or the member profile for the requested member is displayed as allowed by the chapter's permissions. The member id is included in the URL to make it easy to bookmark profiles.
Precondition: User must be logged in, otherwise the user is redirected to the login page. The user must also have read privileges to whatever user is being requested, as well as admin access to the directory to add new members, otherwise the user is redirected to the No Access page before the page is rendered.

Main flow

Actors	System
1. The user clicks on the Member ID from the directory page, navigation banner, or opens a direct link to a member's page.	
	2. The system checks that the page is not for the same Member ID as the browsing user. It is not. 3. The system checks that the user has admin rights. He/she does not. 4. The system checks that he/she has permission to view that page. He/she does. 5. The system queries the database for the completed profile of the member who was requested, from the same organization and chapter as the logged in user.
6. The database returns the requested information.	
	7. The system checks whether the user has read or read-write permissions to the page. He has read-only permissions to the page. 8. The system prints the page with contents locked. 9. The system prints the Download link.

Alternate flow 1: admin opens a link to a member profile.

Actors	System
	3. The system checks that the user has admin rights. He/she does. 4. The system queries the database for the completed profile of the member who was requested, from the same organization and chapter as the logged in user.
5. The database returns the requested information.	
	6. The system prints the page with contents unlocked. 7. The system prints an Update button at the bottom of the page. 8. The system prints the Download link.

Alternate flow 2: user opens a link to his/her own profile

Actors	System
	2. The system checks that the requested Member ID does not match the user's. It does match. 3. The system queries the database for the completed profile of the member who was requested, from the same organization and chapter as the logged in user.
4. The database returns the requested information.	
	5. The system prints the page with contents unlocked, and an Update button.

Alternate flow 3.1: user opens a link to a member profile which does not exist

Actors	System
1. User tries to open a profile he/she has no access to, or tries to fuzz the profile page.	
	2. The system redirects the user to the No Access page.

Alternate flow 3.2: admin opens a link to a member profile which does not exist

Actors	System
1. The admin tries to open a profile with a valid Member ID format, but for a member that does not exist.	
	2. The system redirects the user to the No Access page.

Alternate flow 4.1: admin opens a link to a new profile: "?localid=(new)"

Actors	System
1. The Admin opens a link to an empty, new profile.	
	2. The user's permissions are checked to make sure he/she is an administrator. He/she is. 3. The page is rendered with all information blank, but modifiable, and with a submit button.

Alternate flow 4.2: user opens a link to a new profile: "?localid=(new)"

Actors	System
1. The User opens a link to an empty, new profile.	
	2. The user's permissions are checked to make sure he/she is an administrator. He/she is not. 3. The user is redirected to the No Access page.

Load Navigation Banner

Name: Load Navigation Banner
Summary: A user loads a page while logged in (non-popup). The page loads the navigation banner to help the user move around the site.
Actors: End-user
Creation Date: 9/20/2015 **Update Date:** 9/20/2015
Version: 1.3 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/commonfiles/navigationbanner.php
Observed Value: User account now has a navigation panel to help them navigate the site's pages to which they have access, as well as visual indicators of their permission levels on those pages.
Precondition: User must be logged in to view the navigation banner. If he/she is not, then the banner rendering is prevented by the redirect to the login page.

Main flow

Actors	System
1. User loads any 2-panel or 3-panel page.	
	2. The system checks that the user is not using GreekDB from a mobile device. He/she is not.
	3. The system requests the calendar embed information from the database.
4. The database returns the calendar embed parameters if it exists for the chapter.	
	5. The system looks up the users' page permissions from the session information.
	6. The system prints the "My Profile" link.
	7. The system prints links to the tools the user has normal access to.

Alternate flow 1: user is using greekDB from a mobile device

Actors	System
	2. The system checks that the user is not using GreekDB from a mobile device. He/she is.
	3. The system does not lookup or print the calendar to reduce the horizontal footprint of the page.
	4. The system skips to step 5.

Alternate flow 2: user has administrative rights to a linked tool

Actors	System
	7. The system prints an “Officer Tools” header and prints the links to that and any other tools where the user has admin access beneath it.

Alternate flow 3: user has no rights to the linked tool

Actors	System
	7. The system does not print a link to the tools to which the user has no access.

Load Officer Permissions Overview

Name: Load Officer Permissions Overview
Summary: **The system retrieves all officers created for the organization, and provides a table of their permissions and links to pop-ups to edit the officers' permissions.**
Actors: End-user, Database
Creation Date: 11/3/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/officer_permissions_manager.php
Observed Value: This allows users to view what permissions the various chapter officers hold and to edit them if they have permission to do so.
Precondition: The user must be logged in. The user must have permission to edit officer permissions.

Main flow

Actors	System
1. The user clicks the Officer Permissions link in the navigation banner.	
	2. The system checks that the user has rights to access the page. He/she does. 3. The system requests the list of officers and their permissions from the database.
4. The database retrieves all of the officers for the chapter, including their permission levels for each tool, and returns them to the system.	
	5. The system prints a table of the the Officer Titles, and their associated tool permissions in a 2-panel render. Each of the Officer Titles is a link which brings up a pop-up to change the officer's permissions.

Alternate flow 1: the user does not have rights to access the page

Actors	System
	2. The system checks that the user has rights to access the page. He/she does not.

	3. The system redirects the user to the No Access page.

Load Open Accounts

Name: Load Open Accounts
Summary: The system pulls a list of all members who have an outstanding balance owed to the chapter, presents their name and quantity owed, and provides a link to send an SMS reminder to the member of there standing balance.
Actors: End-user, Database
Creation Date: 11/14/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/open_accounts.php
Observed Value: This allows a member with edit access to members' financial records to notify members of their current financial state.
Precondition: The user must be logged in. He/she must also be allowed to read and edit members' financial information.

Main flow

Actors	System
1. The user clicks the Open Accounts link under the special Treasurer Tools header.	
	2. The system checks that the user has permission to access the page. He/She does. 3. The system prints the Balance-Holding Members title. 4. The system requests from the database, a list of all members (current and inactive) who have an outstanding balance owed to the chapter, and their balance.
5. The database sums the transactions for the chapter according to the member, and returns to the system, a list of the members with outstanding balances owed, and the amount owed.	
	6. The system prints a 3-column table of all members with an outstanding balance, where the first column contains a reminder link, the second is the member name, and the third is the outstanding charges quantity. 7. The final row contains two columns: The first contains the number of

	members who have unpaid expenses. The second is the sum of the money owed to the chapter.

Alternate flow 1: the user does not have permission to access the page.

Actors	System
	2. The system checks that the user has permission to access the page. He/She does not. 3. The system redirects the user to the No Access page.

Load Permissions Editor

Name: Load Officer Permissions Editor
Summary: The system looks up the selected officer and his/her permission, and returns the values for the user to modify.
Actors: End-user, Database
Creation Date: 11/3/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/officer_permissions_editor.php?officer=##
Observed Value: Users may update officer permissions to help with data management.
Precondition: The user must be logged in. The user must also have permission to access the page.

Main flow

Actors	System
1. The user clicks an Officer Title link on the Officer Permissions Overview.	
	2. The system checks that the user has rights to access the page. He/she does.
	3. The system requests the selected officer and his/her permissions from the database.
4. The database selects the officer title and the officer's permissions for that officer and returns the values to the system.	
	5. The system prints the Officer Title as the page header.
	6. The system prints an editable blank containing the officer's current title, a member selector, and a selector for each of the tools permissions, with the current permission level for each tool selected, and an Update Officer button.

Alternate flow 1: the user does not have rights to access this page

Actors	System
	2. The system checks that the user has rights to access the page. He/she does not.

	3. The system redirects the user to the No Access page.

Load Punishment Totals Report

Name: Load Punishment Totals Report
Summary: The user launches the Punishment Totals Report by clicking the Punishment Totals link under the Reports header on the Navigation Banner. This launches the tool with the start and end-date passed in (some defaults are set initially). The system prints a header for the report as well as information on the organization submitting it. It then prints a list of every active member, and the sum of all of the punishments levied on the members where the cases resulted in a guilty decision.
Actors: End-user, Database
Creation Date: 11/11/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/reports/judicialcases/punishment_totals.php?startdate=YYYY-MM-DD&enddate=YYYY-MM-DD
Observed Value: The system generates a list of each member's punishment totals from within the specified period for reporting and distribution in a concise, printer-friendly document.
Precondition: The user must be logged in. He/she must have permission to read all members' judicial information.

Main flow

Actors	System
1. The user clicks the Punishment Totals link.	
	2. The system opens a pop-up window, passing in the current date as the end-date and the date 1 year ago as the start-date to the Punishment Totals Report.
	3. The system checks that the user has permission to access the page. He/she does.
	4. The system requests the Member ID, First and Last Name, Additional Required Service Hours, Demerits/Marks/Checks, Monetary Fines, and Additional Cleaning Details for each active member, beginning with the start-date and stopping after the end-date, from the database.

5. The database pulls the list of all active members and sums each of the judicial case punishment types for each of those members, from within the passed-in dates. It returns the list of members and sum of each of the judicial case punishment types for each member.	
	6. The system prints the Punishment Totals Report title. 7. The system prints the organization name and chapter designation. 8. The system prints a text box, pre-populated with the date 1-year-ago-today, and an end-date text box populated with today's date. 9. The system prints a table containing a row for each active member (even if he/she has no logged records within the time-period), and a column for each the Member ID, Member Name, Additional Required Service Hours, Demerits/Marks/Checks, Monetary Fines, and Additional Cleaning Details.
10. The user changes the Start Date, the End Date, or both and clicks Reload.	
	11. The system continues to step 4 with the new date parameters.

Alternate flow 1: the user does not have permission to access the page.

Actors	System
	3. The system checks that the user has permission to access the page. He/she does not. 4. The system redirects the pop-up window to the No Access page.

Load Recent Judicial Cases Report

Name: Load Recent Judicial Cases Report
Summary: The user launches the Judicial Cases Report by clicking the Recent Cases link under the Reports header on the Navigation Banner. This launches the tool with the start and end-date passed in (some defaults are set initially). The system prints a header for the report as well as information on the organization submitting it. It then prints a list of every case and the resulting punishment (if a guilty verdict was reached) from within the specified time period.
Actors: End-user, Database
Creation Date: 11/11/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/reports/judicialcases/case_list.php?startdate=YYYY-MM-DD&enddate=YYYY-MM-DD
Observed Value: This allows a user to get a quick overview of the cases that have been seen recently by the judicial committee and the result of each, when additional details are not necessary.
Precondition: The user must be logged in. The user must also have permission to view every member's judicial cases.

Main flow

Actors	System
1. The user clicks the Recent Cases link under the Reports header.	
	2. The system opens a pop-up window, passing in the start and end-date to the Recent Cases report.
	3. The system checks that the user has permission to access the page. He/she does.
	4. The system requests the Case Opened Date, First and Last Name, Offense Title, Verdict, Punishment Quantity, and Punishment Type for cases which occurred in after the start-date and before the end-date from the database.
5. The database collects and returns the Case Opened Date, First and Last Name, Offense Title, Verdict, Punishment Quantity, and Punishment Type for cases which occurred in after the start-date and before the end-date, and returns them to the System.	

	6. The system prints the Judicial Cases Report header. 7. The system prints the organization name and chapter designation. 8. The system prints a text box, pre-populated with the date 1-year-ago-today, and an end-date text box populated with today's date. 9. The system prints a table including a column for each: the Case Opened Date, the Member name, the Case Title, the Verdict, and the Punishment. It prints a row for each case found between the two dates.
10. The user changes the Start Date, the End Date, or both and clicks Reload.	
	11. The system continues to step 4 with the new date parameters.

Alternate flow 1: the user does not have permission to access this page

Actors	System
	3. The system checks that the user has permission to access the page. He/she does not. 4. The system redirects the pop-up window to the No Access page.

Load Service Event Creator

Name: Load Service Event Creator
Summary: The user loads the Service Event Creator. It is used to add community service events for multiple members, speeding up the process of adding service records.
Actors: End-user
Creation Date: 10/25/15 **Update Date:** 6/20/16
Version: 1.2 **Person Responsible:** Aaron Simmons
Path: https://greek-db.com/service_event_creator.php
Observed Value: The Service Event Creator allows the user to bulk-add service events. Originally, users had to add events individually through the Service Event Editor. This allows an administrator for the tool to bulk-add events. Depending on the permission level, the tool will also reduce the multi-select tool to only show the user. An admin can see all active users.
Precondition: The user must be logged in. The user must have permission to access the tool.

Main flow

Actors	System
1. The user clicks the Add Event link.	
	2. The system checks that the user has permission to access the page. He/she does.
	3. The system launches the Service Event Creator.
	4. The system prints the page with a place-holder event id, a blank service event title blank, a date blank, pre-populated with today's date.
	5. The system checks that the user has permission to create events for all members. He/she does.
	6. The system requests a list of all active members from the database.
7. The database retrieves a list of all active members, including their Member ID, First Name, and Last Name, and returns it to the System.	
	8. The system calls the Print Member-Selector Box function, passing in the list of all active members.
9. The print Member-Selector prints the selector box with the members that were passed in.	

	10. The system prints a blank Quantity, Type, Description, Event Contact, and Event Contact Info box, and an Add button.

Alternate flow 1: the user does not have access to the service event creator

Actors	System
	2. The system checks that the user has permission to access the page. He/she does not. 3. The user is redirected to the No Access page.

Alternate flow 2: the user does not have permission to create events for all members

Actors	System
	5. The system checks that the user has permission to create events for all members. He/she does not. 6. The system skips to Step 8, using the current user's Member ID, First Name, and Last Name as the only input to the function.

Load Service Event Editor

Name: Load Service Event Editor
Summary: The user loads the Service Event Editor. It can be used to view more details on community service events when an id is supplied, as well as edit the details for an event depending on the user's permissions on the event record. Depending on the permissions, the page either redirects the user to No Access, renders the page in read-only, or renders it as editable.
Actors: End-user, Database
Creation Date: 10/25/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://greek-db.com/service_event_editor.php?idevent=####
Observed Value: The user may view extended event details, or edit an existing event.
Precondition: The user must be logged in. The user must have rights to access the record supplied. The user must have permission to access the tool as well.

Main flow

Actors	System
1. The user clicks the edit link adjacent a service event record on the Chapter Service List.	
	2. The system checks user privileges on the record. He/she has edit rights to this record. 3. The system opens a new pop-up window. 4. The system requests details about the event linked, identified by the Event ID passed in via the URL, from the Database.
5. The database returns the Service Event Title, the Date Performed, the Member ID, First and Last Name, Quantity, Record Type, Description, Event Contact, and Coordinator Contact Information.	
	6. The system prints the Event ID in a locked blank, the Event Title, the Date Performed, a Member Selector with the Event's Member selected, the Service Quantity, the Service Record Type, Description, Event Contact, Event Contact Information, an Update Button, all populated with the data from the record, and a link to delete the event.

Alternate flow 1: the user has only view access to the record

Actors	System
1. The user clicks the view link adjacent a service event record on the Chapter Service List.	
	2. The system checks user privileges on the record. He/she has read-only rights to this record. 3. The system opens a new pop-up window. 4. The system requests details about the event linked, identified by the Event ID passed in via the URL, from the Database.
5. The database returns the Service Event Title, the Date Performed, the Member ID, First and Last Name, Quantity, Record Type, Description, Event Contact, and Coordinator Contact Information.	
	6. The system prints the Event ID, the Event Title, the Date Performed, a Member Selector with the Event's Member selected, the Service Quantity, the Service Record Type, Description, Event Contact, Event Contact Information, all populated with the data from the record in locked text boxes/selectors.

Alternate flow 2: the user has no access to the record

Actors	System
	2. The system checks the user privileges on the record. He/she has no access to the record. 3. The system redirects the user to the No Access page.

Load Session Bar

Name: Load Session Bar
Summary: A user loads a page while logged in (non-popup). The page loads the session bar to help the user move around the site, indicate their login status, allow them to manage their account, and log out.
Actors: End-user
Creation Date: 9/26/2015 **Update Date:** 9/26/2015
Version: 1.3 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/commonfiles/globalnavigationbar.php
Observed Value: User account now has session bar to help them manage their user, log out, log in, and navigate the site.
Precondition: None.

Main flow

Actors	System
1. User loads any 2-panel or 3-panel page.	
	2. The system checks that the user is logged in. He/she is.
	3. System prints logo to link back to landing page.
	4. System prints Edit Account link
	5. System prints Log Out link.

Alternate flow 1: user is not logged in

Actors	System
	2. The system checks that the user is logged in. He/she is not.
	3. System prints the logo as a link to the home page.
	4. System prints Guests link to present information on GreekDB.
	5. System prints Login/Sign Up link to take users to the login page.

Print Member-Selector Box

Name: Print Member-Selector Box
Summary: The system looks up the current active membership, and prints a checkbox adjacent each name and ID, in a 5-column wide independently-scrollable box.
Actors: End-user, Database
Creation Date: 11/18/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: {Internal Function}
Observed Value: The Member-Selector Box allows a user to apply changes to multiple members at a time, without submitting multiple copies of the form.
Precondition: The user must be logged in. He/she must have permission to access a page which calls this function. The function must be called by the system, and cannot be accessed directly by a user.

Main flow

Actors	System
1. Another tool calls this function, providing a list of members to include.	
	2. The system prints a 5-column wide table of members, including their Member ID, First and Last Name, and a checkbox which is tied to that member, all start deselected.
	3. At the end of the table the system prints an additional row that contains a single Toggle All checkbox.

Send Balance Reminder

Name: Send Balance Reminder
Summary: The system accepts the member id and balance owed, and requests the phone number from the database. The system send a pre-defined message about the user's balance to the SMS service provider, who sends the message to the user.
Actors: End-user, Database
Creation Date: 11/14/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/open_accounts.php
Observed Value: A user with permission to manage financial information may notify a member that they have a balance owed to the chapter.
Precondition: The user must be logged in. He/she must have permission to view and edit member financial information.

Main flow

Actors	System
1. The user clicks the Send Reminder link on the Open Accounts page.	
	2. The system checks that the member has permission to access the page. He/she does.
	3. The system requests the phone number for the member adjacent the link from the database.
4. The database retrieves and returns the phone number for the requested member.	
	5. The system removes any non-numeric characters from the returned phone number.
	6. The system checks that the phone number is the correct length. It is.
	7. The system checks the phone number for a country code. One is found.
	8. The system url-encodes the api key, api secret, source phone number, destination phone number, and the message text.
	9. The system makes an encrypted post to the Nexmo service who processes the information and sends a text message on GreekDB's behalf.
	10. The system checks that there are no more messages in-queue. It has completed its queue.

	11. The system notifies the user that the message has been sent.

Alternate flow 1: the user does not have access to the page

Actors	System
	2. The system checks that the member has permission to access the page. He/she does not. 3. The system redirects the user to the No Access page.

Alternate flow 2: the phone number does not include the corresponding country code

Actors	System
	7. The system checks the phone number for a country code. One is not found. 8. The system assumes the phone number to originate in the US, and so pre-pends a 1 to the phone number. 9. The system returns to step Step 8 in the Main Flow.

Alternate flow 3: the phone number is too short

Actors	System
	6. The system checks that the phone number is the correct length. It is too short. 7. The system ignores the number.

Send Mass-Announcement

Name: Send Mass-Announcement
Summary: The system pulls the phone numbers for each of the selected members. It then corrects phone numbers in improper formats, encodes the announcement text, creates an https connection to the Nexmo service (responsible for sending the sms messages), and sends one message at a time to comply with service requirements, and displays a wait message for the user until the process completes, then reloads the announcer page.
Actors: End-user, Database, Send Text Notification function
Creation Date: 11/5/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/announcer.php
Observed Value: Members and administrators can send announcement text messages to a list of active members about important news or information.
Precondition: The user must be logged in. He/she must also have permission to access the page.

Main flow

Actors	System
1. The user fills out an sms message and checks the box adjacent one or more members and clicks Submit.	
	2. The system checks that the user has permission to use the tool. He/she is. 3. The system requests from the database a list of the phone numbers for each member selected to be sent a message.
4. The database pulls the phone number for each of the members passed in from the member information table and returns the results to the system.	
	5. The system requests that the user not reload the page as it processes the messages. 6. The system calls the "Send Text Notification" function, passing in the first unsent phone number and the user-supplied message.
7. Send Text Notification processes the supplied phone number and message. **8. Send Text Notification reports completion.**	

	9. The system checks that there are no more messages in the queue. It has completed its queue. 10. The system notifies the user that the messages have been sent. 11. The system reloads the page.

Alternate flow 1: the user does not have access to the page

Actors	System
	2. The system checks that the user has permission to use the tool. He/she does not. 3. The system redirects the user to the No Access page.

Alternate flow 2: there are more messages in-queue

Actors	System
	9. The system checks that there are no more messages in-queue. It finds an additional message. 10. The system halts for 1 second to meet the Nexmo post frequency requirements. 11. The system returns to step 6 in the Main Flow, with the next phone number in-queue selected.

Send Text Notification

Name: Send Text Notification
Summary: The system looks up the recipient's phone number, checks that the number is valid and can be sent to. It then encodes the input and sends it off to an SMS provider, who sends the text message to the user.
Actors: End-user, Database, SMS Provider
Creation Date: 11/16/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: {Internal Function}
Observed Value: This allows various tools in the system to send SMS notifications to members of the organization to inform them of any important information.
Precondition: The user must be logged in. He/she must have permission to access a tool that features text notifications. He/she must also be sending the notification to a member that has a valid cell phone number populated. The function is only called by another tool, and cannot be called directly by the user.

Main flow

Actors	System
1. Another tool calls this function, providing a message and a destination phone number.	
	2. The system removes any non-numeric characters from the returned phone number.
	3. The system checks that the phone number is the correct length. It is.
	4. The system checks the phone number for a country code. One is found.
	5. The system url-encodes the api key, api secret, source phone number, destination phone number, and the message text.
	6. The system makes an encrypted post to the Nexmo service who processes the information and sends a text message on GreekDB's behalf.
	7. The system responds back to the calling function indicating completion.

Alternate flow 2: the phone number is too short

Actors	System

	3. The system checks that the phone number is the correct length. It is too short. 4. The system ignores the number and jumps to step 7.

Alternate flow 3: the phone number does not include the corresponding country code

Actors	System
	4. The system checks the phone number for a country code. One is not found. 5. The system assumes the phone number to originate in the US, and so pre-pends a 1 to the phone number. 6. The system returns to step Step 7 in the Main Flow.

Toggle All Members

Name: Toggle All Members
Summary: The user may click the Toggle All selector checkbox to cause all selections in the Member Selector box to become checked or unchecked depending on the value of the selector.
Actors: End-user, System
Creation Date: 11/18/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://greek-db.com/service_event_creator.php
https://www.greek-db.com/judicial_case_creator.php
https://www.greek-db.com/announcer.php
Observed Value: The user may select all members or clear selections much faster.
Precondition: The user must have access to a page with the Member Selector box.

