6-11-2020

ClubGrove

Travis Le

Lyman Shen

SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 11, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

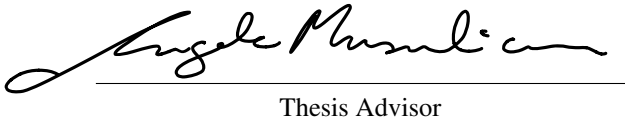
Travis Le
Lyman Shen

ENTITLED

ClubGrove

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



Thesis Advisor

N. Ling

Department Chair

ClubGrove

by

Travis Le
Lyman Shen

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 11, 2020

ClubGrove

Travis Le
Lyman Shen

Department of Computer Science and Engineering
Santa Clara University
June 11, 2020

ABSTRACT

The current Santa Clara University Club system does not provide the necessary information in a relevant and concise fashion. As a solution, we built a system that caters towards students who are looking for clubs. In addition, the system also caters towards club leaders who want to increase their clubs' visibility. Our system allows students to look up clubs, search club events, and look at interested clubs as well as previously joined clubs. The goal of this project is to provide an easy way for students to be involved in more clubs and allow clubs to gain members without spending much financial or temporal resources.

Table of Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Solution . . . 1

2 Requirements **2**
- 2.1 Functional Requirements . . . 2
  - 2.1.1 Critical . . . 2
  - 2.1.2 Recommended . . . 3
  - 2.1.3 Suggested . . . 3
- 2.2 Non-functional Requirements . . . 3
  - 2.2.1 Critical . . . 3
  - 2.2.2 Recommended . . . 3
  - 2.2.3 Suggested . . . 4
- 2.3 Exceptions . . . 4

3 Use Cases **5**
- 3.1 Use Case Diagram . . . 5
- 3.2 Actions . . . 6
  - 3.2.1 Add/Delete Clubs . . . 6
  - 3.2.2 Search Clubs . . . 6
  - 3.2.3 View Clubs . . . 6
  - 3.2.4 Manage Club Page . . . 6
  - 3.2.5 Accept/Decline Membership for Clubs . . . 7
  - 3.2.6 Request Membership for Clubs . . . 7

4 Activity Diagram **8**
- 4.1 Introduction . . . 8
- 4.2 Admin User . . . 9
- 4.3 Guest User . . . 10
- 4.4 Student User . . . 11
- 4.5 Club Leader User . . . 12

5 Technologies Used **13**
- 5.1 Vue.js . . . 13
- 5.2 Bootstrap . . . 13
- 5.3 Flask . . . 13
- 5.4 PostgreSql . . . 13
- 5.5 Git . . . 13

6 Design Rationale **15**
- 6.1 Technology . . . 15
- 6.2 User Interface . . . 15

7 Architectural Diagram **16**
7.1 Introduction . . . 16

8 Description of System Implemented **17**
8.1 Introduction . . . 17
8.2 General Pages and Features . . . 17
8.2.1 Home Page, About Page, and The Navigation Bar . . . 17
8.2.2 Club Search and Event Search . . . 17
8.2.3 Club Page . . . 18
8.2.4 Club Event Page . . . 19
8.3 User Limited Systems . . . 20
8.3.1 User Page . . . 20
8.3.2 Manage Club Page . . . 21
8.3.3 Manage Club Events Page . . . 22
8.3.4 Admin Page . . . 23

9 Difficulties Encountered **24**
9.1 Introduction . . . 24

10 Test Plan **26**
10.1 Introduction . . . 26
10.2 Testing process . . . 26

11 Development Timeline **27**
11.1 Development Timeline . . . 27

12 Ethical Concerns **29**
12.1 Why build this application . . . 29
12.2 The ethics required in our project . . . 29
12.3 Implication of application . . . 29

13 Conclusion **30**
13.1 Suggested Changes . . . 30
13.2 Lessons Learned . . . 30

Appendices **31**

A Installation guide **32**
A.1 Start Back-end locally . . . 32
A.2 Assigning a Database . . . 33
A.3 Generating the front-end locally . . . 33
A.4 Apply Front-end to Back-end(optional) . . . 33

B User Manual **34**
B.1 Start-up User Guide . . . 34
B.2 Guest Guide . . . 34
B.3 Student Guide . . . 34
B.4 Club Leader Guide . . . 35
B.5 Admin Guide . . . 35

List of Figures

3.1 Use Case Diagram 5

4.1 Admin Activity Diagram 9
4.2 Guest Activity Diagram 10
4.3 Student Activity Diagram 11
4.4 Club Leader Activity Diagram 12

7.1 Homepage 16

8.1 Home Page 18
8.2 About Page 18
8.3 Search Clubs 19
8.4 Search Events 19
8.5 Club Page: Guest View 20
8.6 Club Page: Student View 20
8.7 Club Page: Leader View 21
8.8 Club Events Page 21
8.9 User Page 22
8.10 Manage Club Page 22
8.11 Manage Club Events Page 23
8.12 Admin Page 23

11.1 Development Timeline, 1st Half 27
11.2 Development Timeline, 2nd Half 28

List of Tables

9.1 Risk Analysis 24
9.1 Risk Analysis 25

Chapter 1

Introduction

1.1 Motivation

Students have a difficult time signing up for a club after the Student Involvement Fair. The student involvement fair happens so suddenly that students do not have enough time to decide on an organization. In addition to students, clubs themselves have a hard time bolstering their numbers after the fair. The difficulty of joining a club along with factors such as lack of motivation or conflicting schedules lowers club membership. Providing an accessible, centralized solution will encourage students to sign up for clubs after adjusting to school life.

Current solutions are limited. These include the student organization directory, the event calendar, and regular emails that the school sends out about the club. The directory does not provide enough information for a student to make a decision about joining an organization. The event calendar does not have personalized event dates and is cluttered. The generic emails about clubs have to be selective because information about a club has to be condensed.

1.2 Solution

Our solution combines the strengths of all these solutions by centralizing the information to one website. Students see a list of clubs, publicized club events, and are able to request to sign up for clubs. Club officers have the ability to accept member requests and manage events for their club. Admins can maintain the service by managing, creating, and removing groups.

We developed a website that will streamline the process of joining clubs and make it easier for smaller clubs to market to students outside of the student involvement fair. The user interface is easy to use so guests and students can navigate the website and perform actions quickly. Along with viewing a list of clubs, students and guests can search for clubs based on category of the club, keywords, and the name of the club.

After students log in, they can save the club in an interested list or request to join the club. When club officers log in, they will be able to accept join requests by those students. Additionally, the club officers can show basic information about the club on the website. The information can include a detailed description of the club, regular meeting times, and events that are coming soon.

Chapter 2

Requirements

2.1 Functional Requirements

2.1.1 Critical

- Can be run on a web browser.
- Show a list of clubs.
- Show information about a club
- Search for a club
- Allow keywords to be assigned to clubs
- Search for events
- Show information about an event
- Sign up for a club
- Request to join club
- Have a sign in system
- Alter functionality depending on whether a user is signed in or not
- Enable club leaders to publish events
- Enable club leaders to change events
- Enable club leaders to accept or revoke requests
- Allow admins to publish clubs or delete clubs
- Allow for multiple club leaders

2.1.2 Recommended

- Allow club leaders to have specified roles
- Alter view depending on whether a user is signed in or not
- Saving or removing interested clubs
- Allow for multiple admin
- Allow removal of club member from club upon request of a club leader or admin
- Allow deletion of user upon the request of admin or other reasons such as a student no longer being enrolled

2.1.3 Suggested

- Uses SCU color palette
- Remember the user's username from last sign in on the same system
- Allow user to stay signed in upon request or for a certain amount of time when the user is not using the application
- User can sign in on multiple browsers.
- Recommend users for clubs based on interests and current club memberships
- Allow a method of communication with club leaders via the system

2.2 Non-functional Requirements

2.2.1 Critical

- Web site is organized
- The system will have good usability.
- Web pages have good user interface
- Web site good user experience
- Can be reusable

2.2.2 Recommended

- The system is accessible to as many users as possible by following Web Content Accessibility Guidelines
- The system is secure.
- The system is fast.
- The system is robust.

2.2.3 Suggested

- Support mobile functionality with good interface and experience

2.3 Exceptions

- The website is able to run on Firefox and Chrome on macOS, Windows, and Linux.
- The system will run on a desktop or laptop.
- The system is a web application.

Chapter 3

Use Cases

3.1 Use Case Diagram

Guests are anybody who uses the system. Students are guests who log in to the system so they can request to join clubs. Club leaders are students who are leaders of a club. Club leaders can manage their club by accepting requests from students and adding or removing club events. Admins will keep the service running by managing the clubs.