Main flow

Actors	System
1. The user clicks the Toggle All checkbox.	
	2. The system checks the Toggle All selector value. It is now selected.
	3. The system selects each of the Member Selector checkboxes.

Alternate flow 1: the toggle all selector is changed to unchecked

Actors	System
	2. The system checks the Toggle All selector value. It is now deselected.
	3. The system deselects each of the Member Selector checkboxes.

Update Judicial Case

Name: Update Judicial Case
Summary: From a populated judicial case page, the details of a case can be updated.
Actors: End-user, Database
Creation Date: 11/3/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/judicial_case_editor.php?idjudicialcases=####
Observed Value: Administrators can edit judicial cases for others (provided they have rights to do so).
Precondition: The user must be logged in. The user must also have permission to update the record.

Main flow

Actors	System
1. The user updates any information from the record they deem necessary.	
	2. The system checks that the user has permission to update the judicial case record. He/she is. 3. The system sanitizes the user input. 4. The system sends the data to the database for storage.
5. The database updates the record in the Judicial Cases table containing the user-submitted data.	
	6. The system closes the pop-up window.

Alternate flow: the user doesn't have permission to update cases

Actors	System
	2. The system checks that the user has permission to update case records. He/she is not. 3. The user is redirected to the No Access page on submission.

Update Member

Name: Update Member
Summary: An existing member profile is updated in the database.
Actors: End-user, Database
Creation Date: 10/16/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/memberinfo.php?localid=####
Observed Value: The user adds new information to an existing member profile
Precondition: The user attempting to add a new member must be an administrator to the membership tools. The user must also be logged in.

Main flow

Actors	System
1. The User fills in or changes some information about the member.	
2. The user clicks the Update button.	
	3. The system sends the new data to the database.
4. The Database update the data the member table for that member.	
	5. The system refreshes the user's pop-up window to reveal the updated information.

Alternate flow 1: member ID not entered

Actors	System
1. The user does not fill in a Member ID, and clicks Update.	
	2. The system notifies the user that the Member ID field is required and does not allow submission of the form.

Update Officer Permissions

Name: Update Officer Permissions
Summary: The system allows the user to update the officer's permissions, officer title, and member acting as the officer (if he has permission to do so).
Actors: End-user, Database
Creation Date: 11/4/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/officer_permissions_editor.php?officer=###
Observed Value: Officer permissions may be updated according to chapter policies and requirements. The title may change as policies are updated, and the member performing the duties of that officer may be updated as elections, promotions, and evictions occur.
Precondition: The user must be logged in. The user must have permission to access the page and to update officer permissions.

Main flow

Actors	System
1. The user updates the title of the officer, the acting member, and/or the officer permissions and clicks Update Officer.	
	2. The system checks that the user has permission to update member permissions. He/she does.
	3. The system sends the updated settings to the database for storage.
4. The system selects the officer being edited and updates the permissions values, updated title, and/or the updated holding member.	
	5. The system closes the Officer Permission Editor pop-up window.

Alternate flow 1: the user does not have permission to use the tool

Actors	System
	2. The system checks that the user has permission to update member permissions. He/she does not.
	3. The system redirects the user to the No Access page.

Update Service Event

Name: Update Service Event
Summary: From a populated service event page, the details of an event can be updated.
Actors: End-user, Database
Creation Date: 10/31/15 **Update Date:** 6/20/16
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/service_event_editor.php?idevent=####
Observed Value: Users can update community service events for themselves or administrators can update service event details for members.
Precondition: The user must be logged in. The user must also have permission to update the record.

Main flow

Actors	System
1. The user updates any information from the record they deem necessary and clicks the Update button.	
	2. The system checks that the user has permission to update the service event.
	3. The system sanitizes the user input, and sends the data to the database for storage.
4. The database updates the record in the Service Event table containing the user-submitted data.	
	5. The system closes the pop-up window.

Alternate flow: the user doesn't have permission to update events

Actors	System
	2. The system checks that the user has permission to update service events. He/she is not.
	3. The user is redirected to the No Access page on submission.

Update User Information

Name: Update User Information
Summary: A registered user updates his/her user account information.
Actors: End-user
Creation Date: 9/19/2015 **Update Date:** 9/19/2015
Version: 1.1 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/edit_account.php
Observed Value: User now has a new, updated email address or password.
Precondition: User must have an account, and must be logged in.

Main flow

Actors	System
1. User clicks Edit Account on the Session Bar and is redirected to the site.	
	2. System grabs the username and email from the Session data.
	3. System prints the edit account page with the email address blank pre-populated with the current email.
4. User enters an updated email address and/or password, and clicks Update Account.	
	5. System sanitizes user input to protect against SQL Injection.
	6. System checks for match between entered email and current email.
	7. System validates new email.
	8. System checks that new password is greater than zero-length.
	9. System generates a 1-time hash to use as a salt for the password.
	10. System generates a password hash using the user-supplied password and the newly-generated salt.
	11. System sends updated email, password, and salt to the database.
12. Database updates records in the user information table.	
	13. System redirects user back to landing page.

Alternate flow 1: email address not changed

Actors	System

4. User does not change the email address, and clicks Update Account.	
	5. System sanitizes user input. 6. System detects the email has not changed and continues to step 8 without validating the email.

Alternate flow 2: password left blank

Actors	System
4. User does not update the password, and clicks Update Account.	
	5. System sanitizes user input. 6. System validates email and skips to step 11 to update the email.

Alternate flow 3: email left blank

Actors	System
4. User submits blank email.	
	5. System prints an error stating that the email is invalid.

User Clicks Logo

Name: User Clicks Logo
Summary: A user clicks the logo at the center of the navigation bar.
Actors: End-user
Creation Date: 9/20/2015 **Update Date:** 9/20/2015
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/commonfiles/globalnavigationbar.php
https://s3.amazonaws.com/greekdb/resources/sessionbar/icons/greekDB-logo.png
Observed Value: User is redirected in an expected manner.
Precondition: None.

Main flow

Actors	System
1. User clicks the GreekDB Logo.	
	2. The system checks that the user is logged in. He/she is.
	3. The system redirects the user to the landing page.

Alternate flow 1: user is not logged in

Actors	System
	2. The system checks that the user is logged in. He/she is not.
	3. The system redirects the user to the GreekDB homepage.

User Login

Name: User Login
Summary: A user is attempting to log into the GreekDB system.
Actors: End-user
Creation Date: 9/19/2015 **Update Date:** 9/19/2015
Version: 1.3 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/login.php
https://www.greek-db.com/login/logincode.php
https://www.greek-db.com/commonfiles/common.php
Observed Value: User account is logs into the GreekDB system, allowing them to view their information and make changes/interact with the system.
Precondition: The user must have an account registered on GreekDB. The user must also be currently logged out.

Main flow

Actors	System
1. User enters a Username and Password, and clicks Login.	
	2. System sanitizes user input to protect against SQL Injection. 3. User account information and session information is requested from database.
4. The Database returns basic user information, including the password salt and the hashed password.	
	5. User-supplied password is hashed with the salt for the tested user. 6. The result matches the password hash for the user in the database. 7. The password matched, so the salt and password are removed from the session data. 8. The system requests the permission information for the officer roles held by the user, from the database.
9. The database returns the user's officer permission information.	
	10. The system processes all of these permissions and allocates the users maximum assigned permissions, or the defaults where no permissions were assigned.

	11. The user is redirected to the landing page.

Alternate flow 1: missing information

Actors	System
1. User enters too little information.	
	2. System prints a login failed message.

Alternate flow 2: incorrect password

Actors	System
	6. The hashed result does not match the hashed password for the user in the database. 7. Information imported into the session data is dropped. 8. System prints login failed message.

User Logout

Name: User Logout
Summary: A user logs out of GreekDB.
Actors: End-user
Creation Date: 9/19/2015 **Update Date:** 9/19/2015
Version: 1.0 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/logout.php
Observed Value: User is now logged out and must log in again to view their content or make changes.
Precondition: User must be logged in.

Main flow

Actors	System
1. User clicks Log Out link in the session bar.	
	2. System removes the user's current session information from memory. 3. System redirects the user to the login page.

Register User

Name: Register User
Summary: A user is registering in the system so they may begin using the system to manage their personal details.
Actors: End-user
Creation Date: 9/19/2015 **Update Date:** 9/22/2015
Version: 1.2 **Person Responsible:** Aaron Simmons
Path: https://www.greek-db.com/register.php
Observed Value: User account is now associated to the organization and may log in.
Precondition: User logging in must know the ID of the account they are registering to, and the organization values. He/she must also not be attempting to register to an account that has already been tied to another user. This information is managed/granted by an administrator for the joined organization.

Main flow

Actors	System
1. User opens a link to the registration page.	
	2. System queries the available organizations from the database.
3. Database returns the list of all available organizations and their chapters in {organization name} – {chapter designation} format.	
	4. System prints the registration page, with the available organizations and chapters in a selector for the user.
5. User enters username, email address, password, member id, and selects their organization and chapter, then clicks register.	
	6. System validates Username, password, and email were input by user. 7. System sanitizes user input to protect against SQL Injection. 8. System requests username, email, and member id from database.
9. Database returns no matching username, no matching email, and no user account registered to that member id.	
	10. System checks that the member id, email, and username have not already been registered to another user.

	11. System generates a 1-time hash to use as a salt for the password. 12. System generates a password hash using the user-supplied password and the newly-generated salt.
13. Database stores username, password, salt, email, organization id, chapter id, and member id.	
	14. System redirects user back to the login page.

Alternate flow 1: invalid information entered

Actors	System
9. Database returns a matching email address, matching username, or an existing registration to the member id.	
	10. System detects the issue with supplied information and prints an error explaining the issue to the user.

VIII. APPLICATION DATA MODEL

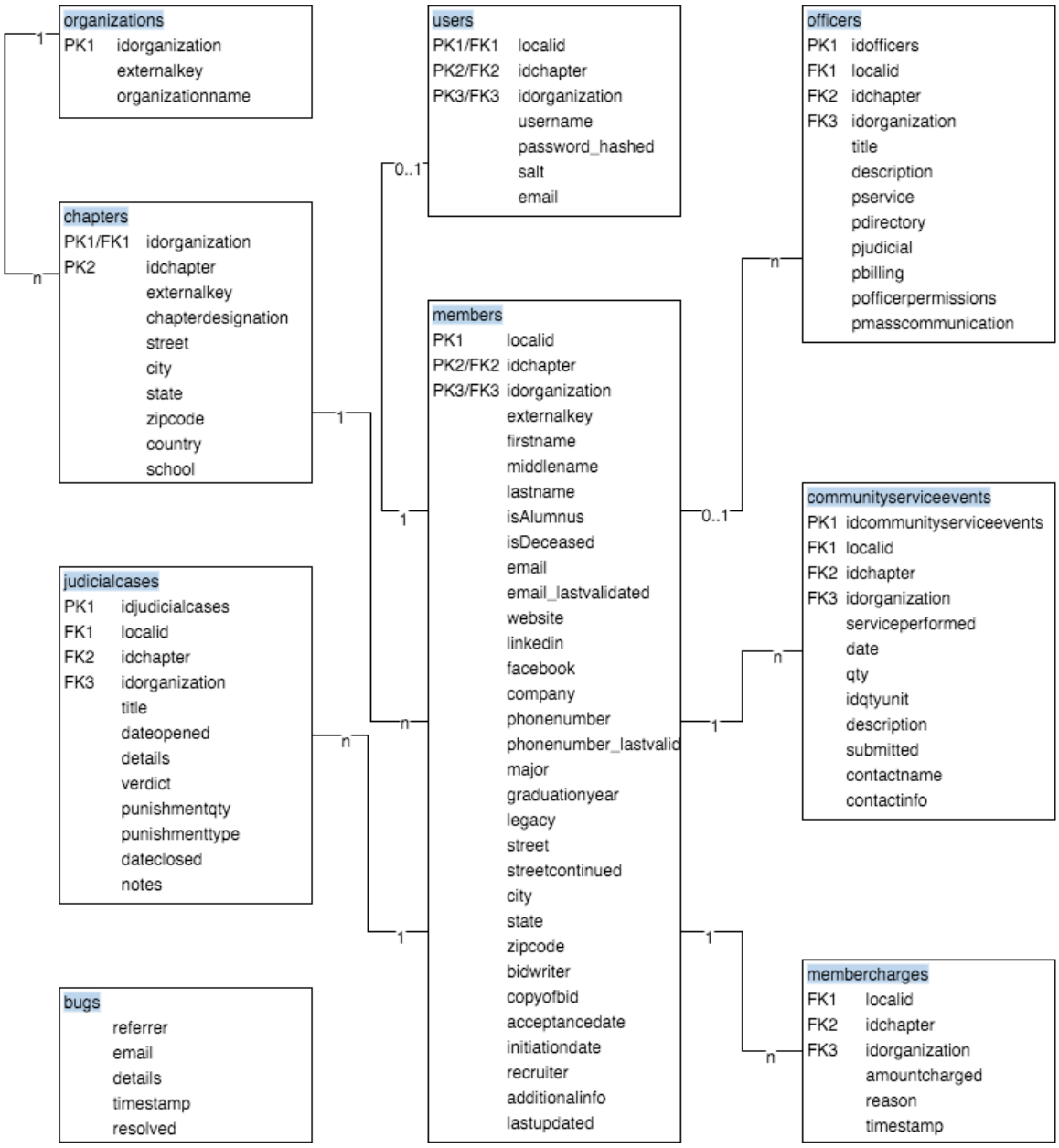


Figure 5. Data model.

About the Data Model Diagram

The diagram shows above describes the data structure of the database, including the relationship between the different tables in the database. The top row in the blue

highlight for each table is the name of the table. The first thing to notice is that all tables (with the exception of the bugs table) can be traced back to the member table. At its core, GreekDB is about tracking information about an organization's members, and their activities. In the current version of the software, much of the information being tracked can be traced back to an individual member performing an action, and the data structure reflects this.

Relationships

The lines drawn between tables are the relationship between the tables. On these lines are numbers or characters such as '1', 'n', and '0..1'. This indicates the type of relationship between the records in the table. A line showing a '1' on the left and a 'n' on the right indicates that for the table on the left, for every one record in that table, there may be any number of records in the table at-right. '0..1' means there may be no records, or one record in the table, but no more.

Keys

Primary keys in the tables are indicated by 'PK#' adjacent the column name. In some cases there are multiple primary keys, and in most cases these are a composite primary key, where a single key is made up of multiple columns. Foreign Keys are also indicated in the table, but by 'FK#'. A foreign key is a record whose value is intended to refer to a value in another table.

Members vs. Users

The next thing to note is the distinction between a member and a user. In GreekDB, a member is an individual who is a part of an organization. A member cannot necessarily interact with GreekDB because according to the system, they are just a record

in the ‘members’ table. This is distinct from a user, because a user is an account that can actually log in and interact with a system. This reduces the amount of information that was required in the Members table (which would already be rather large), and because the login operation is performed infrequently, it is expected that this separation would not provide a noticeable performance impact. Because of this relationship, where a user requires a member record, but a member does not require a user record, the relationship is ‘0..1’ to ‘1’.

Officers

Similar to Users, there is a ‘0..1’ to ‘n’ relationship between members and officers. The decision was made to tie officers to members rather than users to make management easier—An officer role could be attached to a member who didn’t have a user account made. This allows a full roster of officers to be created for record-keeping purposes, even if there are no special permissions an organization requires for them. It also means that there is no need to wait for a member to sign up for an account before assigning them an office. It should also be noted that the relationship is ‘0..1’ to ‘n’ because an officer can exist without being tied to a member, and one member can hold multiple offices (but not vice-versa).

Organizational Heirarchy

Finally, when looking at the members table, we see it has chapter and organization foreign keys. This models the real-life hierarchy for an organization, where a national organization is composed of individual chapters, for which a member is a part. This separation also allowed for adding custom settings on the organization and chapter

levels, including default settings and permissions, which could be applied from the top-down for these organizations as they saw necessary.

IX. APPLICATION FRONT-END

2-Panel Render

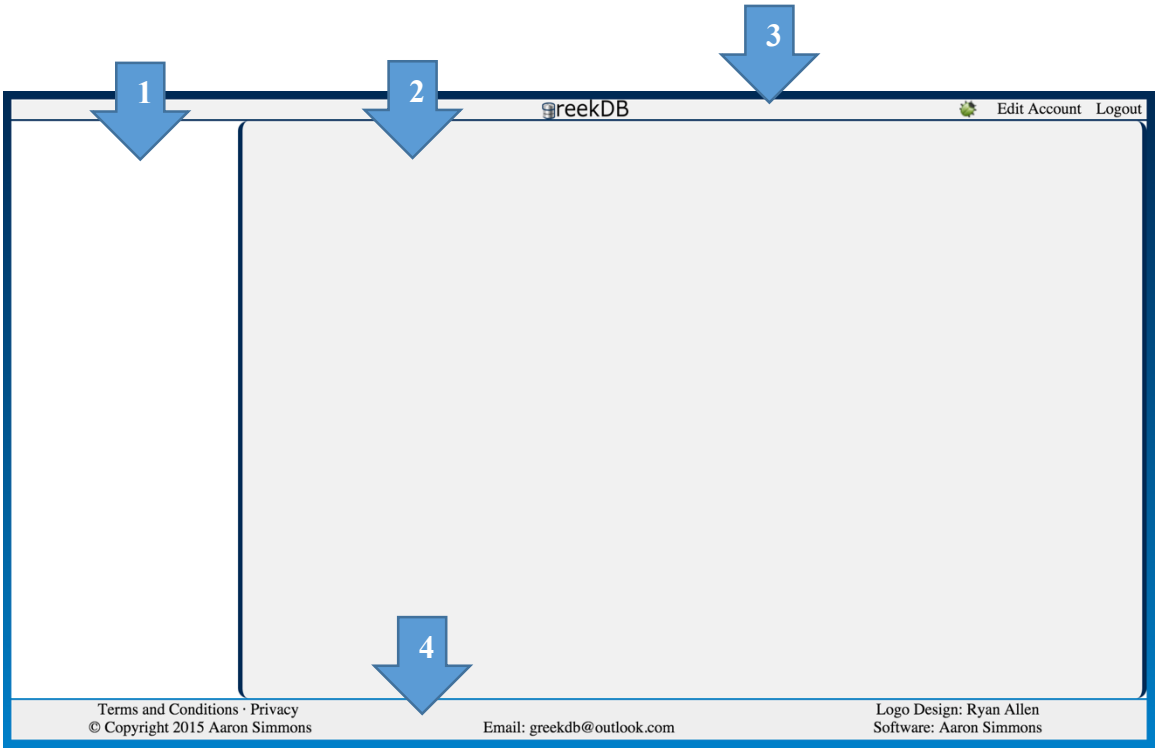


Figure 6. 2-panel render.

The 2-Panel Render is used when extra horizontal space may be necessary to display all contents on the center panel of the page without scrolling (assumed based on 1280 px horizontal resolution).

1. This white space is the location for the Navigation Banner. It is granted 20% of the horizontal space in the UI, with a minimum width of 300 px.

2. This top bar with the GreekDB logo is the Session Bar. The Session Bar resizes to the full width of the application window (minus the border widths).

3. This grey, boxed space is the application panel. The Application Panel is where the active page contents are displayed, such as the table of officer permissions. In the 2-Panel Render, the Application Panel is granted 80% of the horizontal space in the UI (minus border widths). It has a minimum width of 490 px.

4. This is the footer for the application, which is displayed on every page. The footer resizes to the full width of the application window (minus the border widths).

3-Panel Render

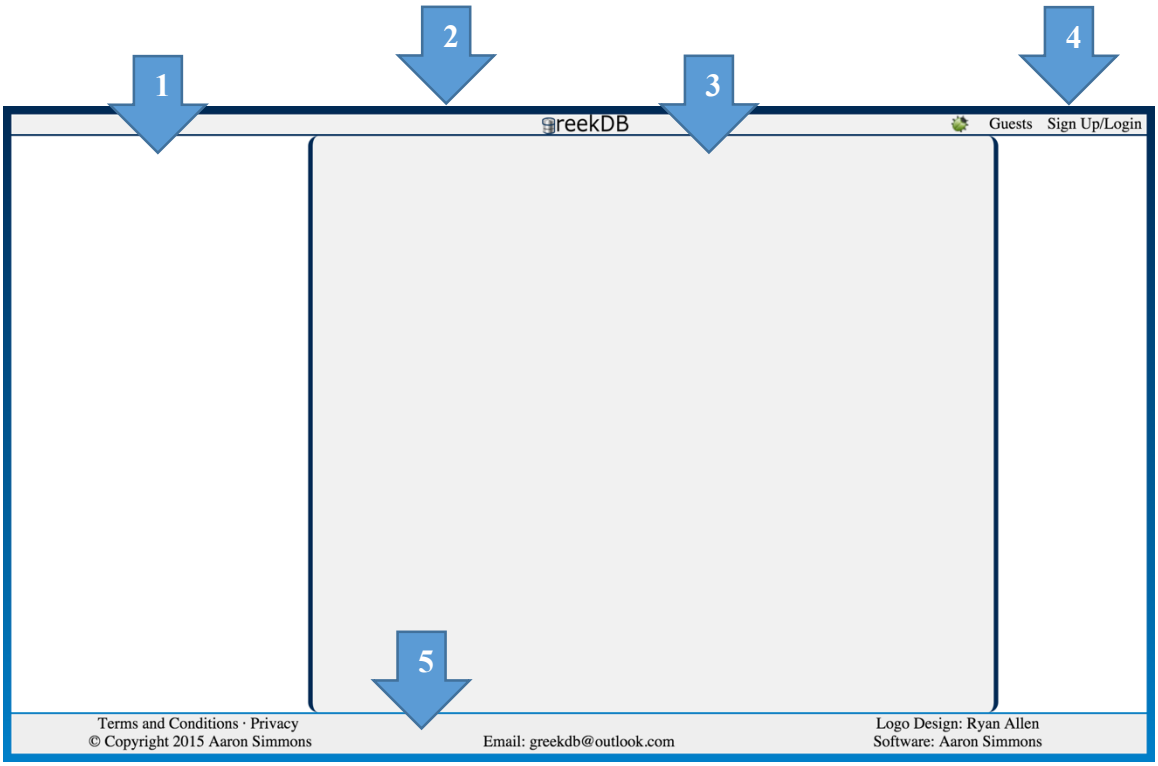


Figure 7. 3-panel render.

The 3-Panel Render is the standard rendering format.

1. This white space is the location for the Navigation Banner. It is given 20% of the horizontal space in the UI, with a minimum width of 300 px.

2. This top bar with the GreekDB logo is the Session Bar. The Session Bar resizes to the full width of the application window (minus the border widths).