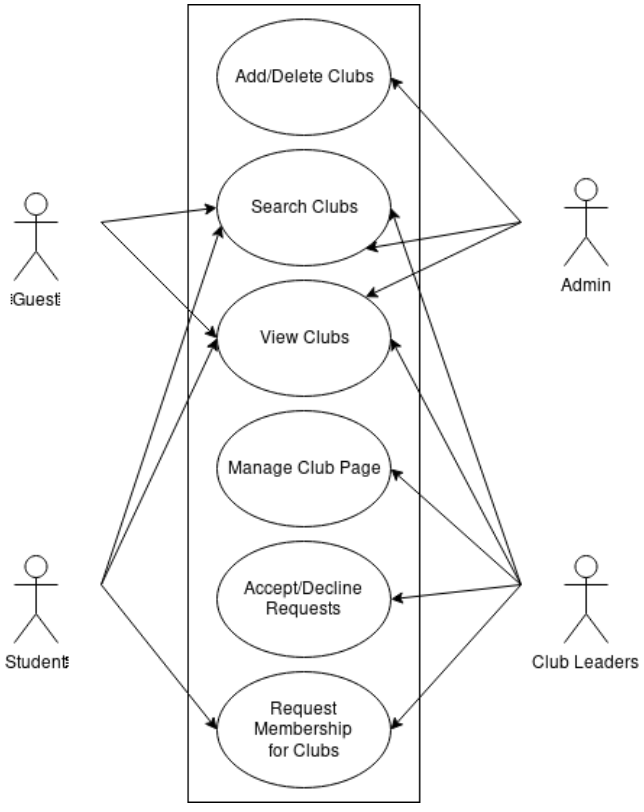


Figure 3.1: Use Case Diagram

3.2 Actions

3.2.1 Add/Delete Clubs

- Description: Add or remove clubs from the service
- Actors: Admin
- Pre-condition: Club name not taken for adding clubs; Club exits for deleting clubs
- Post-condition: Club is added or removed from the list of clubs
- Exception: None

3.2.2 Search Clubs

- Description: Search for clubs in the service
- Actors: Guest, Students, Club leaders, Admin
- Pre-condition: None
- Post-condition: Clubs found are shown, or none if none is found
- Exception: None

3.2.3 View Clubs

- Description: View detailed descriptions about clubs
- Actors: Guest, Students, Club Leaders, Admin
- Pre-condition: The club exists
- Post-condition: Club information is displayed
- Exception: None

3.2.4 Manage Club Page

- Description: Add or remove club events
- Actors: Club leaders
- Pre-condition: The club exists
- Post-condition: Relevant information is recorded
- Exception: None

3.2.5 Accept/Decline Membership for Clubs

- Description: Accept or decline requests to join a club
- Actors: Club leaders
- Pre-condition: None
- Post-condition: The request to join a club is removed. If the request is accepted, the student who made the join request is stored as part of the club members.
- Exception: None

3.2.6 Request Membership for Clubs

- Description: Request to join a club
- Actors: Student, Club Leaders
- Pre-condition: The club exists
- Post-condition: Request to join membership is stored in the system
- Exception: None

Chapter 4

Activity Diagram

4.1 Introduction

The diagrams below show how our users can interact with our system. The activities will change depending which type of user they are. With the exception of the admin, users will have some shared functionality.

4.2 Admin User

Figure 4.1 shows an admin's interaction with the system.

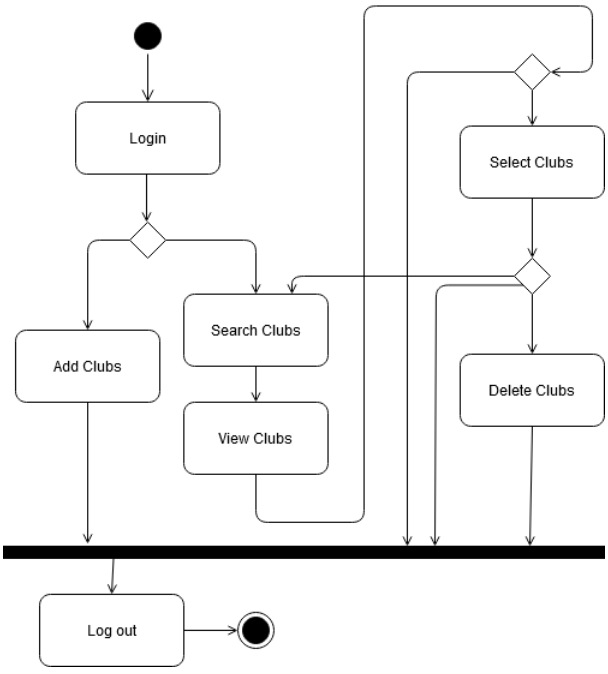


Figure 4.1: Admin Activity Diagram

4.3 Guest User

Figure 4.2 shows a guest's interaction with the system.

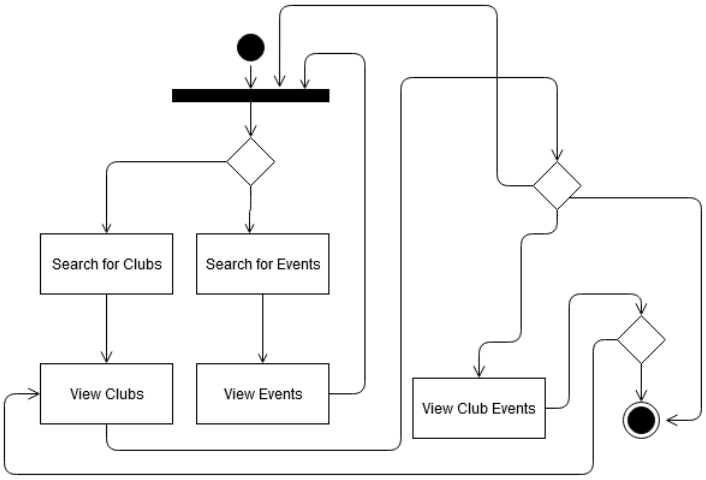


Figure 4.2: Guest Activity Diagram

4.4 Student User

Figure 4.3 shows a student's interaction with the system.

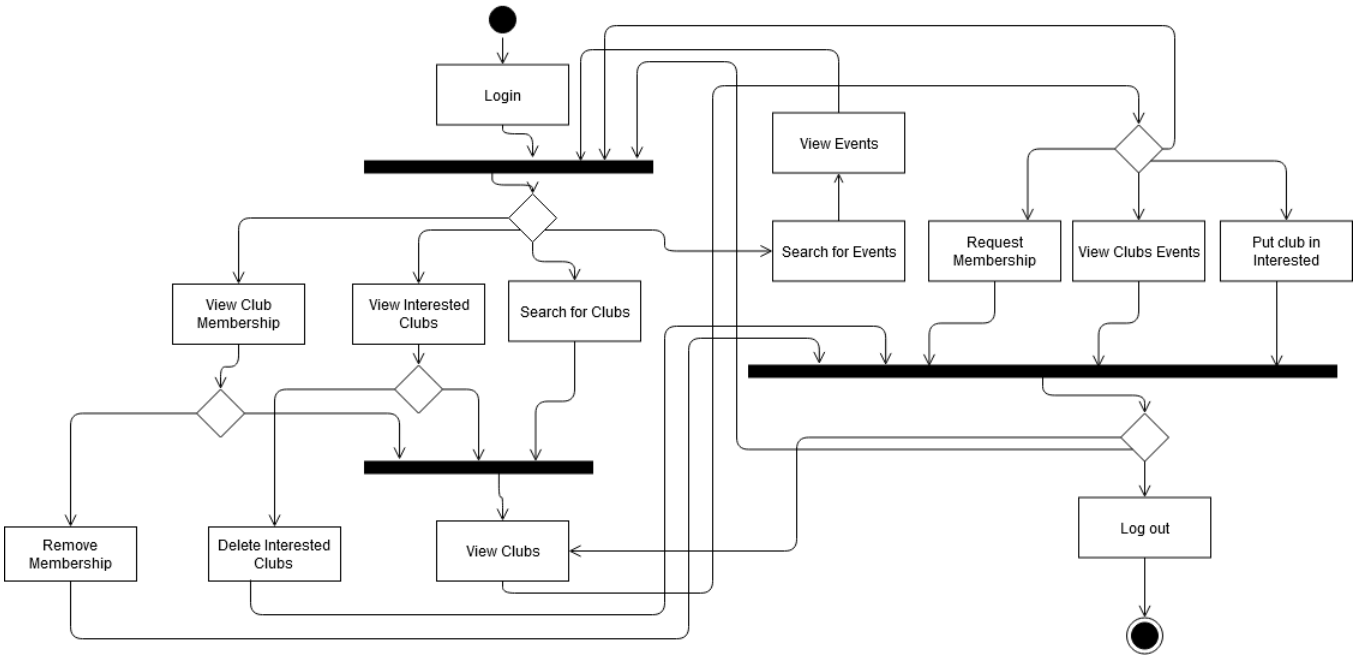


Figure 4.3: Student Activity Diagram

4.5 Club Leader User

Figure 4.4 shows a club leader's interaction with the system.