3. This is grey, boxed space is the application panel. The application panel is where the active page contents are displayed, such as the table of officer permissions. In the 2-Panel Render, the Application Panel is granted 60% of the horizontal space in the UI (minus border widths). It has a minimum width of 490 px.

4. This right-hand white space is to fill the window, while moving the content of the page toward the center. It is granted 20% of the horizontal space in the UI, but has no minimum width. If the resolution is small enough, the page will be rendered without this right-hand panel (800 horizontal pixels).

5. This is the footer for the application. It is displayed on every full-page render. The footer resizes to the full width of the application window (minus the border widths).

Navigation Banner

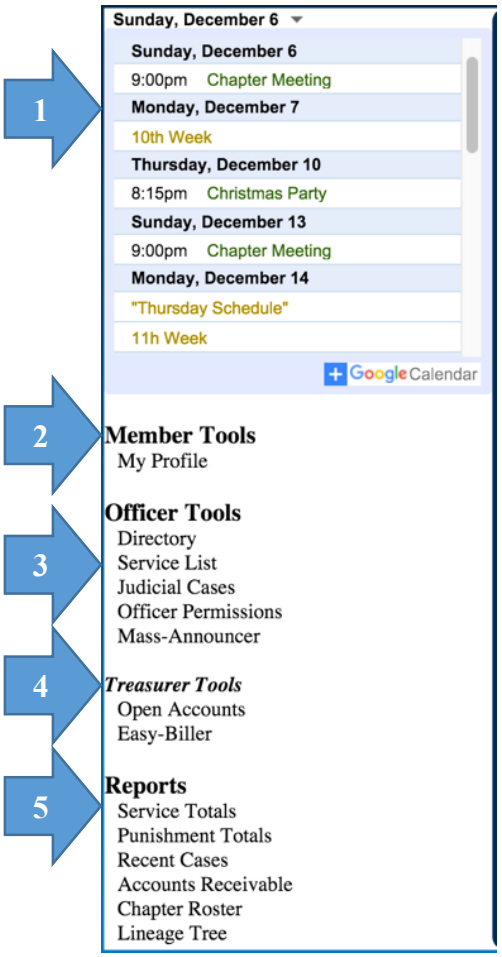


Figure 8. Navigation banner.

The Navigation Banner contains links to all of the tools for members and officers. It is rendered on every full-page tool in GreekDB.

1. This is a test of a Google Calendar® embed. The code is supplied by them, and isn't an official part of GreekDB.

2. This is the Member Tools section. This section contains all of the tools that the user has access to, but is not an Administrator to those tools.

3. This is the Officer Tools section. This section contains all of the tools that the user has access to, and is an Administrator for.

4. This is the special Treasurer Tools section. This only appears for users who have administrative rights to all members' financial information. There's a large number of tools that are specific to chapter financials, so a decision to add a separate section was made.

5. This is the Reports section. It contains all of the generateable reports, relevant to the offices the user holds.

Standard Footer

Figure 9. Standard footer.

This is the standard footer that is rendered on every full-page tool in GreekDB.

Session Bar

Not Logged in:

Figure 10. Session bar (not logged in).

Logged in:

Figure 11. Session bar (logged in).

The Session Bar indicates the user's logged-in status. It also provides links to edit user settings, a link back to the user's main page, and a link to allow the users to log out. It is rendered on every full-page tool in GreekDB.

1. The Logo serves as a link to the main page if the user is not logged in.

2. The bug icon is a link to the bug reporter. Clicking the icon opens the bug reporter as a new pop-up window.

3. The Guests link redirects the user to the Registration page.
4. The Sign Up/Login link redirects the user to the Login page.
5. The Logo serves as a link to the Landing page if the user is logged in.
6. The Edit Account link redirects the user to the Edit Account page, where the user can update their email address and password.
7. The Logout link allows the user to log out of GreekDB.

Member-Selector

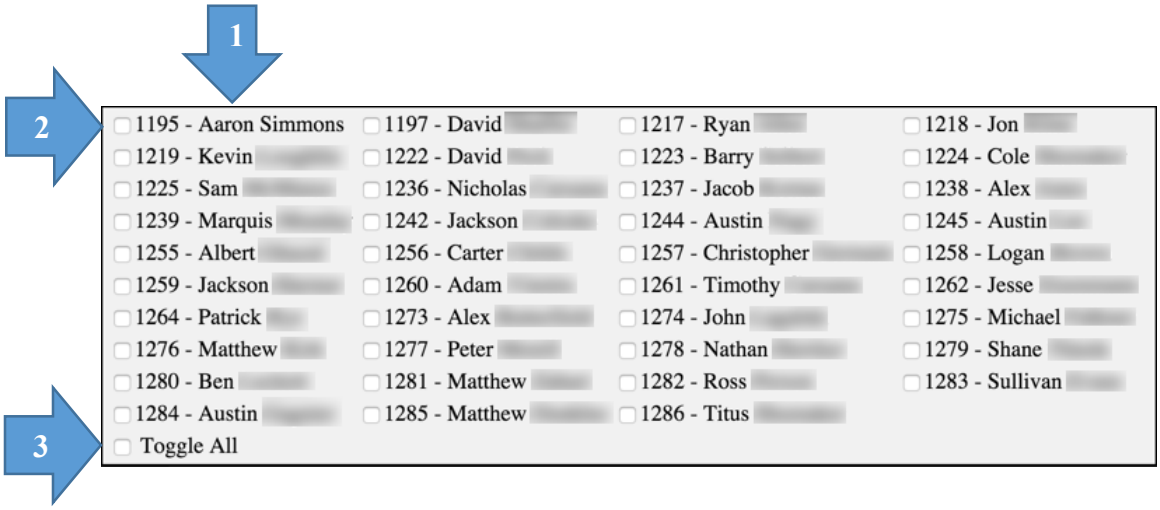


Figure 12. Member selector.

This is the Member-Selector box, which is printed on various tools to make bulk operations easier. The Member-Selector is independently scrollable to accommodate issues with available screen/window size.

1. A record is printed for each active member. Each record consists of a selector box, the member's ID, and his/her first and last names.
2. The tool can configure the number of member records to print in each row, and will print rows until it runs out of records to populate it with.
3. At the end of the list is printed a Toggle All button. When clicked, it selects or de-selects all checkboxes in the Member-Selector to match the value of the Toggle All button.

Login Page

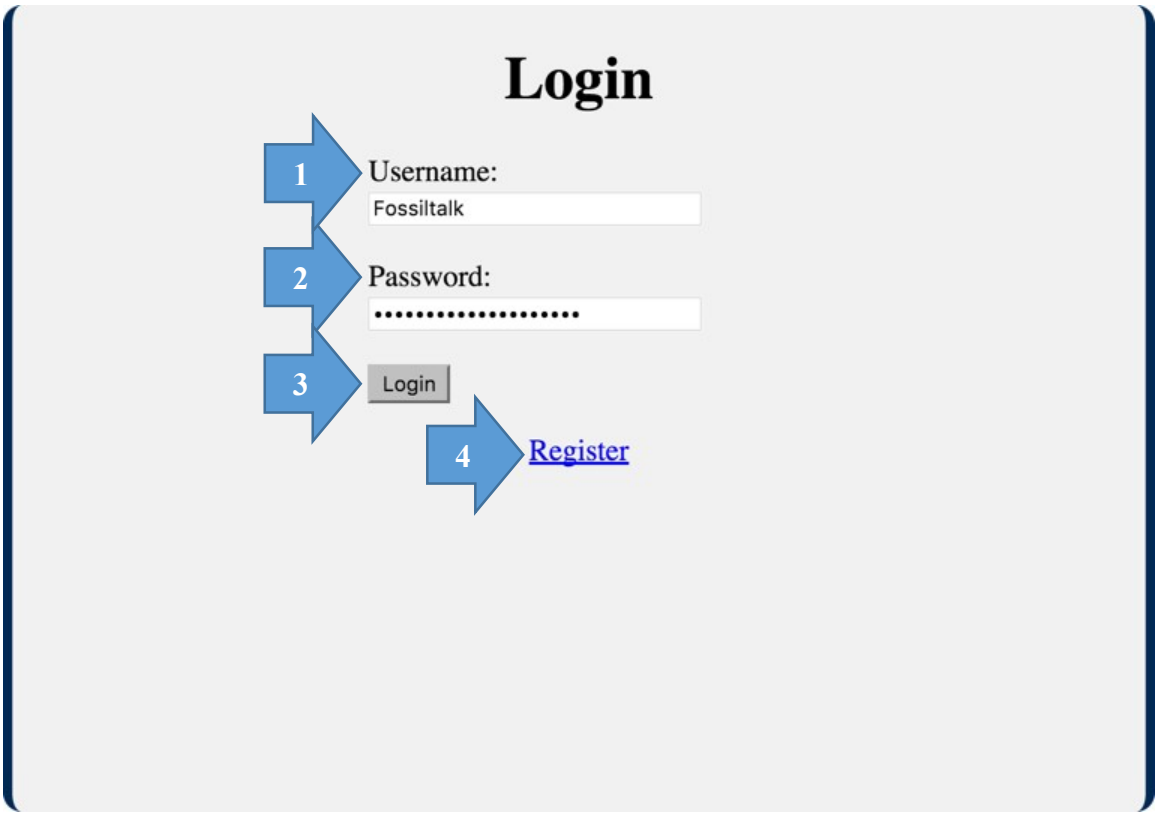


Figure 13. User login page.

The Login is presented to the user when he/she clicks the Sign Up/Login link in the session bar, or attempts to access the landing page when not logged in. The Login Page is rendered using the standard 3-Panel render.

1. The system requires the user to present a username for login. Text is rendered in the user's system-default font.
2. The system requires a password for login. It will not proceed if the password blank is left unpopulated. The Password blank hides the password characters that the user enters for additional security.
3. The user must click the login button with the Username and Password populated. The Login button is printed in the user's system/browser-default button style.
4. The Register link takes the user to the user registration page.

Landing Page

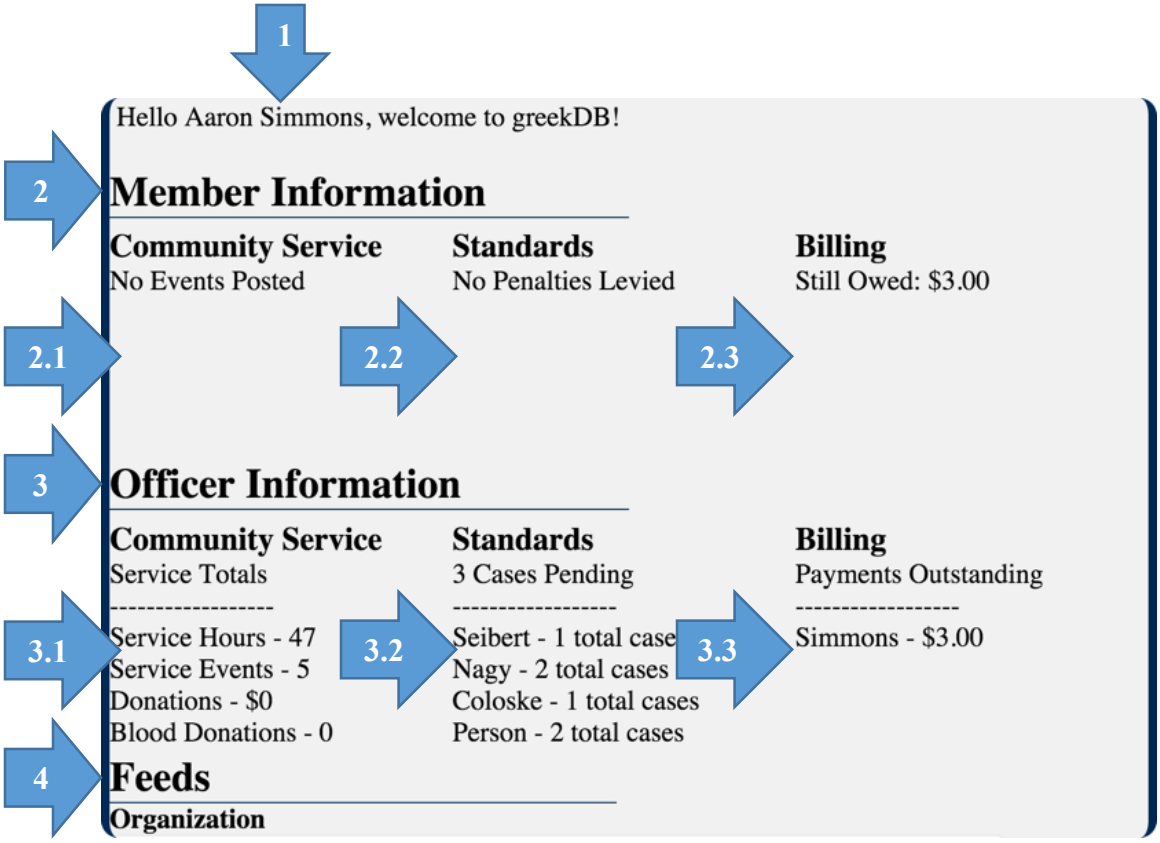


Figure 14. Landing page.

The Landing Page is intended to contain most of the information that a user would need at a glance, either for their day-to-day questions, or as an officer performing their duties. The Landing Page is displayed using the standard 3-Panel Render. Because of potential issues with available screen size, this panel has been made independently-scrollable. The scrollbar is set up to only appear when all of the content would not fit within the available area provided for it on-screen.

1. The system generates a welcome message for the user, indicating which user is logged in.

2. The system generates a Member Information header. The header indicates to which section the following information belongs. If this header is not displayed, it means that the user does not have permission to access any of the information needed to generate the data in these applets.

2.1. This section contains the user's current Community Service contribution. The data listed here is generated from the last 6 months. If there are no records found within this timeframe, it will display "No Events Posted" to make it clear that no events have been logged and the user may need to consult the chapter's service chairman if he/she believes this information is incorrect.
* If the user does not have permission to access the information needed by this applet, neither the header, nor the applet data will be displayed.

2.2. This section contains the user's current Standards/Judicial Case results for the last 6 months. If no records are found, it will display "No Penalties Levied" to make it clear that no cases have been logged and the user may need to consult that chapter's judiciary chairman if he/she believes this information is incorrect.
* If the user does not have permission to access the information needed by this applet, neither the header, nor the applet data will be displayed.

2.3. This section contains the user's current financial information. If the member owes no money, it will state "Balance: $0.00", if the member is owed money by the chapter, it will state that the user has a balance and the amount (eg. Balance: $3.21). If the user owes money to the chapter, it will state "Still Owed:", followed by the amount owed by the member to the chapter.
* If the user does not have permission to access the information needed by this applet, neither the header, nor the applet data will be displayed.

3. This system generates an Officer Information header. The header indicates to which section the following information belongs. If this header is not displayed, it means that the user does not have permission to access any of the information needed to generate the data in these applets.

3.1. This section contains the chapter's Community Service contribution. The data listed here is generated from the last 6 months, and is broken down into the types of service/donations performed/given. If there is no data logged for the chapter in the last 6 months, the applet will indicate this by populating each category with the respective sum (which should be 0).
* If the user does not have permission to access the information needed by this applet for all users, neither the header, nor the applet data will be displayed.

3.2. This section contains the chapter's current judicial case status, and any members who might be causing problems. The data is broken into two parts:

- Cases Pending: This is the number of cases that have not been closed by a member of the judicial committee. This serves as a reminder that there are still outstanding cases which need to continue through the judicial process.

- Repeat Offenders: This is a list of members who have been part of judicial proceedings in the last 6 months. The list first pulls the members who have been involved in judicial proceedings in the last 6 months, and displays a

random set of 4 from that list, and the total number of cases that member has been involved in.

* If the user does not have permission to access the information needed by this applet for all users, neither the header, nor the applet data will be displayed.

3.3. This section contains the chapter's members' current financial status. It pulls a list of all members who have outstanding balances owed to the chapter, and displays a random set of 4 members who still owe money to the chapter and their outstanding balance.
* If the user does not have permission to access the information needed by this applet for all users, neither the header, nor the applet data will be displayed.

4. The system generates a Feeds Header. This contains the Twitter ® feeds for the national organization and for the chapter. This code was provider by Twitter's APIs and is not an official part of the GreekDB System.

Chapter Directory

Figure 15. Chapter directory.

The Chapter Directory contains a brief overview of each member's contact information. It's displayed as a standard table format to make copying data off of the webpage simple. Not shown is the link to add a new member which appears at the bottom of the page if the user has permissions to add members. The Chapter Directory is displayed using the standard 3-Panel Render. Because of potential issues with available screen size, this panel has been made independently-scrollable. The scrollbar is set up to only appear when all of the content would not fit within the available area provided for it on-screen.

1. The system prints the Chapter Directory header.
2. This column contains the ID # for the member. Some organizations have their own system of unique IDs that thaey choose to use, and this is supported by GreekDB and displayed here (such as Bond Number).
3. This column contains the First Name of the member.
4. This column contains the Last Name of the member.
5. This column contains the Email Address of the member.
6. This column contains the Phone Number of the member in a user-friendly display format.
7. This column contains the Graduation Year of the member, which can help in identifying members when the ID is not known and more than one member has the same name.
8. Each member record has its own row in the table. They are displayed in the Chapter Directory in descending order by Member ID (Bond #). There are 3 color-codes associated with the records as they are displayed:
    - 8.1. Blue text indicates that the member is currently an active member and will be displayed in lists and tables that are filtered to only display active members.
    - 8.2. Black text (system default color) indicates that the member is no longer active in the organization.
    - 8.3. Gray text indicates that the member is deceased, and should likely not be sought for contact, nor is the members information likely to be updated.

Chapter Service List

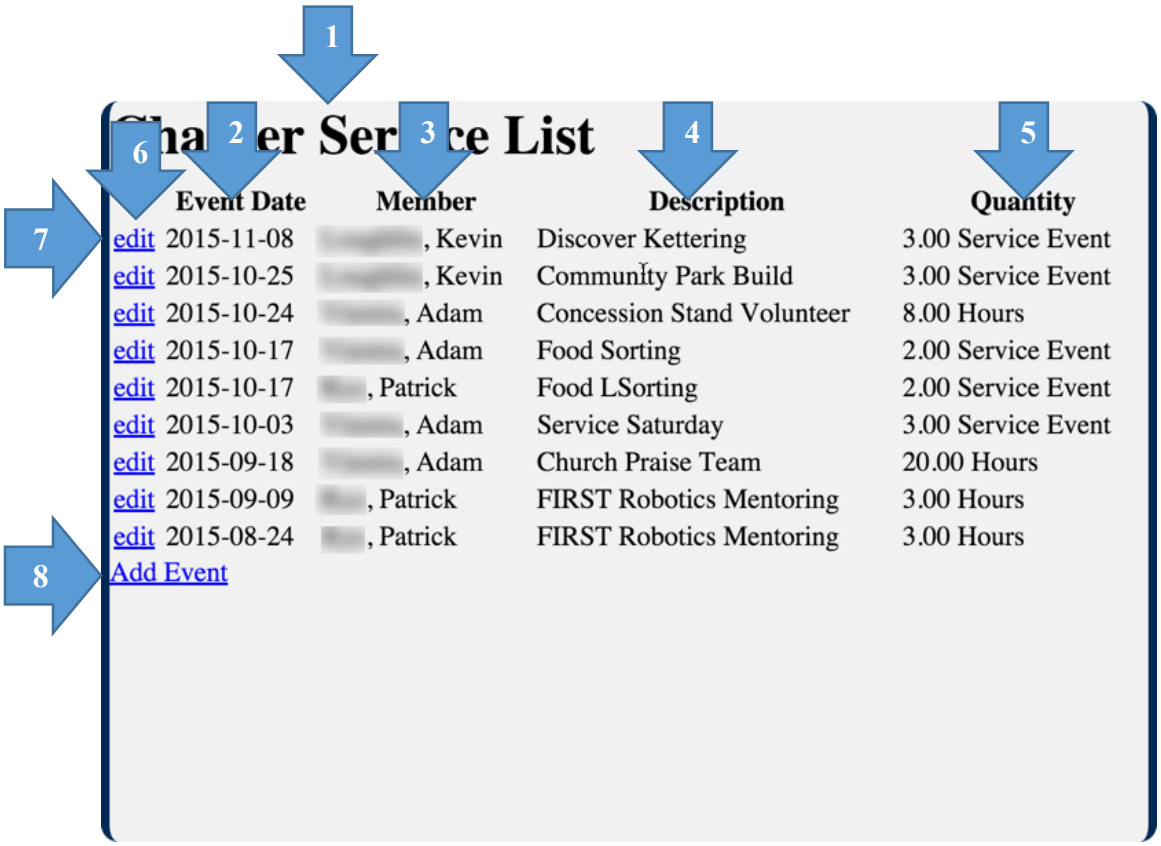


Figure 16. Chapter service list.

The Chapter Service List contains a list of all community service events logged in the last 6 months. The list is printed in a standard table format for easy copy-pasting into other applications. The Chapter Service List is displayed using the standard 3-Panel Render. Because of potential issues with available screen size, this panel has been made independently-scrollable. The scrollbar is set up to only appear when all of the content would not fit within the available area provided for it on-screen.

1. The title of the utility is printed at the top of the page.
2. This column contains the date of the logged event.

3. This column contains the first and last name of the member who performed the service (in [Last Name], [First Name] format).

4. This column includes the title of the service event, which should describe in a couple of words what the service was performed, or for whom.

5. This column includes the quantity of service/donation that was performed/given and how it is to be logged.

6. This link appears next to each event that the user has access to view details about. If the user has read-only rights, it will say, “view”. If the user has rights to edit the event, it will say, “edit” instead (as shown in the screenshot). Members can have mixed rights on different events, so it is common to see a combination of edit, view, and no links.

7. Each record is printed in its own row. The records are ordered initially by date in descending order, and then alphabetically by last name where conflicts occur.