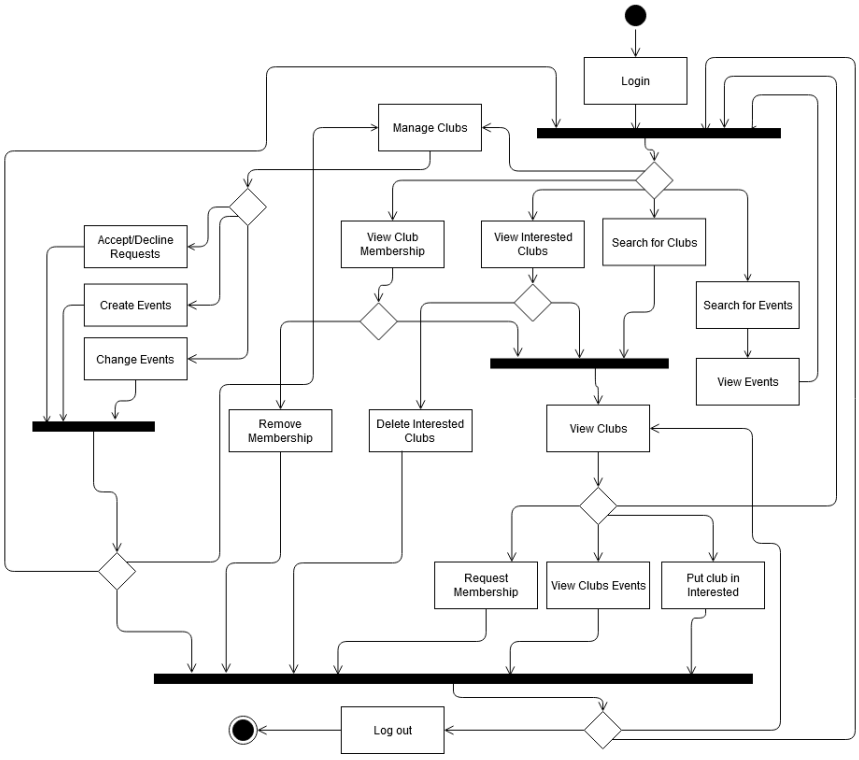


Figure 4.4: Club Leader Activity Diagram

Chapter 5

Technologies Used

5.1 Vue.js

We chose Vue.js as our front end because it is simple, has extensive documentation, and is easy to use. It is recommended by many front-end developers. As a smaller package compared to other popular front-end systems like React, using Vue.js allows web sites to load faster.

5.2 Bootstrap

We want to style our web pages to make it look more user friendly. This is typically done using CSS as part of any front end including Vue.js, but Bootstrap simplifies the implementation. With Bootstrap styling, web pages become easy to style. In addition, Bootstrap make decorating for mobile devices easier than just applying CSS.

5.3 Flask

We chose Flask as our back end because it supports using multiple database systems and allows flexibility for extensions that makes implementation simpler. It works well with Vue.js, our front-end choice, and is easy to use.

5.4 PostgreSql

We originally planned to use Oracle database based on our previous experience. During the implementation, we found that Oracle did not integrate easily with Flask server, so we switched to using the free and open source PostgreSql database. Based on research, we learned PostgreSql can securely and efficiently manage data, and supported the SQL language that we knew from previous experience. Since PostgreSql also integrated easily with our Flask server, we chose to use it.

5.5 Git

Git made developing separately easier. We were able to develop the application at different points of time and did not need to be in the same location to develop code. We chose to use Git because it was familiar to us. We had used Git over the previous summer, and we were able to develop our skills over the school year

to become proficient in it. It was also linked to the website GitHub which provided us a way to upload code and see each other changes. We were able to merge our code together and also check if our code conflicted with each other upon making changes. Overall, Git was a necessary part of our technologies to enable fast development and easy communication.

Chapter 6

Design Rationale

6.1 Technology

We implemented ClubGrove as a website so it could be more readily available in different environments. This allows ClubGrove to be accessed by more students by not limiting them to an operating system platform and be less likely to have compatibility issues.

As a web app that requires much data, a database system like PostgresSql is used to better manage the data. A back end is needed to retrieve information from the database and deliver content, and we chose Flask because it can have many features and is easy to use.

A front-end is needed to style different web pages and make them interact well with each other in the web site. We chose to use Vue.js as a way to accomplish this.

To make styling the web pages easier, and we will use a combination of CSS and the Bootstrap plugin.

One notable thing is that as a website, HTML will be used heavily as part of the Vue.js front-end and the Flask back-end as HTML is the backbone of any web site.

6.2 User Interface

We want the web site to be easy to use. Therefore, searching for clubs, viewing clubs, and other frequent actions should be quick, easy, and intuitive. Less frequent actions like requesting club membership and accepting or declining those requests for club leaders should still be easy to access.

Thus, we kept our design simple to emphasize its usability and functionality. A clean, uncluttered UI will make navigation much easier for new, unfamiliar users.

Chapter 7

Architectural Diagram

7.1 Introduction

This is the general structure of our web system. Vue.js, our front-end framework, represent the general view. Depending on what view a client has, the controller will fetch data from our model. Flask, our controller, will pull data from our database to send to our view. The database we are using to represent our model is Oracle SQL. Afterwards, the view will show a particular web page to the user based on the what type of user they are and the data it got from the controller.

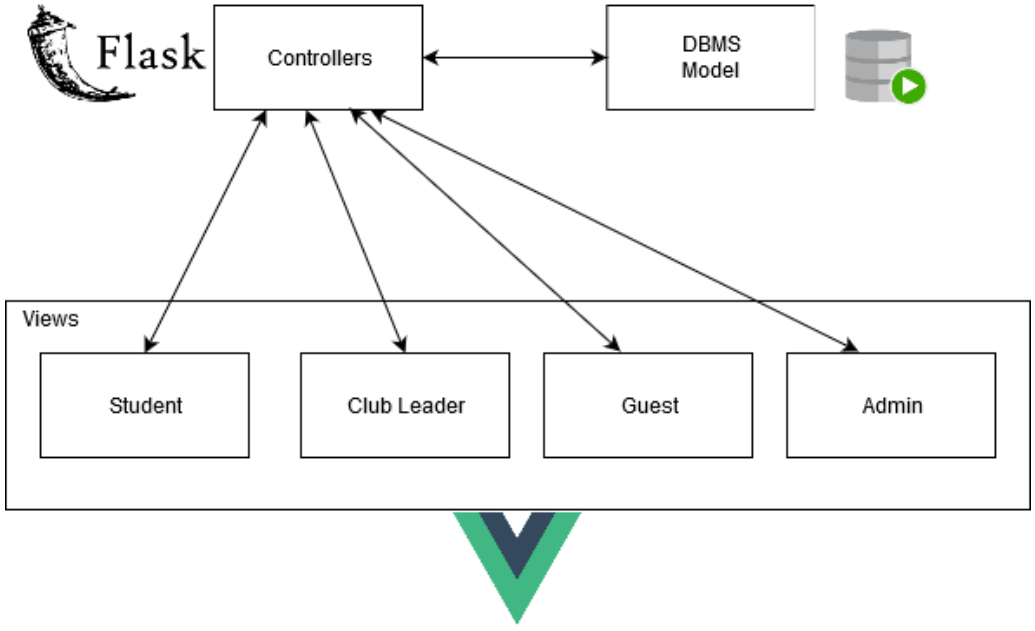


Figure 7.1: Homepage

Chapter 8

Description of System Implemented

8.1 Introduction

We developed a web application that allows different privileges depending on the user's view. As a guest, they can search for clubs and events. Along with the actions of a guest, a student can join clubs and manage their list of memberships, interested clubs, and pending requests. Club leaders can also do all the actions a student can perform along with a few more options. Club leaders can manage clubs and modify, create, or delete their club events. Admins have the option to delete clubs or add clubs.

8.2 General Pages and Features

There are many pages in this website that are accessible to the public. Even though there are some features of various pages that can only be accessed by certain users, the pages can be accessed by everyone.

8.2.1 Home Page, About Page, and The Navigation Bar

The Home Page is the first page the user will see on the website. The page is a description of what the application aims to achieve and how to use the navigation bar. Figure 8.1 shows how the page looks like.

The About Page is the page which lists details about the purpose of this project. In addition, it includes the names of the people who worked on this project. The About Page is shown in figure 8.2.

The navigation bar will always be shown regardless of which page a user is on. The navigation bar contains five options, four of them either take the user to the home page, the club search page, the event search page, or the about page. The fifth option allows the user to sign into the system. If a user is logged in, the option will become a drop down in which two more options appear. The user page option directs the user to the user page while the sign out button signs the person out. If the user is an admin, admin options will be in that drop down list. The admin option takes the user to the admin page.

8.2.2 Club Search and Event Search

The Club Search and Event Search are the two main options a user can select in the navigation bar. Club search allows the user to search within a specific category if the club has a category assigned to it. In addition, the user can query for a title of a club in which the search system will only find options that match the query. The sort options for the search include ascending, descending or unsorted. Clicking the ”go to club page” button allows the user to be directed to the group's club page. This is shown in Figure 8.3.