8. At the bottom of the table is printed a link to add new service events, provided the user has rights to add events.

Judicial Case List

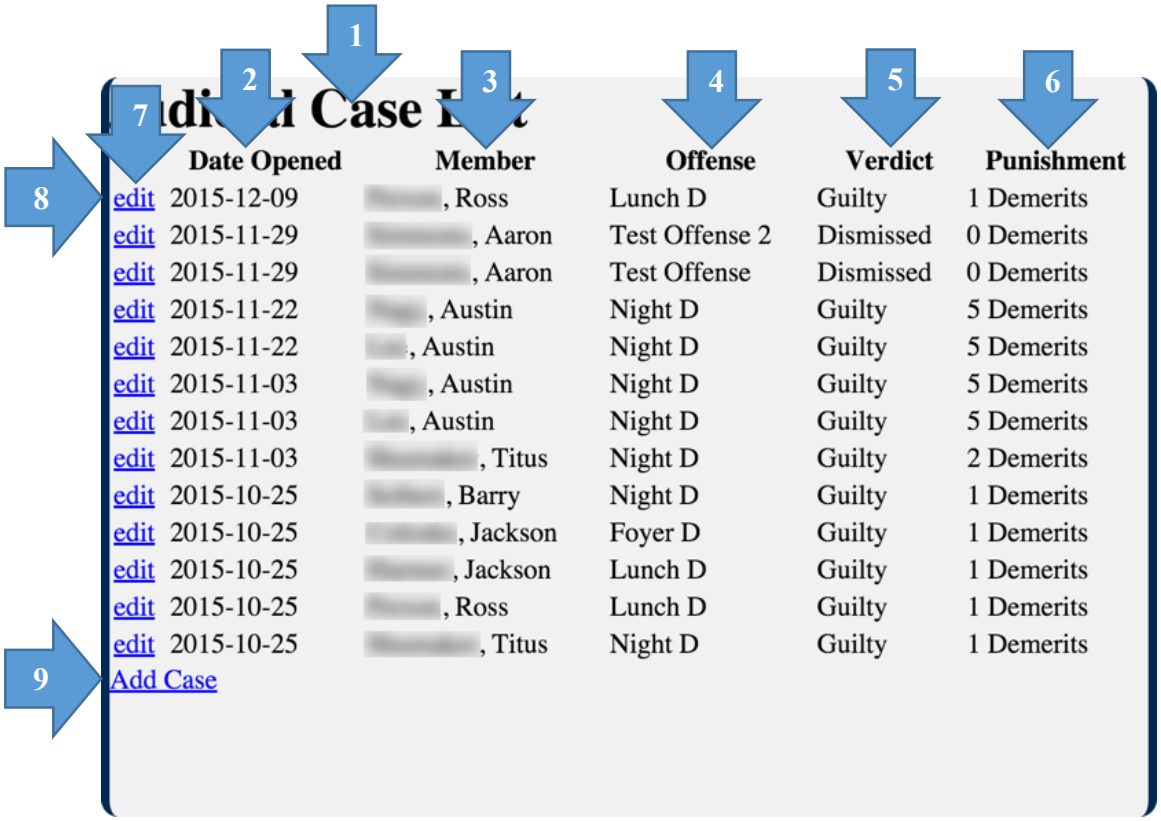


Figure 17. Judicial case list.

The Judicial Case list contains all of the judicial cases that have been logged for the last 6 months. The list is printed in a standard table format for easy copy-pasting into other applications. The Judicial Case List is displayed using the standard 3-Panel Render. Because of potential issues with available screen size, this panel has been made independently-scrollable. The scrollbar is set up to only appear when all of the content would not fit within the available area provided for it on-screen.

1. The title of the utility is printed at the top of the page.
2. This column contains the date that the judicial case was opened.
3. This column contains the first and last name of the member who performed the service (in [Last Name], [First Name] format).
4. This column contains the title of the offense from the case record.

5. This column contains the case result.

6. This column contains any punishments that were levied. When a case is dismissed or a member is found innocent, this should read 0. For guilty or pending, this should read the resulting punishment or the recommended punishment, respectively.

7. This link appears next to each case that the user has access to view details about. If the user has read-only rights, it will say, "view". If the user has rights to edit the event, it will say, "edit" instead (as shown in the screenshot). Members can have mixed rights on different cases, so it is common to see a combination of edit, view, and no links.

8. Each record is printed in its own row. The records are ordered initially by date in descending order, and then alphabetically by last name where conflicts occur.

9. At the bottom of the table is printed a link to add new judicial cases, provided the user has rights to add cases.

Officer Permissions Table

Officer	Community Service	Chapter Directory	Judical Cases	Billing	Permissions Management
A - Alumni Secretary		Edit All			View All
A - Community Service Chairman	Edit All				
A - President	Edit All	Edit All	Edit All	Edit All	Edit All
A - Treasurer		Edit All		Edit All	
A - Vice President		Edit All			View Self/Edit Others
B - Academics Chairman		Edit Self/View Others			
B - Alumni Secretary		Edit All			
B - Athletics Chairman			View Self/Edit Others		
B - Community Service Chairman	Edit All				
B - President	Edit All	Edit All	Edit All	Edit All	Edit All
B - Standards Rep			View All		
B - Standards Rep			View All		
B - Standards Rep			View All		
B - Standards Rep			View All		
B - Treasurer		Edit All		Edit All	
B - Treasurer in Training		Edit All	View All	Edit All	
B - Vice President		Edit All			View Self/Edit Others
B - Warden			Edit All		
B - Works Manager			Edit All		

Figure 18. Officer permissions table.

The Officer Permissions table contains all officers that have been created for a chapter, and their respective permissions for the different kinds of utilities. The table is

printed in a standard table format for copying and pasting into other applications. The Officer Permissions table is displayed using the standard 2-Panel Render. Because of potential issues with available screen size, this panel has been made independently-scrollable. The scrollbar is set up to only appear when all of the content would not fit within the available area provided for it on-screen.

1. The Officer Permissions title is printed at the top of the page.
2. This column contains the officer titles.
3. This column contains the permissions each officer has with regard to community service records.
4. This column contains the permissions each officer has with regard to member information (Chapter Directory).
5. This column contains the permissions each officer has with regard to judicial case records.
6. This column contains the permissions each officer has with regard to member financial/transaction records (Billing).
7. This column contains the permissions each officer has with regard to managing officer permissions.
8. Each officer has a row in the table, indicating what permissions each officer has for each set of tools. The title is a link to reconfigure the officer's permissions.

Mass-Announcer

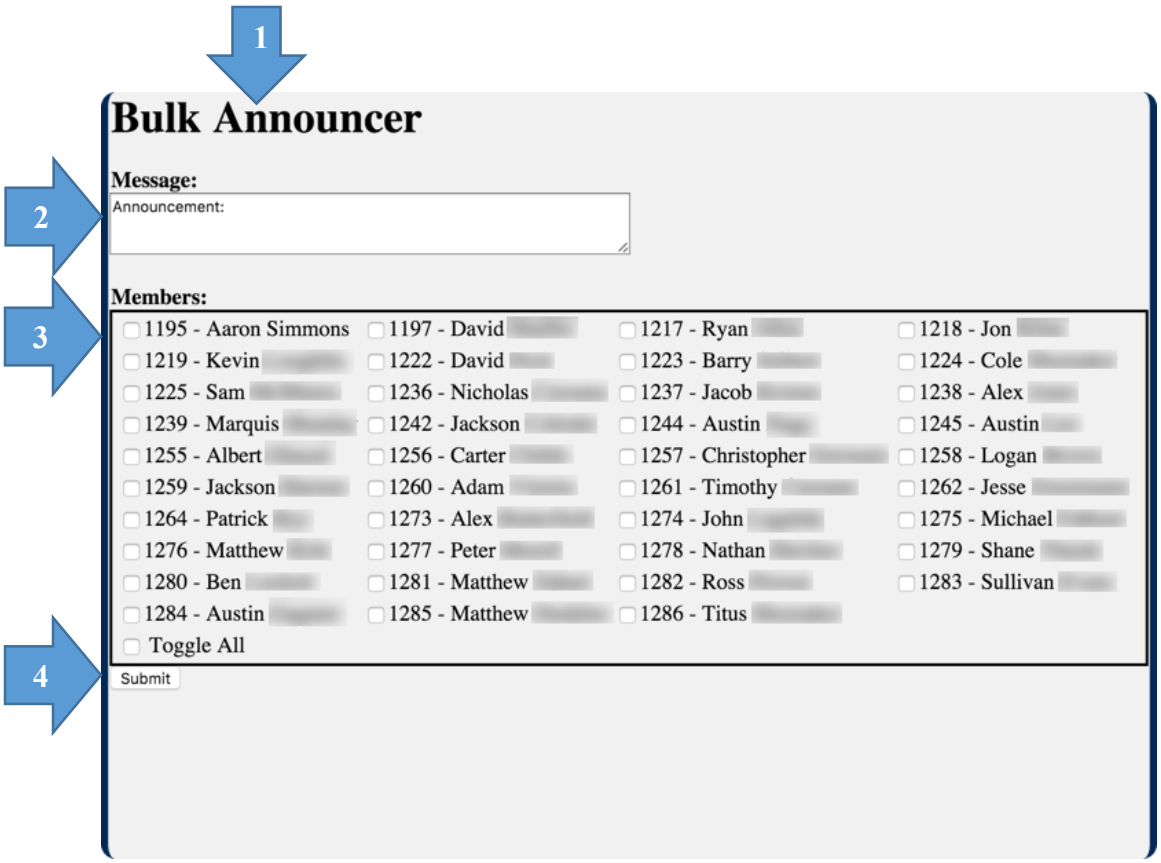


Figure 19. Bulk announcer utility.

The Mass-Announcer allows officers to send SMS (text message) announcements to any subset of the active members of the chapter. The Mass-Announcer is displayed using the standard 3-Panel Render. Because of potential issues with available screen size, this panel has been made independently-scrollable. The scrollbar is set up to only appear when all of the content would not fit within the available area provided for it on-screen.

1. The Mass-Announcer (Bulk Announcer) title is printed at the top of the page.
2. This blank is where the user types the message he/she would like to have sent to the members of the chapter. It is pre-populated with the "Announcement:" text, but the text may be removed to reduce the character count.

3. Printed here is the member-selector box, printed in a 4-column format. See Figure 10 for more information (p. 138).

4. Printed at the bottom-left is the submit button for the announcer.

Edit Account

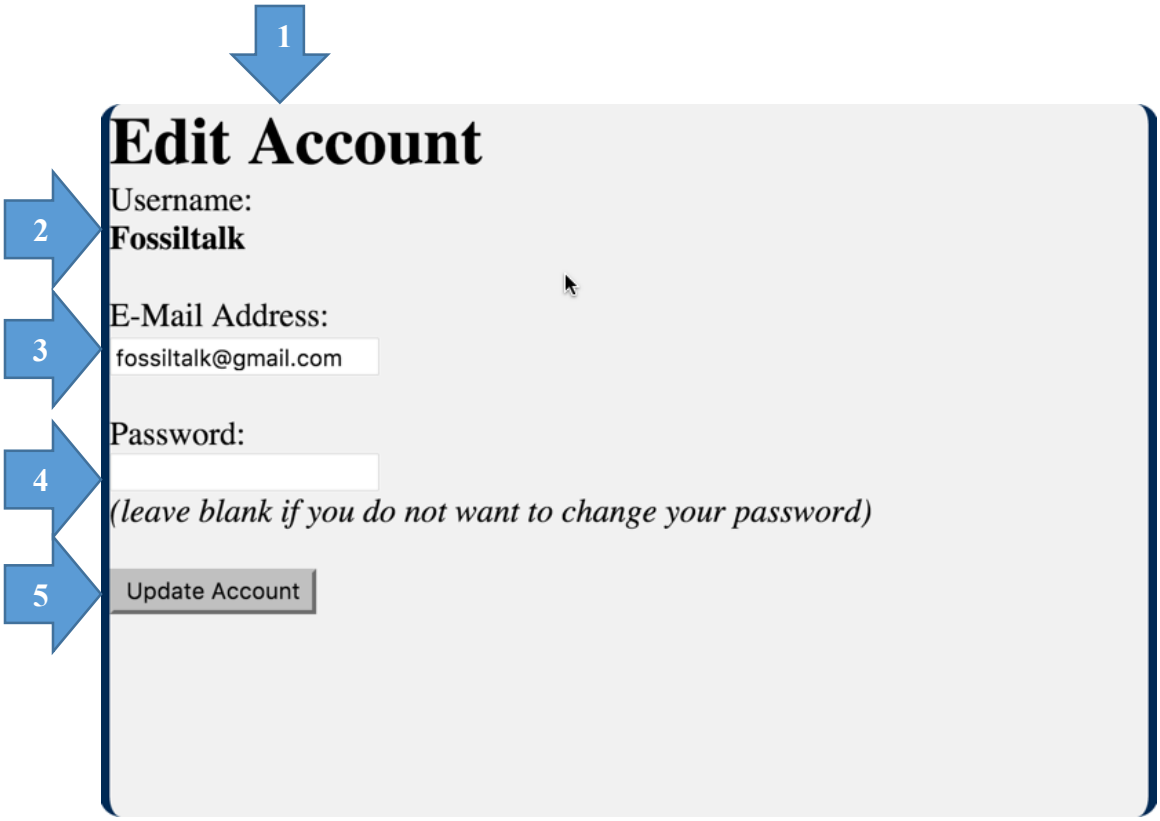


Figure 20. Edit account page.

The Edit Account utility allows a user to change his/her email address and to reset his/her password. The page is printed using the standard 3-Panel render, and the panel is independently-scrollable to accommodate issues with screen resolution.

1. The Edit Account title is printed at the top of the page.

2. The user's username is printed at the top of the page to identify who the logged-in user is.

3. An e-mail address blank is printed with the user's current email address pre-populated for easy comparison.

4. A blank password blank is printed. If text is entered, the text is hidden for security.

5. An update account button is printed in the default style of the user's browser/operating system.

Open Accounts

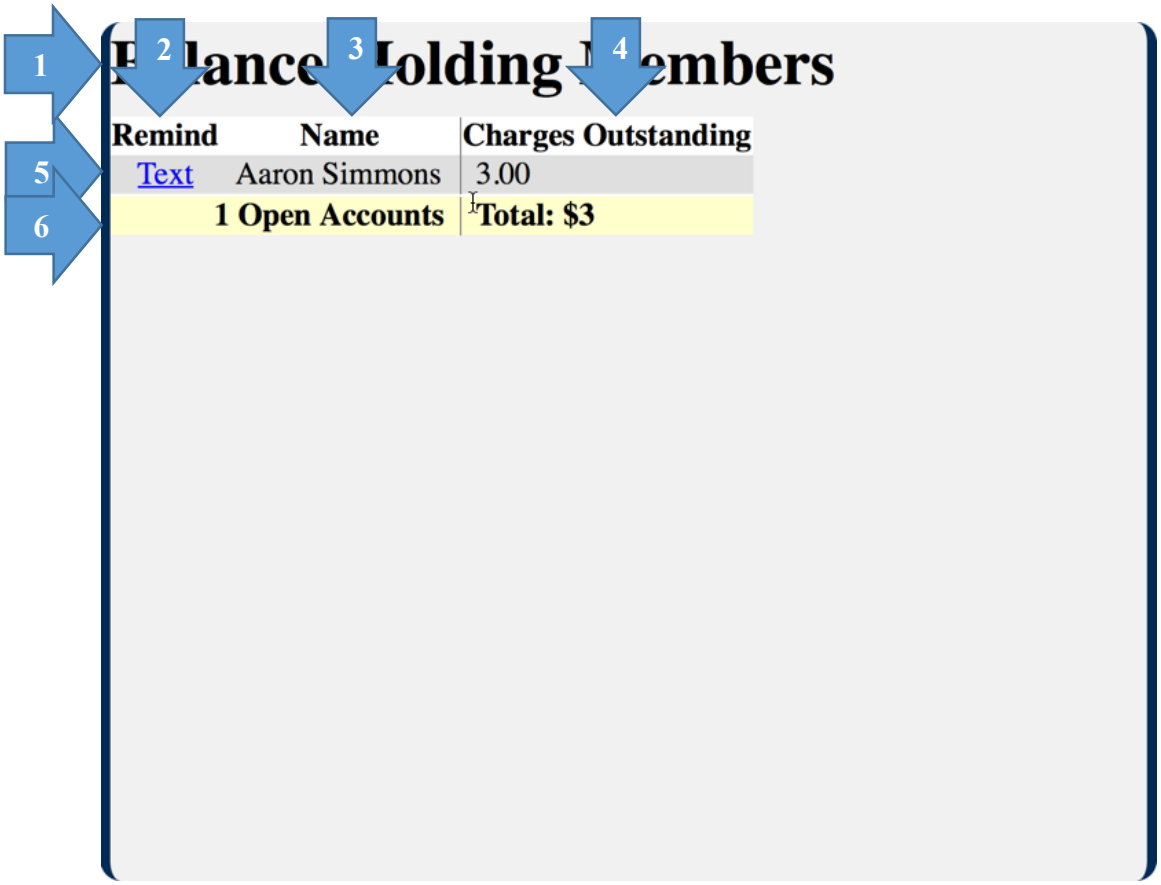


Figure 21. Open accounts page.

The Open Accounts utility allows an officer to quickly view all members who currently have a balance owed to the organization, see the total receivables from members, the total members with balances outstanding, and to notify members via SMS that they have a balance owed to the organization. The page is displayed in the standard 2-panel format and is independently scrollable to accommodate low-resolution screens or a large number of displayed records.

1. The Open Accounts (Balance-Holding Members) title is printed at the top of the page.
2. This column contains the links to send reminder messages to members about their balance.
3. This column contains the name of the member owing a balance.
4. This column contains the current amount that is owed by the member.
5. Each returned balance-owing member is shown in his/her own record with a notification link, the member's First and Last Name, and his/her balance owed to the organization.
6. The final row in the table contains the total number of open accounts, as well as the total sum of all the balances owed to the organization by its members.

Bug Reporter

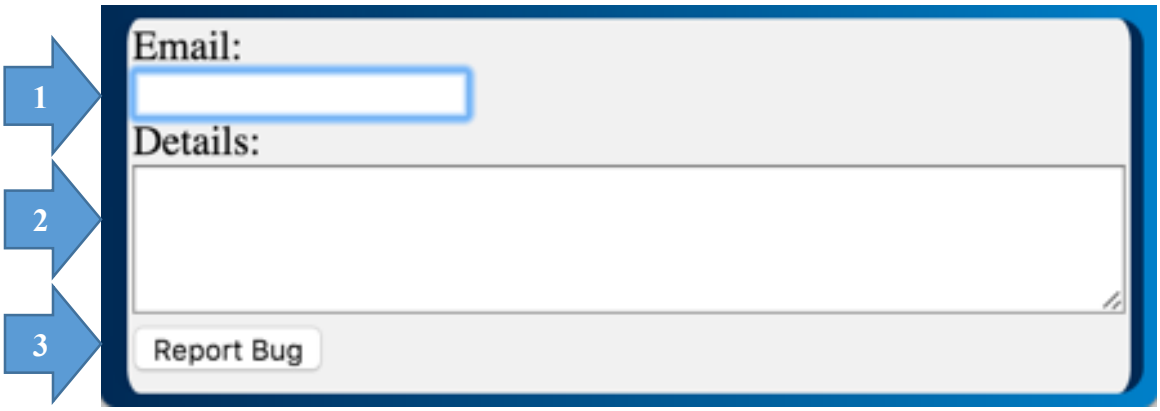


Figure 22. Bug reporting tool.

The bug reporter allows the user to submit bug reports from any page in GreekDB. The page is displayed in a new pop-up window whenever the user clicks the green bug icon in the session bar. When the user clicks the Report Bug button, the pop-up window is closed.

1. This text box is where the user puts his/her email address for return-correspondance. This field is required in case further inquiry into the problem is required.

2. This text field is where the user puts details about the problem he/she is experiencing. This field is also required for submission.

3. This button submits the bug report, and is printed in the default button style of the user's browser/operating system.

Member Information Page

Figure 23. Member information page (member profiles).

The Member Information Page contains all of the logged information about a member of the organization. The information is broken up into sections based on the type of information. The Member Information Page is displayed in a pop-up window, with the member id displayed in the URL to allow bookmarking and sharing links to the

member profile with others (who have permission to view it). Changes can be made to blanks in multiple tabs without the need to click Update Info each time. The window is independently-scrollable in case it is used on a low-resolution device.

1. The Member's First and Last Name are printed at the top of the page.
2. The Member's ID is printed next, above the section tabs.
3. Tabs exist currently for:
    a. General Info, which includes contact information and some other basic information.
    b. Historical Information, which includes information like sign-up dates, and the member who recruiter him/her.
    c. Emergency Contact Information, which includes information about allergies, emergency contacts, and health insurance information
    d. Additional Information, which allows members and officers to log additional information about the member.
4. Information is printed from top to bottom, in a single column in each tab, except where it makes sense (see Address section).
5. Certain records contain last-modified dates, which are updated when the field they are tied to is updated to contain new information (a change in case does not change the update date).
6. This button submits the changes to the member profile.
7. This link allows the member to download the member's contact card (vCard format).

Service Event Creator

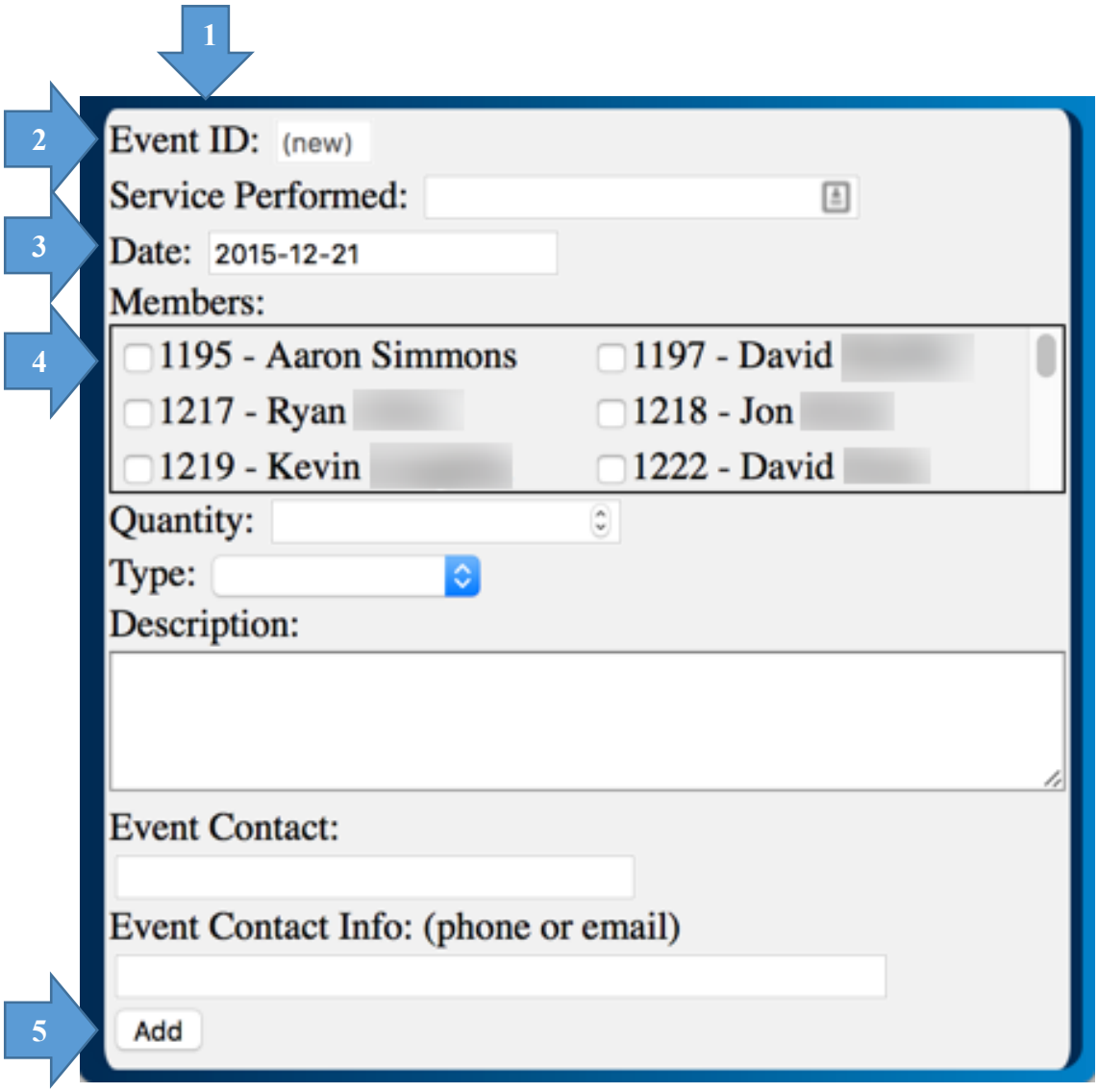


Figure 24. Service event creation tool.

The Service Event Creator allows members and officers to log community service events for 1 or more members. The Service Event Creator is opened in a pop-up window when the user clicks the link to add a new case. The window is independently-scrollable in case it is used on a low-resolution device.

1. Each item is printed top-to-bottom in a single column format.

2. The Event ID is printed with a place-holder value of "(new)" to indicated that is a new event, and the blank is not editable by the user. The ID is generated by the database on submission.

3. The Date blank is pre-populated with today's date for convenience. It uses the html 5 "Date Field", so in some browsers, when clicked, a calendar selector will appear to help the member in selecting an event date.

4. The Member-Selector box is printed in a 2-column format. See Figure 10 for more information (p. 138).

5. An Add button is printed at the bottom of the window to submit the event to GreekDB. It is printed in the default style of the user's Operating System or Browser.

Service Event Editor

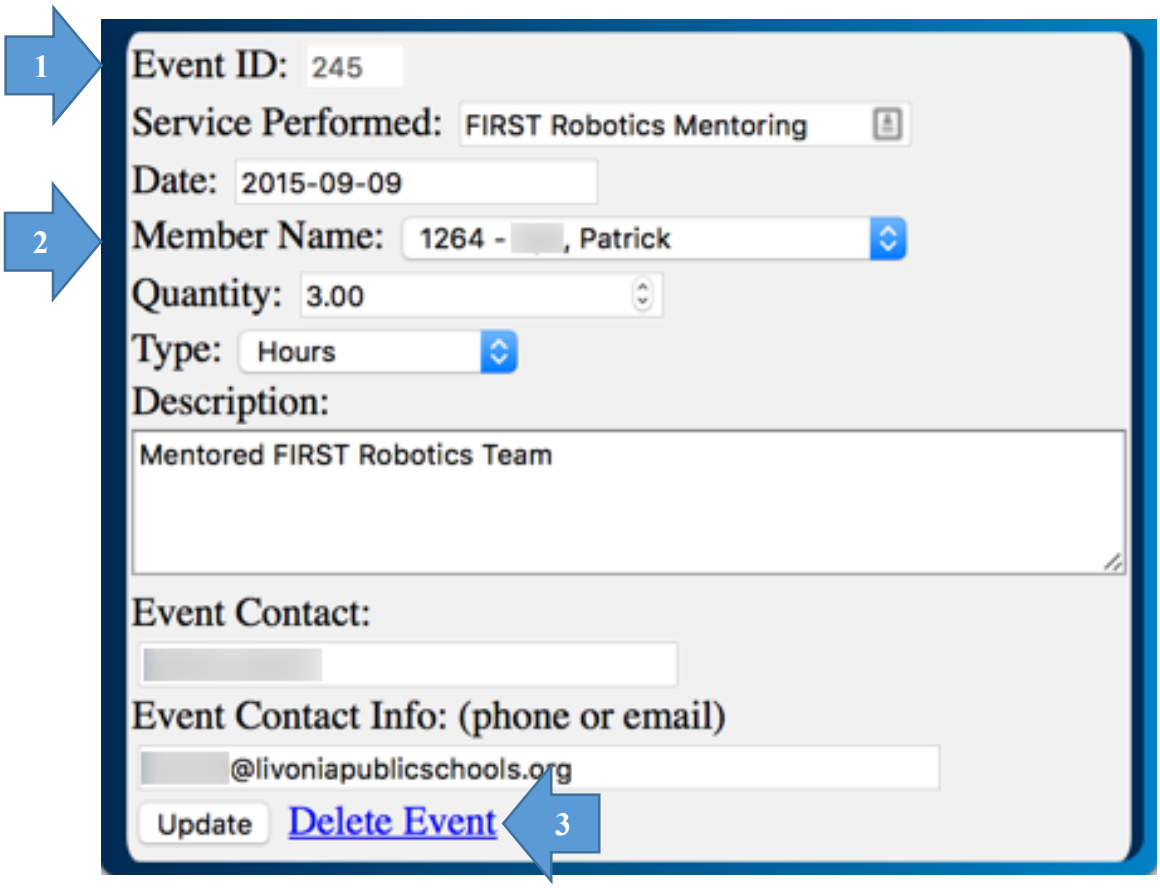


Figure 25. Service event editing tool.

The Service Event Editor allows a member or administrator to update a community service event that has already been logged. Because the Creator tool stores records as single-member service records, the editor is only used to modify one record at a time. The Service Event Editor is opened in a pop-up window when the user clicks the link to add a new service event. The window is independently-scrollable in case it is used on a low-resolution device.

1. The Event ID is the unique identifier for a community services event record. It is displayed on the in the Event Editor window in a non-editable text box, and can be used to look up the event later.
2. Because each event is assigned a single member, the member from the event is shown selected in a drop-down. The value can be changed.
3. The Delete Event link allows the user to delete the event record.

Judicial Case Creator

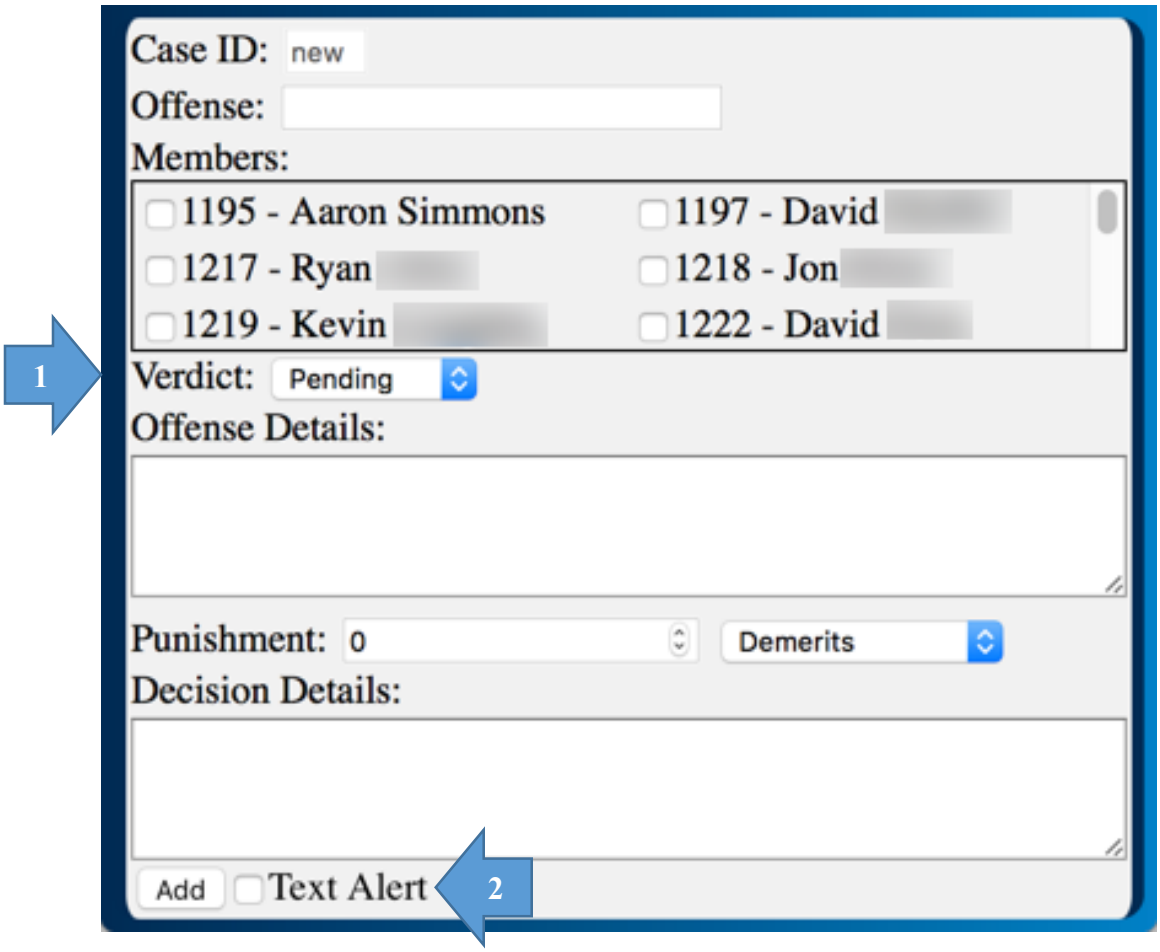


Figure 26. Judicial case creation tool.

The Judicial Case Creator is similar to the Service Event Creator (p. 155), containing a member selector box, and creating multiple records at a time, but used for bulk-creation of judicial cases instead of community service events. The Judicial Case Creator is opened in a pop-up window when the user clicks the link to add a new case. The window is independently-scrollable in case it is used on a low-resolution device.

1. The Verdict Selector determines whether a case is considered closed or open, and whether or not punishments are levied and/or calculated in other parts of GreekDB.

2. Marking the Text Alert box allows the user to notify the members involved in a case that a punishment has been levied or will be discussed regarding their

behavior, and some information about the case that was submitted. Whichever value is selected in the Verdict Selector determines the type of message that is sent to the member's phone.

Judicial Case Editor

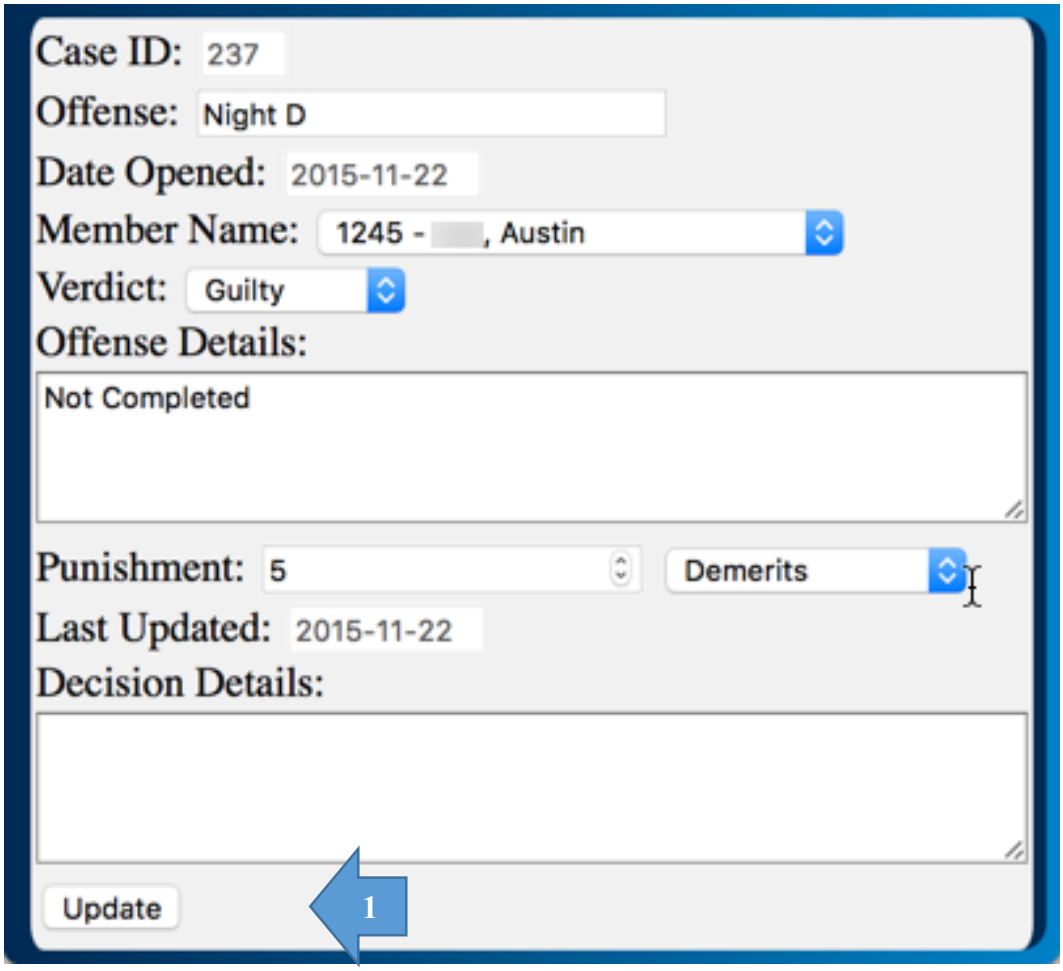


Figure 27. Judicial case editing tool.

The Judicial Case Editor is similar to the Service Event Editor (p. 156), except for Judicial Cases. Because the Creator tool stores records as single-member service records, the editor is only used to modify one record at a time. The Judicial Case Editor is opened in a pop-up window when the user clicks the link to add a new case. The window is independently-scrollable in case it is used on a low-resolution device.

1. Unlike the Service Event Editor, there is no Delete link for Judicial Cases. This is intentional, and is to ensure cases are being created only when necessary, and to ensure that all cases follow due-process and eventually reach a final decision.

Officer Permissions Editor

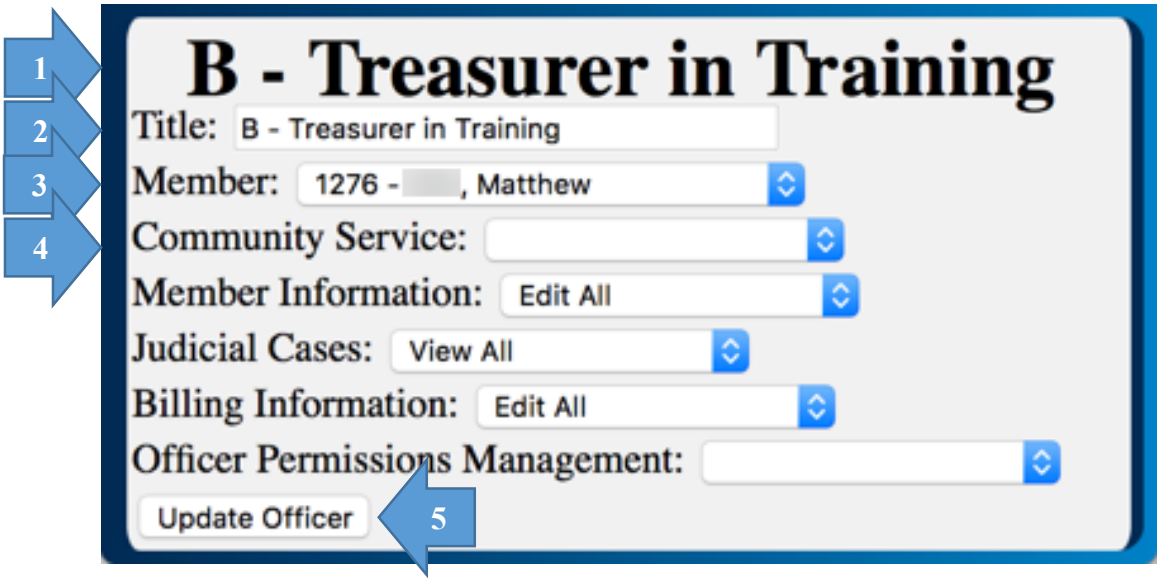


Figure 28. Officer permissions editor.

The Officer Permissions editor allows a user with the correct permissions to modify officer permissions and titles. The Officer Permission Editor is opened in a pop-up window when the user clicks the link to edit an officer. The window is independently-scrollable in case it is used on a low-resolution device.

1. The officer's title is displayed at the top of the page. This is updated on-submission, so if a user changes the title of the officer, it will display the other title until he/she clicks the Update Officer button to commit the changes.

2. The user may update the title for the officer. The field allows up to 40 characters for the officer title.

3. A member selector is printed which contains a list of all active members. This does mean that any members who are listed as alumni, inactive, or deceased are ineligible and will not be displayed.

4. Each of the modifiable permissions is displayed adjacent the type of permissions. The drop-down contains all available permission levels for the officer.

5. The Update Officer button submits any permission changes and title changes made by the user. The button is displayed in the user's default operating system or web browser style.

Easy-Biller

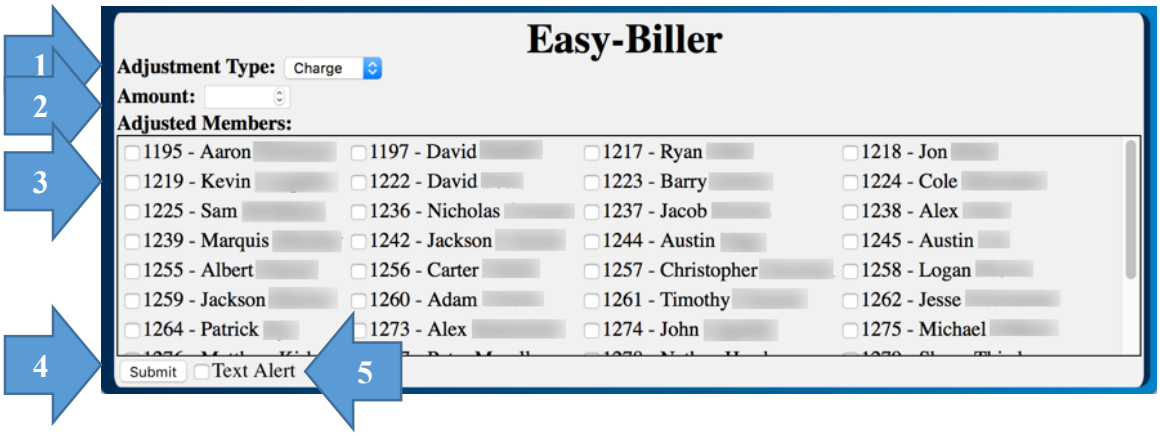


Figure 29. Chapter billing tool.

The Easy-Biller allows officers with the appropriate permissions to bill/credit members within the system and to notify them via SMS that they have been billed or that their transaction has gone through. The Easy-Biller is opened in a pop-up window when the user clicks the link to open Easy-Biller. The window is independently-scrollable in case it is used on a low-resolution device.

1. The Adjustment Type selector allows a user to choose between charging a member and crediting him/her if he/she has made a payment.

2. Amount is a number selector. The user can type a value or (if the browser supports it) use the increment/decrement arrows that are displayed by supporting browsers. The minimum value for the field is $0.00, the maximum value for the field is $9999.99, and the minimum resolution of the increment/decrement arrows is $.01.

3. The Adjusted Members is a member-selector block. It contains a list of all active members, and any members (active or inactive/alumni) that have a non-zero balance (owe money or are owed money). This member-selector box is printed with 4 columns. See Figure 10 for more information (p. 138).

4. The Submit button is printed in the user's browser/operating system default style. Clicking submit stores the transactions for each of the selected members and kicks off the SMS notification process if selected.

5. The Text Alert selector allows the user to notify any selected members of any changes to the financial status. Bills and credits display different messages, but both display the amount that was billed/credited, what kind of transaction it was (bill or credit), and the user's new balance.

Sample SMS Notification

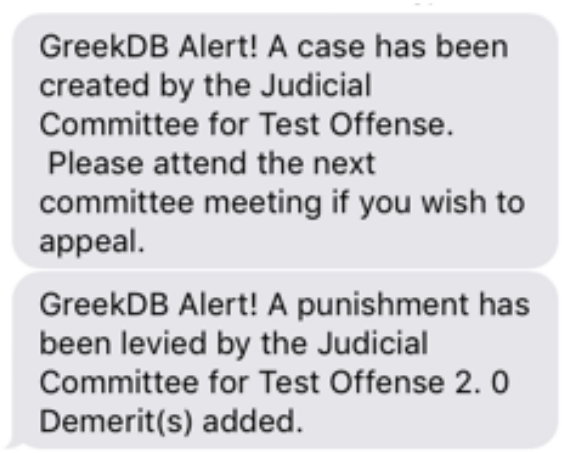
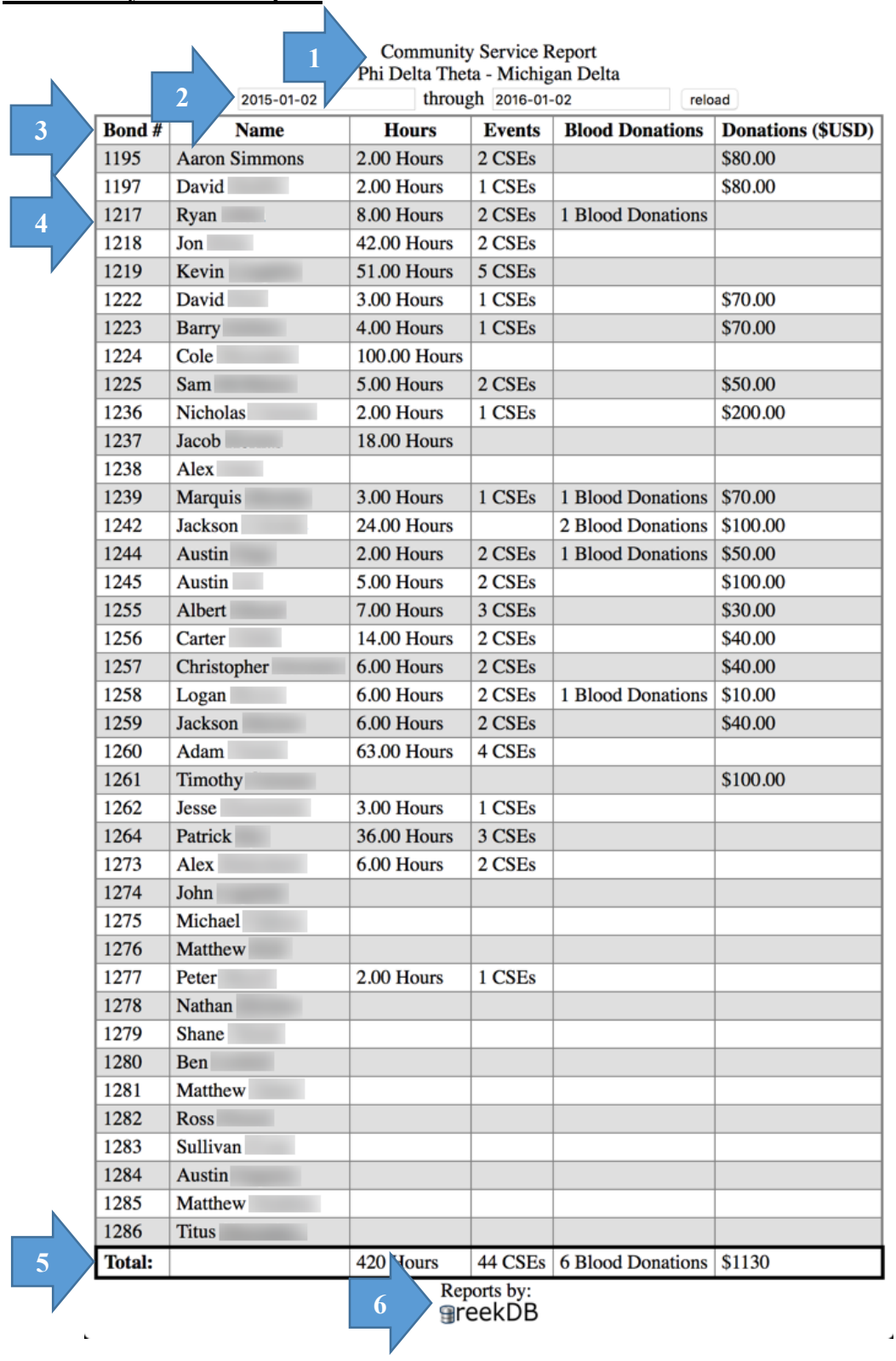


Figure 30. Sample SMS notification.