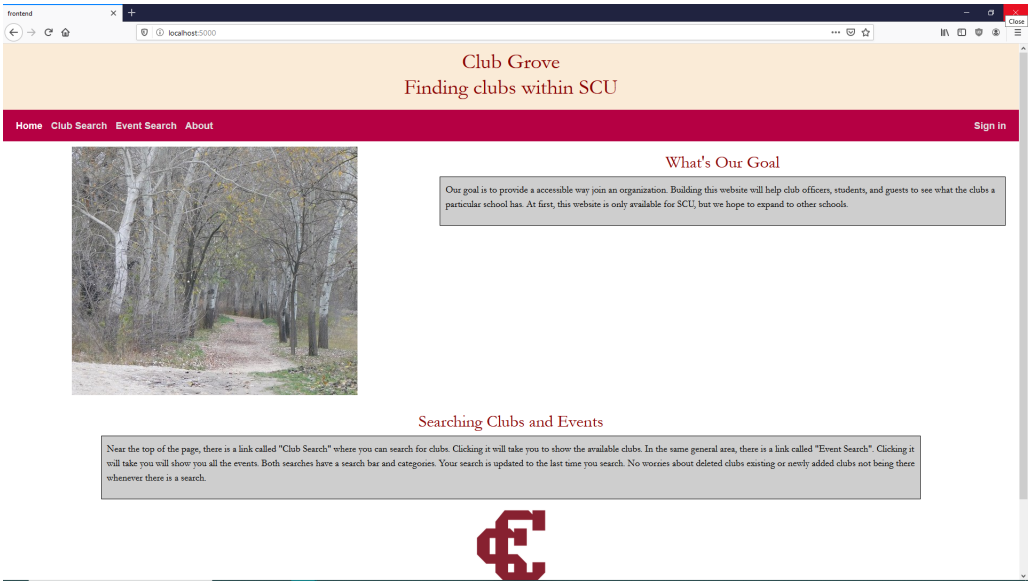


Figure 8.1: Home Page

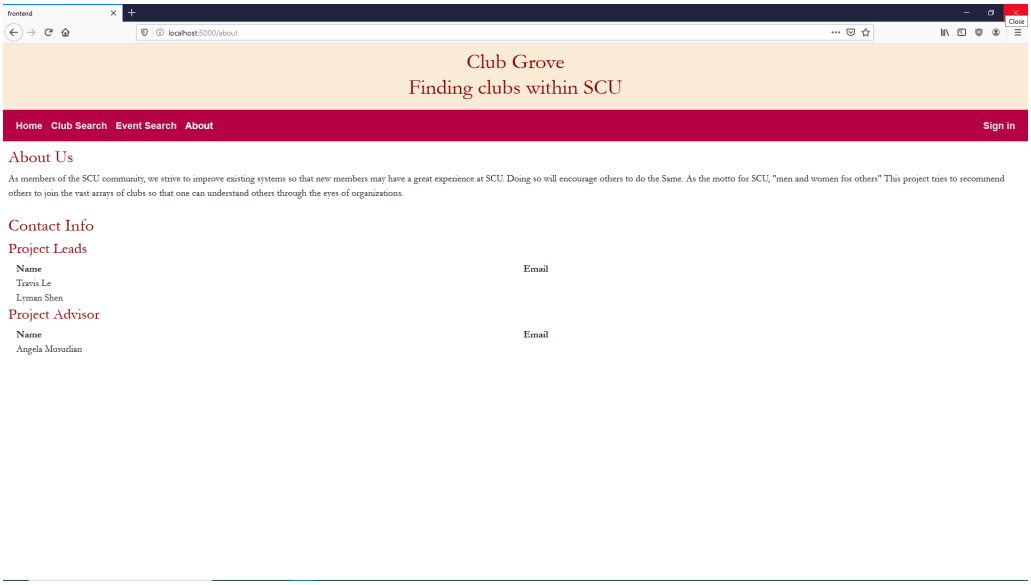


Figure 8.2: About Page

The Event Search page has some similarities to the club search page. The search function is almost entirely the same as the club search except there is no club categories. The Event Search page does not have an event page dedicated to individual events. The page can be found as Figure 8.4.