SMS Notifications are allowed for various tools throughout GreekDB. Depending on the tool, different messages will be displayed that attempt to cover all relevant information. All automated messages coming from GreekDB will come with a prefix "GreekDB Alert!" followed by the message. This means that an announcement from the Mass-Announcer is excepted from this rule. All automated messages from GreekDB should also be less than 160 characters (unless some user-supplied information is unusually long).

Community Service Report



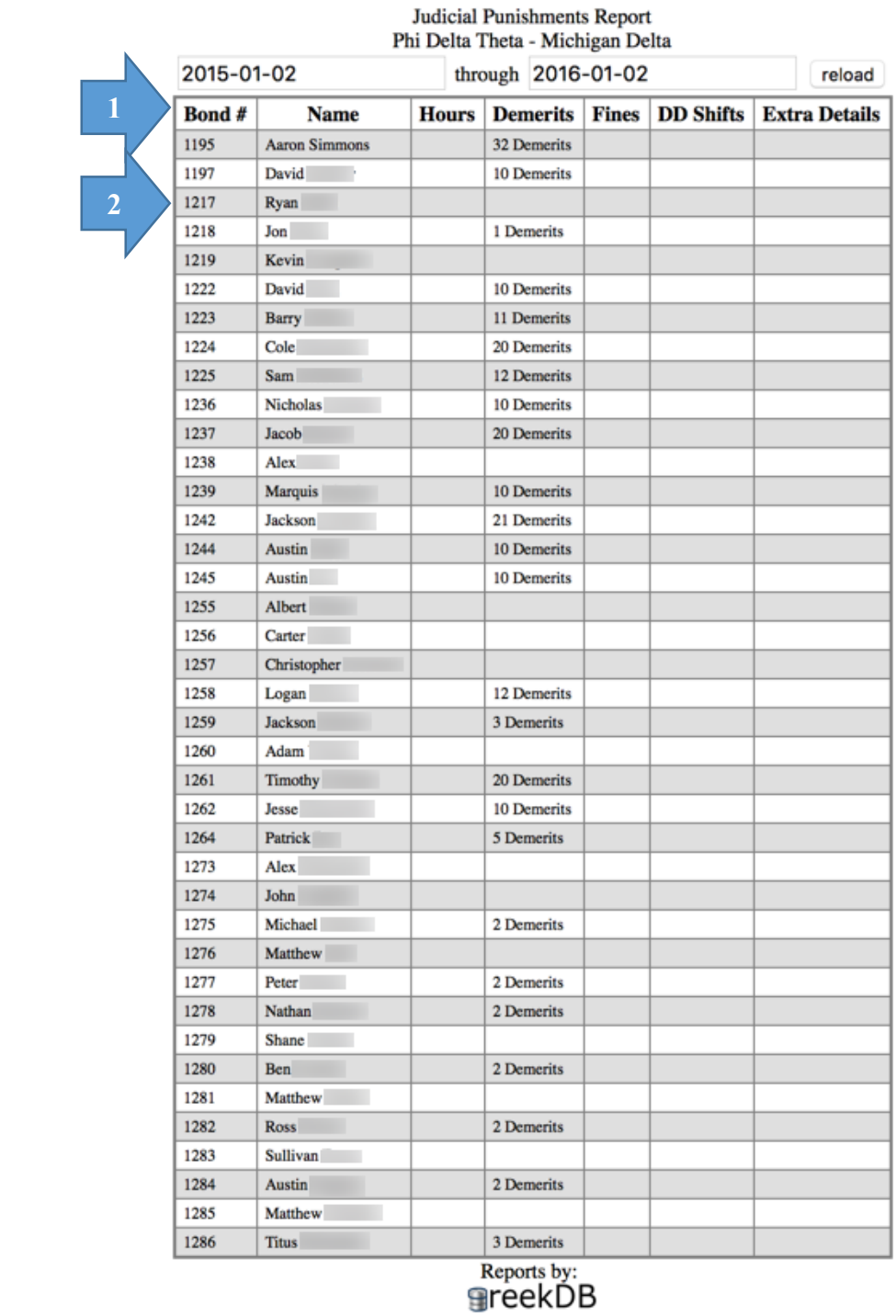
Bond #	Name	Hours	Events	Blood Donations	Donations ($USD)
1195	Aaron Simmons	2.00 Hours	2 CSEs		$80.00
1197	David	2.00 Hours	1 CSEs		$80.00
1217	Ryan	8.00 Hours	2 CSEs	1 Blood Donations	
1218	Jon	42.00 Hours	2 CSEs		
1219	Kevin	51.00 Hours	5 CSEs		
1222	David	3.00 Hours	1 CSEs		$70.00
1223	Barry	4.00 Hours	1 CSEs		$70.00
1224	Cole	100.00 Hours			
1225	Sam	5.00 Hours	2 CSEs		$50.00
1236	Nicholas	2.00 Hours	1 CSEs		$200.00
1237	Jacob	18.00 Hours			
1238	Alex				
1239	Marquis	3.00 Hours	1 CSEs	1 Blood Donations	$70.00
1242	Jackson	24.00 Hours		2 Blood Donations	$100.00
1244	Austin	2.00 Hours	2 CSEs	1 Blood Donations	$50.00
1245	Austin	5.00 Hours	2 CSEs		$100.00
1255	Albert	7.00 Hours	3 CSEs		$30.00
1256	Carter	14.00 Hours	2 CSEs		$40.00
1257	Christopher	6.00 Hours	2 CSEs		$40.00
1258	Logan	6.00 Hours	2 CSEs	1 Blood Donations	$10.00
1259	Jackson	6.00 Hours	2 CSEs		$40.00
1260	Adam	63.00 Hours	4 CSEs		
1261	Timothy				$100.00
1262	Jesse	3.00 Hours	1 CSEs		
1264	Patrick	36.00 Hours	3 CSEs		
1273	Alex	6.00 Hours	2 CSEs		
1274	John				
1275	Michael				
1276	Matthew				
1277	Peter	2.00 Hours	1 CSEs		
1278	Nathan				
1279	Shane				
1280	Ben				
1281	Matthew				
1282	Ross				
1283	Sullivan				
1284	Austin				
1285	Matthew				
1286	Titus				
Total:		420 Hours	44 CSEs	6 Blood Donations	$1130

Figure 31. Community service contribution report.

The Community Service Report contains a summary of all community service totals for a user-specified time period. The page is rendered so that it is printer-friendly for 8.5x11 stock. On-screen is appears as a single table, but contains code that tells the browser to render a page-break when printing to ensure a clean change between pages without splitting a row. The Community Service Report is opened in a pop-up window when the user clicks the link to open the report. The window is independently-scrollable in case it is used on a low-resolution device.

1. At the top of the page is printed the name of the tool, followed by the organization name and chapter designation.
2. Rendered next is the start date and end date for which the report is generated. The default range is 1 year prior to the current day, through the current day. The user may modify this at his/her discretion and click the reload button to regenerate the report. The start and end dates are also passed in via the URL, so a user can bookmark or send a date range to another user.
3. This first row contains a column title for each member name, his/her ID, and finally, each supported donation type. These include:
    a. Community Service Hours
    b. Community Service Events
    c. Blood Donations
    d. Monetary Donations
4. Each active member has a row generated on his/her behalf. If a member has made no donations for a specific donation type, the corresponding table cell is printed blank to appear cleaner. Each populated cell includes the units being measured for easier interpretation.
5. The final row contains the sum of all listed event types, as well as their corresponding units.
6. At the bottom of the page appears the GreekDB Logo.

Judicial Punishments Report



Judicial Punishments Report
Phi Delta Theta - Michigan Delta

2015-01-02 through 2016-01-02 reload

Bond #	Name	Hours	Demerits	Fines	DD Shifts	Extra Details
1195	Aaron Simmons		32 Demerits			
1197	David		10 Demerits			
1217	Ryan					
1218	Jon		1 Demerits			
1219	Kevin					
1222	David		10 Demerits			
1223	Barry		11 Demerits			
1224	Cole		20 Demerits			
1225	Sam		12 Demerits			
1236	Nicholas		10 Demerits			
1237	Jacob		20 Demerits			
1238	Alex					
1239	Marquis		10 Demerits			
1242	Jackson		21 Demerits			
1244	Austin		10 Demerits			
1245	Austin		10 Demerits			
1255	Albert					
1256	Carter					
1257	Christopher					
1258	Logan		12 Demerits			
1259	Jackson		3 Demerits			
1260	Adam					
1261	Timothy		20 Demerits			
1262	Jesse		10 Demerits			
1264	Patrick		5 Demerits			
1273	Alex					
1274	John					
1275	Michael		2 Demerits			
1276	Matthew					
1277	Peter		2 Demerits			
1278	Nathan		2 Demerits			
1279	Shane					
1280	Ben		2 Demerits			
1281	Matthew					
1282	Ross		2 Demerits			
1283	Sullivan					
1284	Austin		2 Demerits			
1285	Matthew					
1286	Titus		3 Demerits			

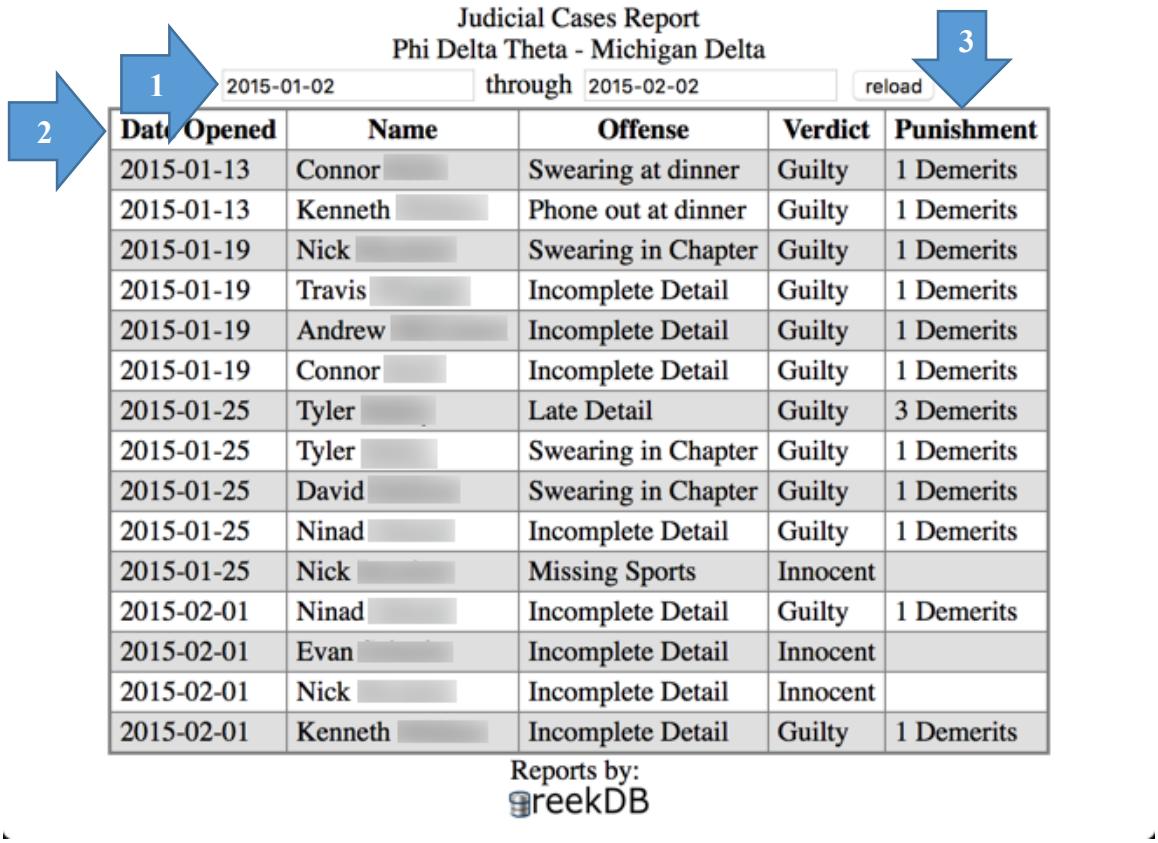
Reports by:
greekDB

Figure 32. Judicial case results report.

The Judicial Punishments Report is in the same style as the Community Service Report (p. 163), and is rendered in the same printer-friendly way. The Judicial Punishments Report is opened in a pop-up window when the user clicks the link to open the report. The window is independently-scrollable in case it is used on a low-resolution device.

1. This first row contains a column title for each member name, his/her ID, and finally, each supported punishment type. These include:
    a. Demerits (marks, checks, etc.)
    b. Monetary Fines
    c. Additional Designated Driver shifts (where applicable)
    d. Extra Cleaning Details (where applicable)
2. Each active member has a row generated on his/her behalf. If a member has received no punishments for a specific donation type, the corresponding table cell is printed blank to appear cleaner. Each populated cell includes the units being measured for easier interpretation.

Recent Cases Report



Date Opened	Name	Offense	Verdict	Punishment
2015-01-13	Connor	Swearing at dinner	Guilty	1 Demerits
2015-01-13	Kenneth	Phone out at dinner	Guilty	1 Demerits
2015-01-19	Nick	Swearing in Chapter	Guilty	1 Demerits
2015-01-19	Travis	Incomplete Detail	Guilty	1 Demerits
2015-01-19	Andrew	Incomplete Detail	Guilty	1 Demerits
2015-01-19	Connor	Incomplete Detail	Guilty	1 Demerits
2015-01-25	Tyler	Late Detail	Guilty	3 Demerits
2015-01-25	Tyler	Swearing in Chapter	Guilty	1 Demerits
2015-01-25	David	Swearing in Chapter	Guilty	1 Demerits
2015-01-25	Ninad	Incomplete Detail	Guilty	1 Demerits
2015-01-25	Nick	Missing Sports	Innocent	
2015-02-01	Ninad	Incomplete Detail	Guilty	1 Demerits
2015-02-01	Evan	Incomplete Detail	Innocent	
2015-02-01	Nick	Incomplete Detail	Innocent	
2015-02-01	Kenneth	Incomplete Detail	Guilty	1 Demerits

Figure 33. Recent judicial case summary report.

The Recent Judicial Cases Report is formatted the same as the previous two reports (Judicial Punishments Report p. 165, and Community Service Report p. 163). The Recent Judicial Cases Report is opened in a pop-up window when the user clicks the link to open the report. The window is independently-scrollable in case it is used on a low-resolution device.

1. Like the previous two reports, the start date and end date for which the report is generated. Unlike the previous two however, the default range is 1 month prior to the current day, through the current day. The user may modify this at his/her discretion and click the reload button to regenerate the report. The start and end dates are also passed in via the URL, so a user can bookmark or send a date range to another user.

2. This first row contains a column title for each the date the case was opened, the member's name, the offense title, the resulting verdict, and whatever punishment was levied.

3. The punishment totals column only prints a punishment if a verdict of guilty was reached. If any other verdict was reached, the corresponding cell is printed blank. Where a punishment is levied, both a quantity and a unit/type is also printed.

Accounts Receivable Report

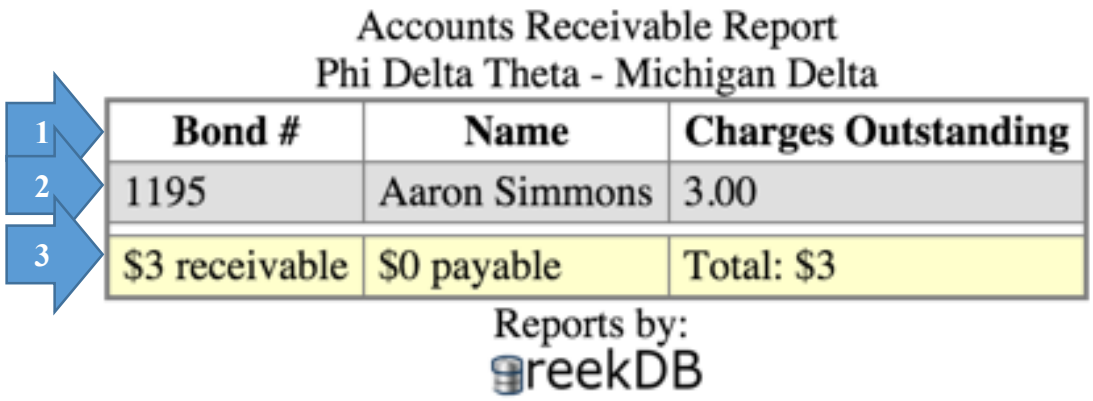


Figure 34. Accounts receivable report.

The Accounts Receivable Report is formatted the same as the previous three reports (Judicial Punishments Report p. 165, Community Service Report p. 163, and the Recent Judicial Cases Report p. 167). The Accounts Receivable Report is opened in a pop-up window when the user clicks the link to open the report. The window is independently-scrollable in case it is used on a low-resolution device.

Unlike the previous 3 reports however, the Accounts Receivable Report takes no date parameters. It generates a row for every member with a non-zero balance, regardless of whether they are active, inactive/alumnus, or deceased.

1. The top row of the table contains the column headers for the table data, including the member's unique ID (Bond #), the member's name, and the amount

outstanding (positive number indicated money owed to the organization, negative numbers indicate money owed by the organization to members)

2. Each member with a non-zero balance will have a record in the report indicating his/her balance.

3. The final row is separated by a spacer for legibility and colored yellow to indicate it's the end of the report. This row contains the total amount owed to the organization, the total amount owed by the organization to its members, and the net balance, accounting for both.

Chapter Roster(s)

Figure 35. Chapter roster print-out.

The Chapter Roster contains a printer-friendly list of all members. It consists of two different reports. The first report is the Active Roster, which contains only active

members. It is also capable of displaying the full roster, including all alumni and some basic information about them as well. As with the previous reports, the page is rendered such that on-screen is continuous, but when printed it will automatically insert page-breaks so rows don't get split between pages when printed on 8.5x11 stock.

1. This link allows the user to switch between the full roster and the active roster. The link is displayed with the text of the current report the user is viewing. Clicking the link will reload the report so that it displays the other report.

2. The first column in the report contains the number of the member printed (eg. The first member printed will display as 1, the second will be displayed as 2, and so on. Below that is printed the member's unique ID. Finally, below that is an empty checkbox. This is a convenience so a user may use the page for logging data on- and off-line. Checking the box in the report has no function other than appearing checked, so the user and his/her browser is responsible for retaining the page and its state, or printing it and logging manually on a printed copy.

3. The second column contains the member's profile picture. This is the same picture that is displayed on the Member Information page for the member.

4. The last column contains 4 pieces of information:

    a. The member's first and last name

    b. The member's college major

    c. The member's phone number

    d. The member's email address

    All of this information is information that was populated on the Member Info page.

5. Each member is printed with his own record, and each cell within has a character limit to ensure a standard page width.

6. After the bottom of the page is reached on the left side, the report renders a second column of member records. Only after the second column is rendered, does the report continue onto a second page. This is repeated until all records have been printed.

X. DETAILED SCENARIOS

Explanation

The following ten scenarios are designed to give a snapshot of the communication that occurs between different components (functions, tables, APIs, etc.) of the system, and the external services that the system utilizes. The enumerated textual scenarios lay out in detail what exactly occurs during that action. The associated collaboration diagram for each, displays in a graphical format the communication between the different components that is necessary for the system to function. Along the top of each collaboration diagram are the names of the components necessary for that scenario; the line extending down from that box represents the part. Arrows between lines represent communication between components; and solid bold lines on a class's vertical line represent an action taken internally to a component. Please note that in the diagrams, the MySQL handler is absent for any operations that do not manipulate the data that is returned to the application, or do not generate new data. The MySQL handler should be thought of as the SQL Interpreter in the MySQL database.

Determination

The five scenarios that follow were primarily derived from the use cases described in USE CASES (pp. 47). In some cases, alternate flows for use cases were split into multiple scenarios; this allows for easy distinction between the two.
In general, a use case was taken and examined in detail to determine what exactly happens being the scenes (on the System side) of that use case; it was mapped out to the classes specifically, and double checked against the static class model to ensure conformity. In the event it was discovered that a method did not exist yet that was

implied in the use case or enumerated requirements, that method was added to the updated static class model.

A collaboration diagram is designed to show the interactions between different parts of the system in order to complete a task. Because of the nature of the application, many of the tools require simple insertions, updates, or recalls of data in a single table. These kinds of interactions will not be shown, as there are many of them and they are non-complex. For the remaining tools, in the diagrams, only parts of the system that perform an action will be displayed in each diagram. It is also important to note that for each of these diagrams, they are created with the assumption of a successful completion. Cases of a user attempting a task for which he/she is not allowed will not be diagrammed. These five scenarios were chosen because they exhibit complexity and/or exhibit behavior that is typical of other tools that perform similar operations (like sending text messages)

User Login

Brief description

The user logs into GreekDB.

Detailed description

The user enters his/her login information, and clicks the login button. The system retrieves the matching account information. The system hashes the user's password and checks that the hashed password matches the one that was stored in the database. If it does, the system unloads the password and hash information, and requests permission data from the database. After calculating effective permissions, the system stores them in the user's session information and redirects the user to the landing page.

Creation Date: 1/30/2016 Update Date: 1/30/2016

Version #: 1.0

1. The user enters a Username and Password, and clicks Login.
2. The Application requests the User Account information from the Users table in the database, based on the username entered by the user.
3. The database returns the hashed password, password salt, and email address.
4. The system hashes the password entered by the user, using the salt that was retrieved from the user record for the matching username.
5. The system detects a match between the user-entered password and the password stored in the database.
6. The system unloads the login information from the session information.
7. The system requests the permissions delegated by the organization, the chapter, and the offices that the user holds.
8. The database returns the permissions the organization, chapter, and officer-delegated permissions.
9. The system calculates the effective permissions from these 3 permission sets, and applies system defaults for permissions which are not set by the afore-mentioned 3.
10. The system saves these to the user's session information.
11. The system redirects the user to the landing page.

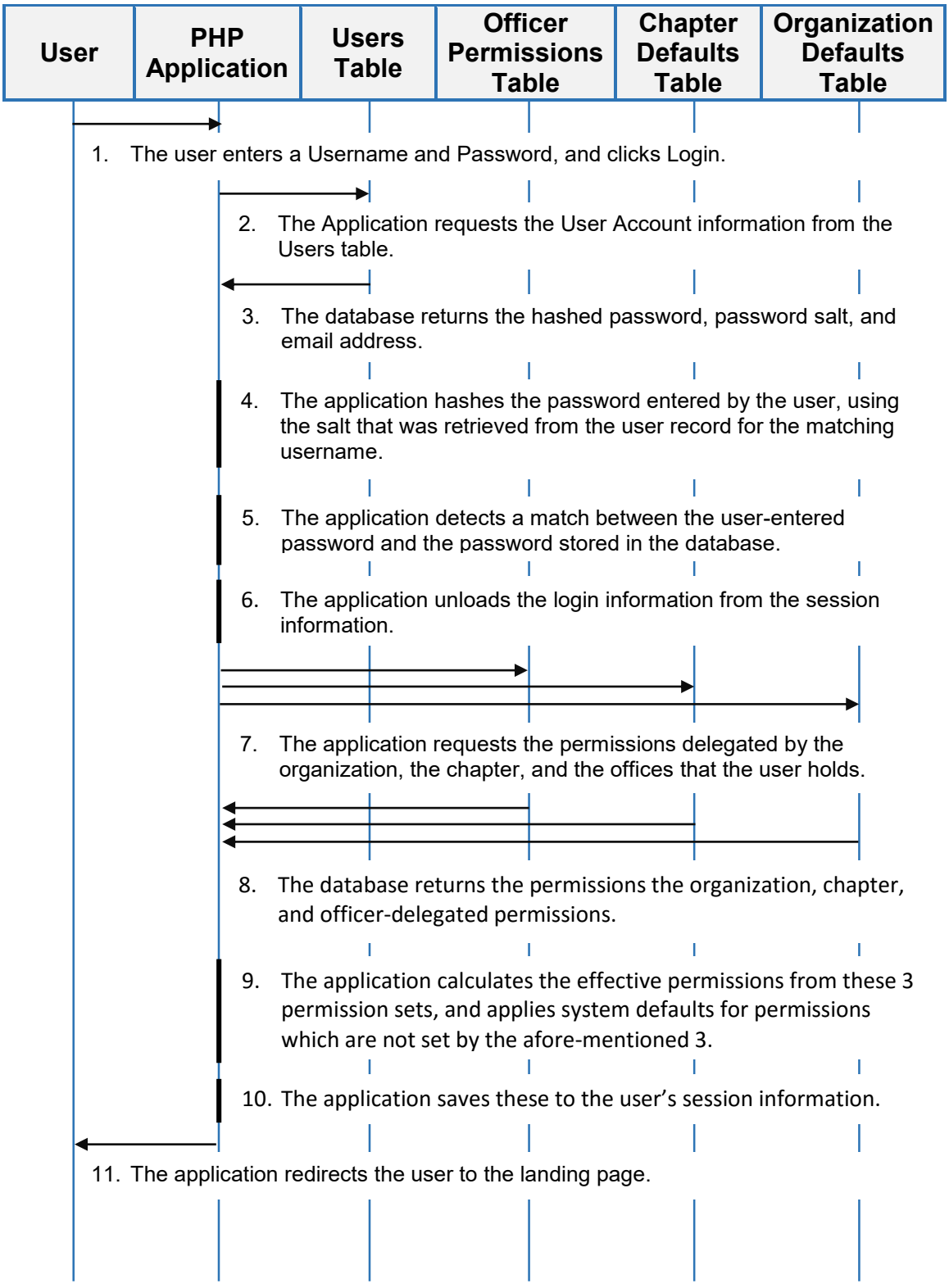


Figure 36. User login collaboration diagram.

Load Accounts Receivable Report

Brief description

The user loads the Accounts Receivable Report.

Detailed description

The user clicks the Accounts Receivable link. The Application requests a list of members requests a list of members with non-zero balances from the database. The database retrieves a list of all member charges and payments. The database then retrieves a list of all fines from judicial cases. The database combines the results and returns a list of all members with non-zero balances to the application. The application then sums all of the payable values, the owed values, and then generates a complete sum. The application prints a table of the users and their balances owed/payable. The application then prints a row containing each of the summed values.

Creation Date: 1/30/2016 Update Date: 1/31/2016

Version #: 1.0

1. The user clicks the Accounts Receivable link.
2. The application requests a list of all members with a non-zero balance in their accounts from the database.
3. The database retrieves a list of all member charges.
4. The database retrieves a list of all member fines from judicial cases, and the amount of the fines.
5. The MySQL handler then sums all of the transactions and keeps only those without a net-zero balance.
6. The database requests a list of member names and IDs for the members who were kept in step 5.
7. The database returns the filtered list of members and their balances.
8. The application generates a table of members with their outstanding balances.
9. The application then sums all of the positive balances, all of the negative balances, and the overall sum, and adds these values as the last row in the table.
10. The application prints the page to the user's browser.

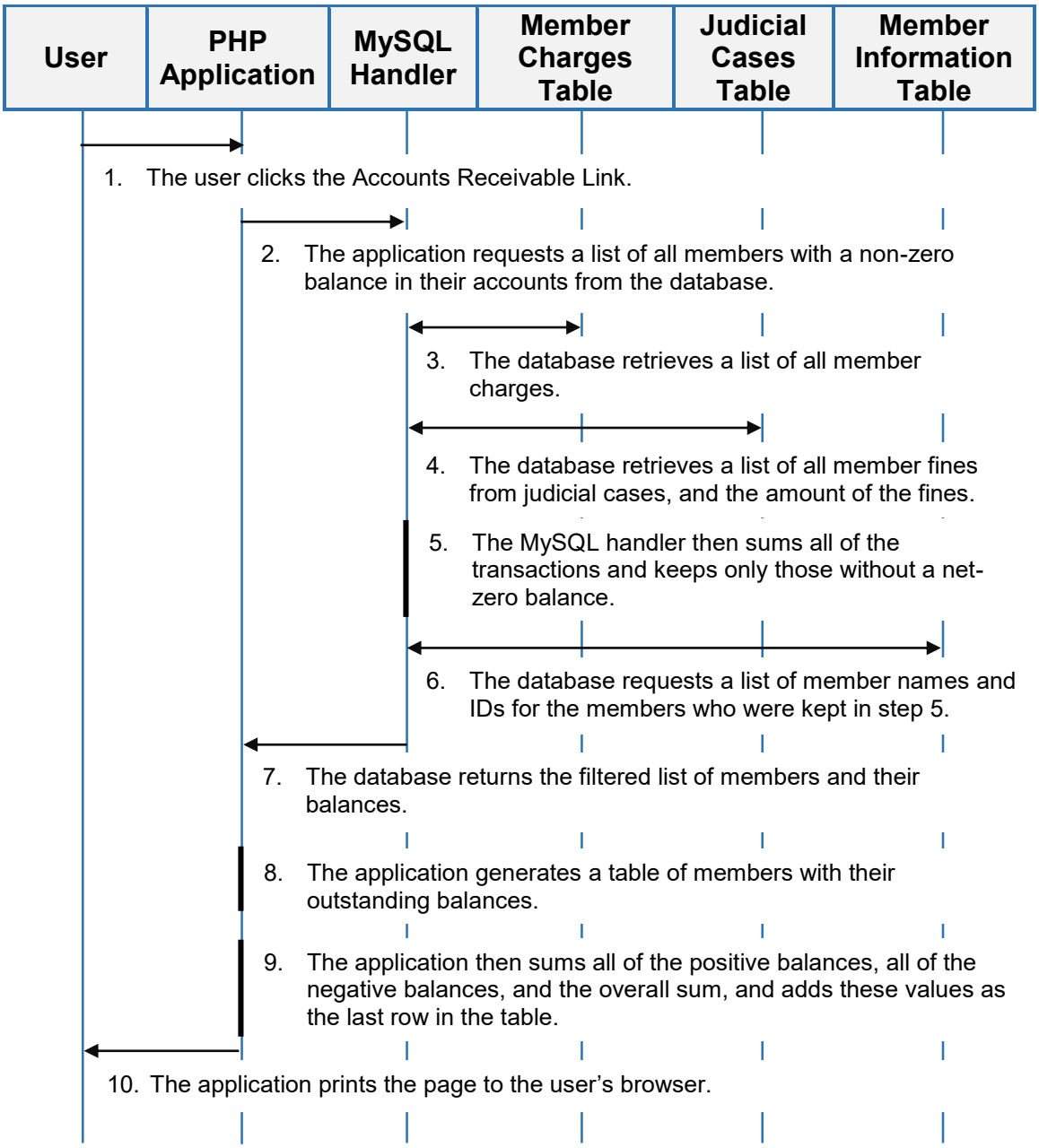


Figure 37. Load accounts receivable report collaboration diagram.

Submit Judicial Case w/ Text Notification

Brief description

The user enters case information, marks the Text Notification box, and clicks submit. The application stores the case information and sends a text message to each member involved in the case.

Detailed description

The user enters case information, marks the Text Notification box, and clicks submit. The system inserts a new case for each involved member into the database. The application then forms a message from the punishment (if one was levied), and the case title, and sends one text message at a time to the members involved in the case. The phone number is gathered for each member from the phone number listed in the members' information pages.

Creation Date: 1/31/2016 Update Date: 1/31/2016

Version #: 1.0

1. The user fills in any case information, checks the Text Alert box, and clicks submit.
2. The application asks the database to insert one copy of the judicial case for each member that was selected.
3. The application requests the phone number from the database for each member that was involved.
4. The database returns a list of phone numbers for each of the members involved if one exists in the member information table.
5. The application crafts a message from the case information, and includes a punishment quantity if the case resulted in a guilty verdict.
6. The application submits the message to the SMS Handler, which is responsible for communication with the SMS Provider.
7. The SMS Handler sends the first message to the SMS Provider in the list.
8. The SMS Handler waits 1 second per the provider's requirements, and then sends the next message in the list.
9. Once the list has been completed, the SMS Handler signals completion to the application.
10. The application closes the pop-up window.
11. The text message is received by the members' phones.

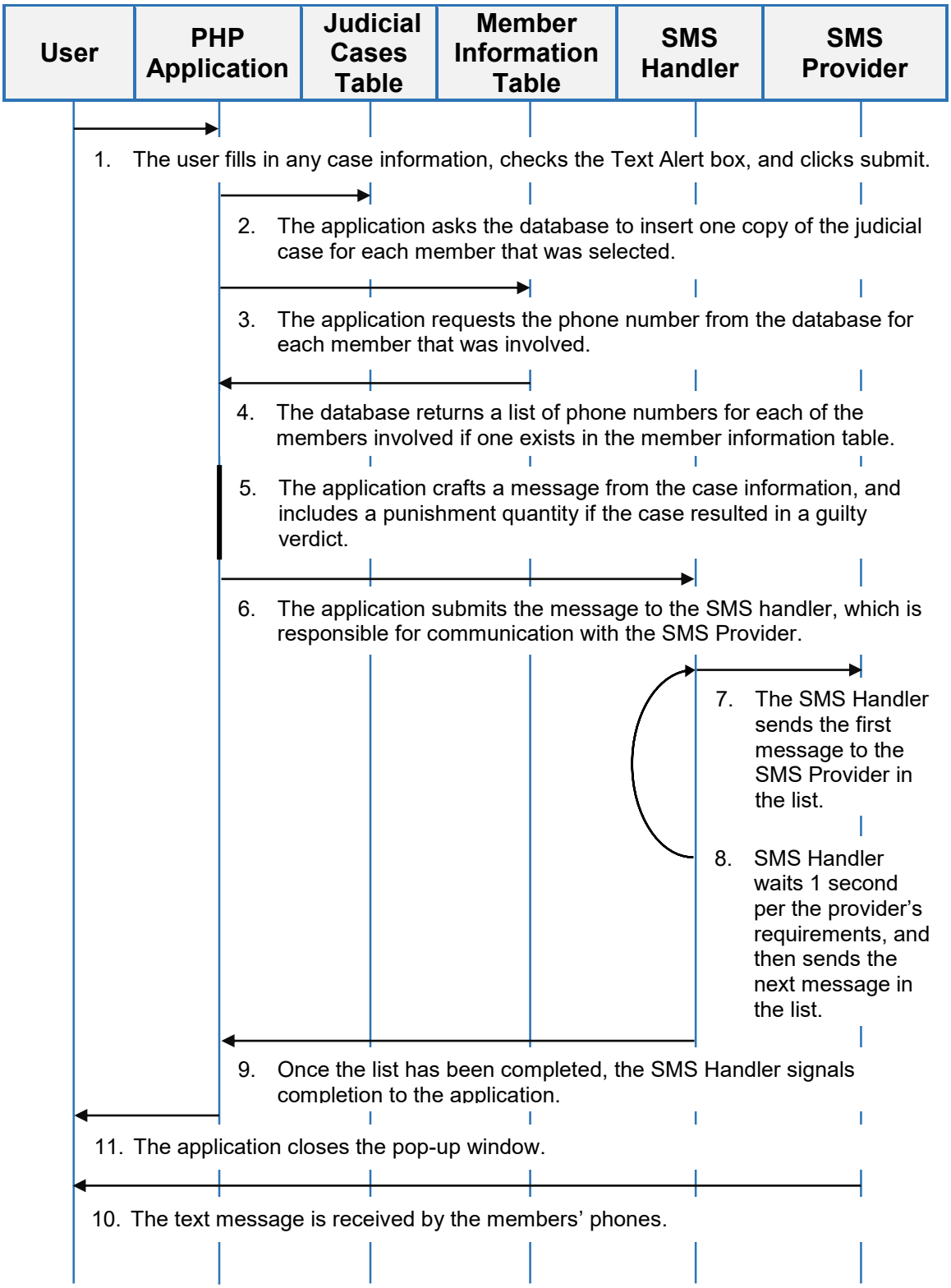


Figure 38. Submit judicial case w/ text notification collaboration diagram.

Load the Navigation Banner

Brief description

The user loads any page that is in either the 2-panel or the 3-panel format. The navigation banner is customized for the user that is logged in.

Detailed description

The user loads any page that is in either the 2-panel or the 3-panel format. The navigation banner checks that the user may edit his/her own member profile. He/she is allowed, so the Member Tools header is printed, followed by a link to the member's profile. The system then checks the member's privileges to the Chapter Directory, Community Service Events, Judicial Cases, Officer Permissions Management, and Chapter Announcements. For the Chapter Directory, Community Service Events, and Judicial Cases, if the user has rights to view their own information, but is not an admin, links to those tools are printed under the Member Tools. The system then prints the Officer Tools header. For all of the tools except for the financial tools, if the user is an admin, the links to the tools are printed under the Officer Tools header. If the user has admin rights to any of the financial tools, the Treasurer Tools header is printed, and then the links to those tools are printed. The Reports header is printed. For any reports that the user has read-all rights to, links to the reports are printed.

Creation Date: 1/31/2016 Update Date: 1/31/2016

Version #: 1.0

1. The user opens any page that is in either the 2-panel or the 3-panel format.
2. The application prints the Member Tools header.
3. The application checks the user's permissions for the Chapter Directory, Judicial Cases, and Community Service Events. The user is an admin to the tools, so only the Member Profile link is printed.
4. The application prints the Officer Tools header and links to the Chapter Directory, Community Service Events, and Judicial Cases.
5. The application checks the user's permissions to the Officer Permissions Manager and the Mass-Announcer. He/she is an admin.
6. Links to the tools listed in step 5 are printed.
7. The application checks the user's permissions to the financial tools. He/she is an admin.
8. The application prints the Treasurer Tools sub-header, followed by the Open Accounts and Easy-Biller links.
9. The application prints the Reports header.
10. The application checks that the user has read-all access for the Community Service, Judicial Cases, Chapter Directory, and Financial Information. He/she does.
11. The application prints links to the Service Totals, Punishment Totals, Recent Cases, Accounts Receivable, and Chapter Roster reports.

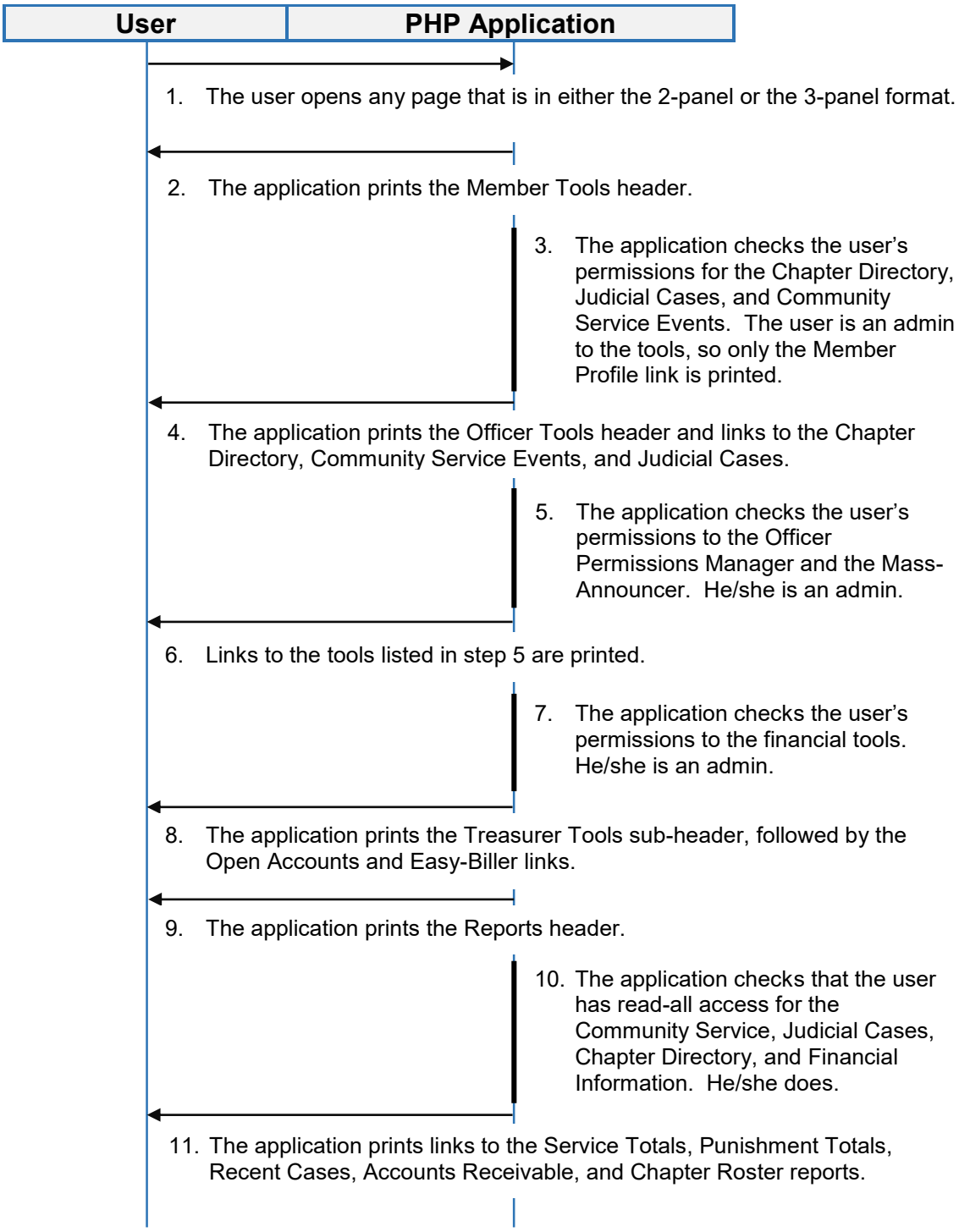


Figure 39. Load navigation banner collaboration diagram.

Load the Landing Page

Brief description

The user clicks the GreekDB logo or completes a successful login. The user is redirected to the landing page. The page display's some basic information about the user and some information about the offices the user holds.

Detailed description

The user clicks the GreekDB logo or completes a successful login. The user is redirected to the landing page. The system prints the user's community service for the last 6 months, the overall results of any judicial cases from the last 6 months, and the user's current financial status. If the user has admin rights to Community Service Events, Judicial Cases, or Financial Information, the system prints officer-relevant information for each, respectively.

Creation Date: 1/31/2016 Update Date: 2/1/2016

Version #: 1.0

1. The user is redirected to the landing page by either clicking the GreekDB logo or completing a successful login.
2. The application prints the welcome message.
3. The application prints the Member Information header.
4. The application requests the user's total community service contributions for each category, for the last 6 months.
5. The MySQL Handler retrieves all of the users' community service events for the last 6 months from the Service Events table.
6. The MySQL Handler sums all of the contributions based on the contribution type.
7. The MySQL Handler returns a table of the sums to the application.
8. The application prints the Community Service sub-header.
9. The application prints the each of the totals with the appropriate labels.
10. The application prints the Standards sub-header.
11. The application requests the results of all Judicial Cases for the user for the last 6 months, from the database.
12. The MySQL Handler retrieves all of the user's judicial cases for the last 6 months from the Judicial Cases table.
13. The MySQL Handler sums all of the judicial case results according to punishment type.
14. The MySQL Handler returns a table of the results to the application.
15. The application prints each of the totals with the appropriate labels.
16. The application prints the Billing sub-header.
17. The application requests the user's overall balance from the database.
18. The MySQL Handler retrieves all of the transactions for the user from the Member Charges table.
19. The MySQL Handler retrieves all of the judicial case results for the user that resulted in a guilty verdict with a fine as a punishment.
20. The MySQL Handler sums all of the financial transactions with the returned judicial case results.

21. The MySQL Handler returns the net result to the application.

22. The application prints the user's current balance.

23. The application checks that the user has admin rights to any of Community Service Events, Judicial Cases, or Financial Information. He/she does.

24. The application prints the Officer Information header.

25. The application checks the user's permissions for Community Service Events. He/she is an admin.

26. The application prints the Community Service sub-header.

27. The application requests the total of each service category for the last 6 months, for the entire chapter, from the database.

28. The MySQL Handler retrieves all of the Community Service records for the last 6 months.

29. The MySQL Handler sums all of the retrieved events according to the contribution type.

30. The MySQL Handler returns the results to the application.

31. The application prints the Service Totals block title, spacer, and the sums for each event type.

32. The application checks the user's permissions to Judicial Cases. He/she is an admin.

33. The application requests the number of cases that are still open for the chapter, from the database.

34. The MySQL Handler counts the number cases that are open for the chapter in the Judicial Cases table.

35. The MySQL Handler returns the count to the application.

36. The application prints the number of pending cases.

37. The application requests a random set of four members who have had judicial cases in the last 6 months, and the total number of cases to each, from the database.

38. The MySQL Handler retrieves a list of all members of the chapter who have had cases in the last 6 months, and the total count of the cases to each member.

39. The MySQL Handler counts the number of cases, grouped by member, and selects a random group of 4 members who have had more than 0 cases.

40. The MySQL Handler returns the list of members and the count of their cases, to the application.

41. The application prints the Last Name of the members and the total count of their cases for the last 6 months.

42. The application checks the user's permissions to the chapter's financial information. He/she is an admin.

43. The application prints the Billing sub-header.

44. The application requests a list from the database of all members who owe money to the chapter.

45. The MySQL Handler retrieves a list of all transactions for the chapter from the Member Charges table.

46. The MySQL Handler also retrieves a list of all judicial cases, which resulted in a guilty verdict and a monetary punishment.

47. The MySQL Handler sums the two lists together according to the associated member.

48. The MySQL Handler returns a list of the 4 members with the highest outstanding balance to the application.

49. The application prints the Payments Outstanding block title and a separator.

50. The application then prints the returned member list with their balances in descending order of the balance owed.

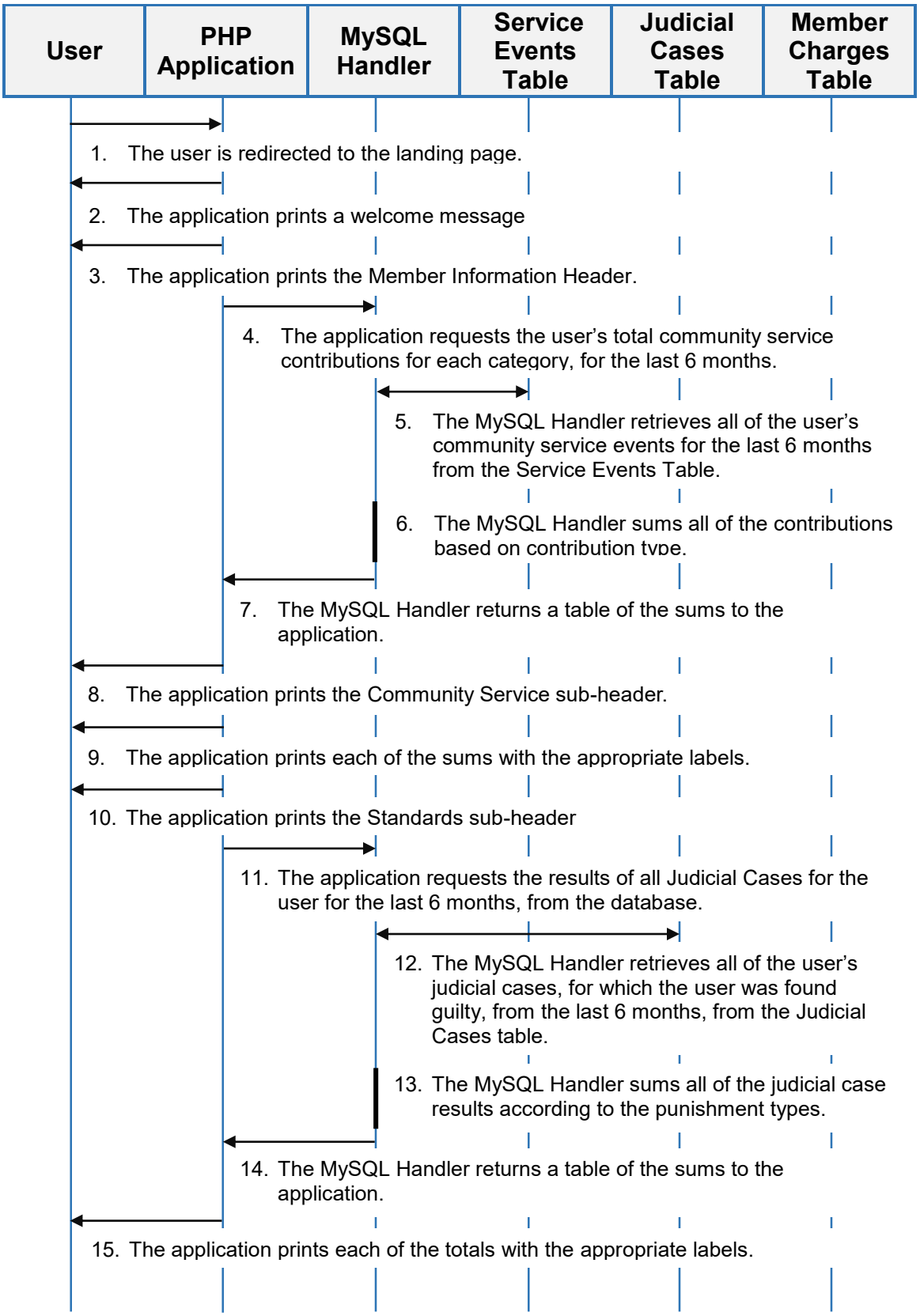


Figure 40. Load landing page collaboration diagram part 1.

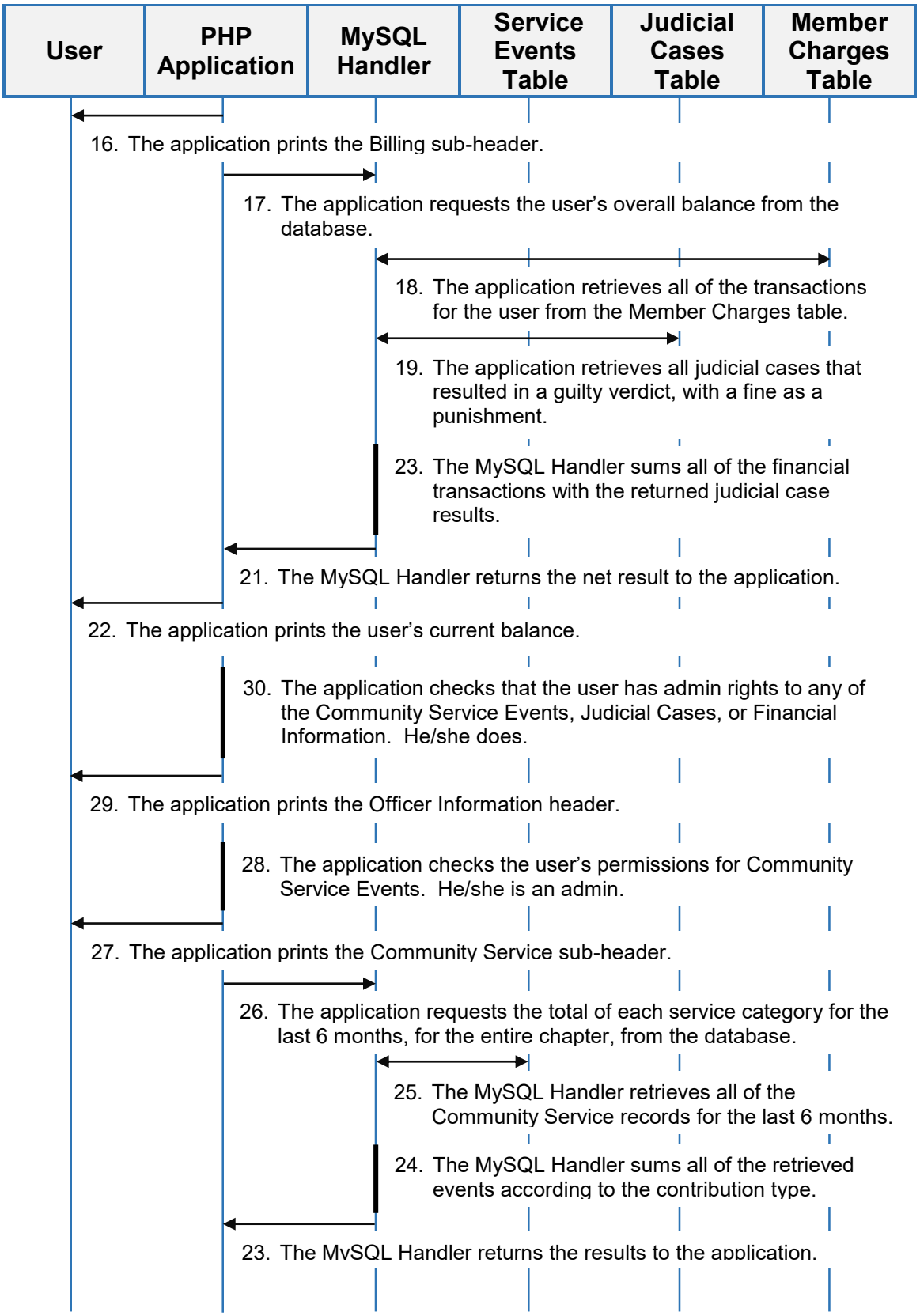


Figure 41. Load landing page collaboration diagram part 2.

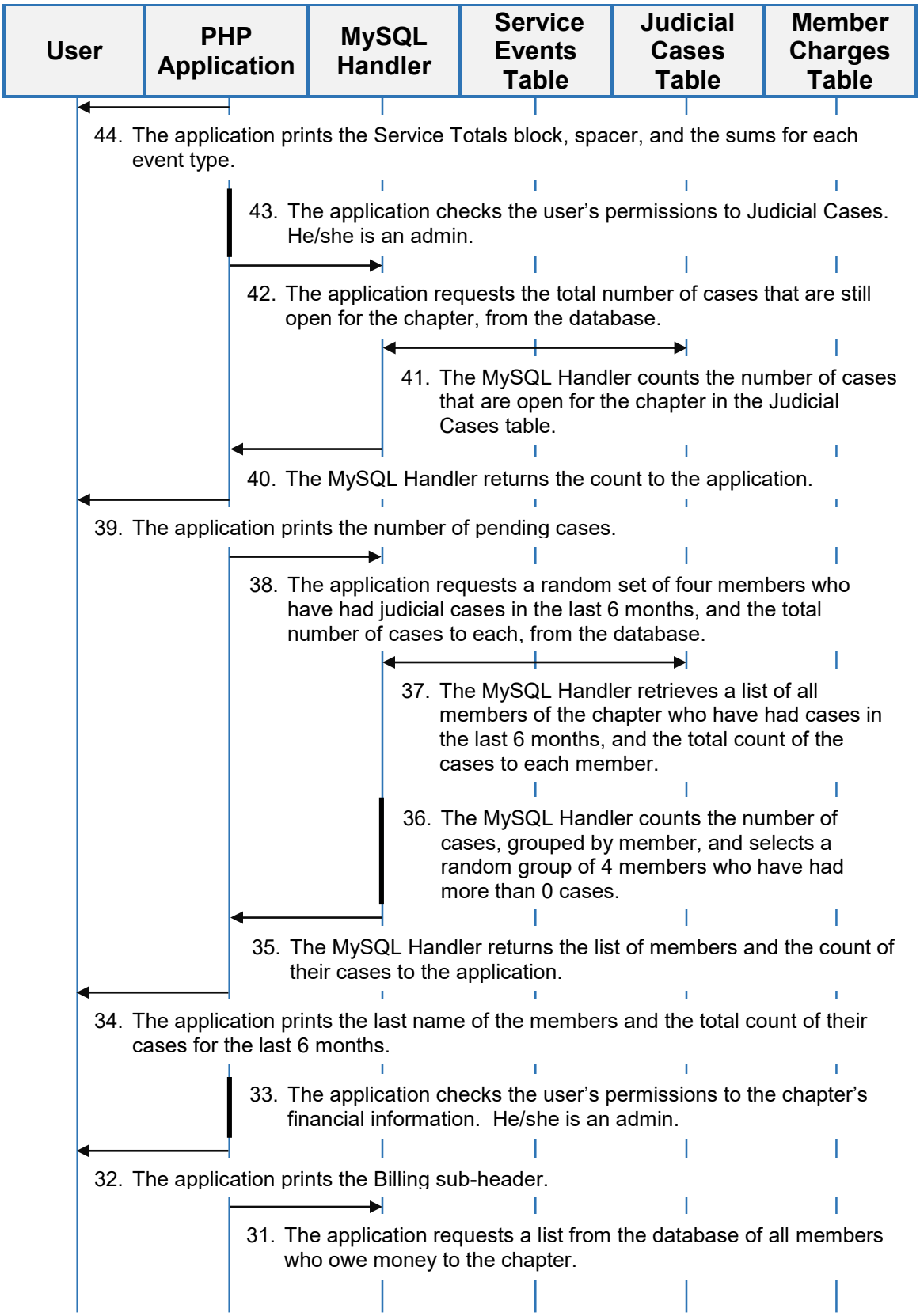


Figure 42. Load landing page collaboration diagram part 3.

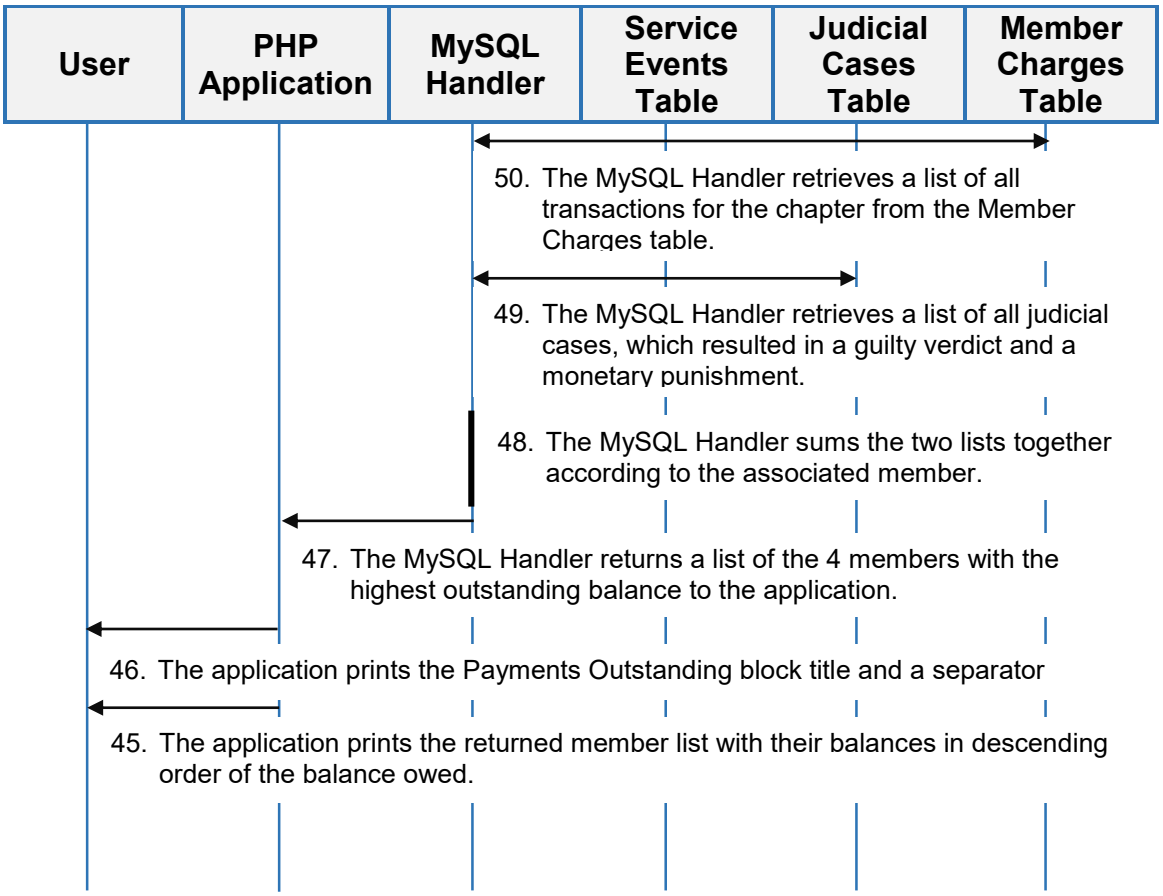


Figure 43. Load landing page collaboration diagram part 4.

XI. APPLICATION DESIGN DECISIONS

This chapter goes into more depth about some applications features, how they were implemented, why they were implemented in the way that they were, any mistakes that were made, and any changes that are expected for the current system.

GreekDB Permissions Handler

Permission structure

I had a few ideas as to how I wanted to implement a permissions system in GreekDB. I could implement the permissions system on a page-by-page basis, on an item-by-item basis, or based what the information actually contained.

The ideal scenario would have been to allow read, read/write, or no access on each of the pieces of information on a page, but unfortunately, I had not built the pages with that information in-mind, nor had I much experience with permissions handling at the time, so I was unable to handle permissions on a field-by-field basis. In addition to this, I would have needed to provide a series of sane default permissions to users, and to hide the complexity from the users, while still making it accessible for power-users. This would have required additional time to implement.

Implementing the permissions on a page-by-page basis would have been an acceptable option as well, but could still be a little too complex for some users without some sane defaults and additional education as-to what different settings would cause.

The option I chose was to implement the permissions based on what the information contained. This means handling all personal information in a single permission configuration, all community service information in a single permission configuration, and so on.

Permission granularity

The next step was to determine how best to cover officer permissions. I not only needed to be able to allow and deny access to a certain record, but I needed to be able to allow a user to access a record for himself, but deny access to the same record, but for another user. For officers, I might need to allow a user access to all records, including ones he/she does and does not own. Finally, in the fringe-case, for officers handling financial information, I needed to be able to allow a user to modify other users' records, but not his/her own records. To make this work, I created 7 different possible permissions for each type of data:

- No Access
- View Self
- Edit Self
- View All
- Edit Self, View Others
- View Self, Edit Others
- Edit All

Nationally-mandated permissions

The final step was to determine how best to allow a national organization to customize the user experience. It is entirely possible for a national organization to want to remove the judicial cases portion of the software, or for a local chapter to desire to limit access to the member directory. It is equally possible that a national organization will want to ensure that each member has a voice in judicial proceedings. Finally, it is very likely than an organization will take no time to customize permissions. This means I

need to provide default options. To handle this, I created a 4-tiered cascading permission system. The 4 tiers in order of precedence are as follows:

1. Officer Roles
2. Chapter-Designated Permissions
3. Organization-Designated Permissions
4. Sane Defaults

This means that in the case that no permissions have been specified for a set of data by any of 1, 2, or 3, the permission for the user will apply the permission specified in 4. If a permission is specified in 3, then that becomes the minimum permission for that field. The same principle applies for 1 and 2, but only if there is not a preceding permission in a previous field (2 and 3, and 3, respectively). An exception is if the No Access permission is set at any level, and the fields before it did not set a value (other than defaults). In this case, that part of the application is blocked.

This schema does require some special handling in some cases. For example: If an organization has specified View All permissions to community service, and a chapter specifies Edit Self, then this permission becomes Edit Self, View All to meet both criteria.

Lessons Learned

Given more time, I would definitely go back and set permissions on at a minimum, page-by-page basis, and likely a field-by-field basis. In cases where the same data is displayed and modifiable on two different pages (which should be few), these will reference the values of the other fields. To manage this level of granularity, I will have to redo the permissions management editor. I expect to do this by having a mock-up of

each page with sample values in each blank, and selections available for each field as-to what the permission value should be.

In addition to this, logging will be added to each record, noting change attempts, and whether or not the action passed or failed.

Database Member Heirarchy (Organization, Chapter, Member)

I initially set up the database to model the organizational hierarchy of the organizations for whom I was building the tools. This initially seemed like a good idea because it meant that I could have a large number of organizations, chapters, and users, at each level, and my foreign keys would all stay fairly small (1,000,000,000 members would require an average key size of 4 digits, using only integers for the keys/indexes). I learned over the course of the project that there were a number of issues associated with having a two-part foreign key for the membership.

First, while having an average index length of 4 digits is nice, it's completely unnecessary. I will not see any benefits to having keys that small, compared to having each member utilize only a single foreign key (as opposed to two). Additionally, and improvements that I could see, would be consumed by the cost of searching an additional column of data.

Second, as implemented, I cannot take advantage of any cascading updated or deletion features of the database. While it is the intention that the software would typically retain data in the case of user-deletion or chapter-deletion, cleaning out old data post-archiving would be much simpler without having to perform the operations by hand.

In the future, I would choose to collapse the organization and chapter foreign keys into just a chapter foreign key. The chapters would then hold an organization identifier, allowing cascaded deletion at both levels.

Member-User Distinction

A question that I have received a couple times is why I chose to distinguish between users and members. There are a couple of reasons. The first is that I can reduce the amount of data stored by not storing login-related information in the members table. Not every member will sign up for the service, so combining the tables would create a large amount of unnecessary data.

In addition to storage benefits, it allows me to store login information on completely separate servers from application information. This means that if the application servers are ever compromised, attackers do not have access to the application information, and visa-versa.

GLOSSARY

Admin Rights/Privileges:	A user is considered to have admin privileges is he/she is able to interact with another member, or modify another member's information.
Chapter:	A chapter is a local instance of an organization. For a fraternity, this would be like Michigan Delta, which is the designation for the instance of Phi Delta Theta located at Kettering University, or Troop 532, which is the designation of the local Boy Scout instance in North Vernon, IN.
Member:	A member is just a record in the members table, which corresponds to a person who was once directly associated with the chapter. A member is non-functional, and the presence of a member record does not indicate the presence of a corresponding user record. The member records contain personal and contact information, but no account information.
Organization:	An organization is the national or regional body, of which all members and chapters are a part. An example would be Phi Delta Theta®, or The Boy Scouts of America®.

Power-user:	A user with an above average understanding of an application, who may be inclined to take advantage of additional functionality that an average user may not need.
Software Development Life Cycle (SDLC):	A standard model representing the stages of application development. The five stages are as follows: • Requirement Analysis • Design • Implementation • Testing • Evolution The fifth stage loops back into Requirement Analysis, representing the ongoing growth of an application or product, and the need to continuously examine how the application or product is used, and what can be improved.
User:	A User is an individual who interacts with the application. By design, any user must have a corresponding member record in the chapter of the organization in which he/she participates. The user records contain only the account information and any information needed to tie the account to a specific member record.