8.2.3 Club Page

The Club Page is where the club's overall details are shown. In the middle of the page, there is a picture that represents the club. In addition, the left side of the page shows the description of the club. On the bottom, up to three recent events are shown. Users that want to see all events can go the Club Event Page. Depending on which view the user is, the right side will be different. If the user is a guest, nothing will be shown to the

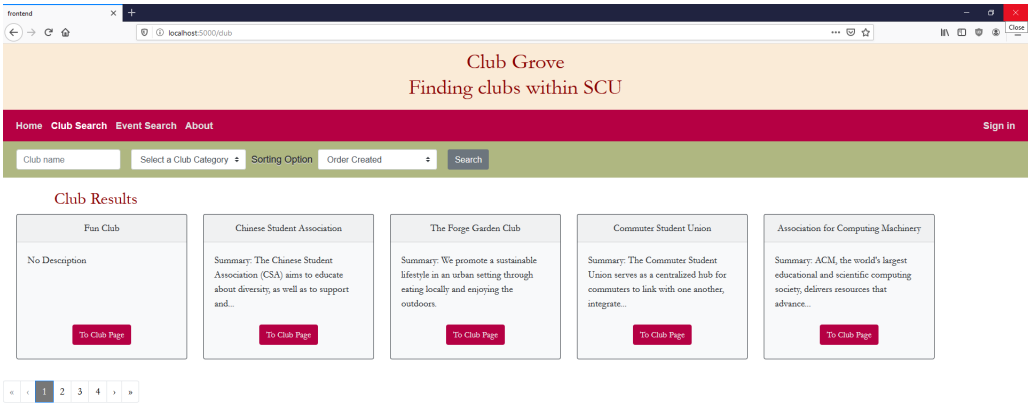


Figure 8.3: Search Clubs

Figure 8.4: Search Events

right. If the user is a student or a club leader that is not the leader of the particular club, then the right side will show buttons to join or put in the interested list. If the user is a club leader that the leader of the particular club, it will show a button which redirects them to club leader options. All these example can be shown in Figures 8.5, 8.6, and 8.7.

8.2.4 Club Event Page

The Club Event Page shows a list of club events that a particular club has. Unlike the Club Page which shows a limited number of clubs, the Club Event Pages show all the events that the club has. This page is shown in Figure 8.8.

Figure 8.5: Club Page: Guest View

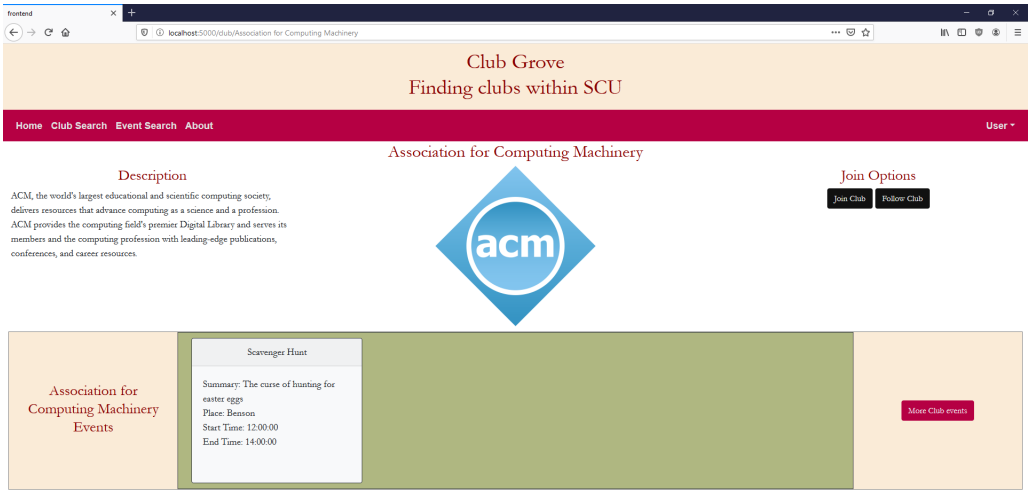


Figure 8.6: Club Page: Student View

8.3 User Limited Systems

These pages either require permissions to access or an account. In other words, these are pages in which a guest cannot access without logging in as a student, guest, or club leader.

8.3.1 User Page

The User Page allows students to see their interested list, pending requests, and memberships. The search options functions similar to the Club Search Page search bar. The search is limited to the interested list, pending requests, and membership; however, the user can choose to limit the search to only memberships.

Figure 8.7: Club Page: Leader View

Figure 8.8: Club Events Page

The left side shows the options in which a user can manage all their memberships, requests, and interests. They can choose to remove the clubs from the lists or transfer the interested club to request membership list. Figure 8.9 is an example of this.

8.3.2 Manage Club Page

The Manage Club Page allows club leaders to manage requests. In addition, there is a link to the Create Events Page for the club leader to create events. For managing requests, they can choose to accept club request or decline club requests. Figure 8.10 is one such example.

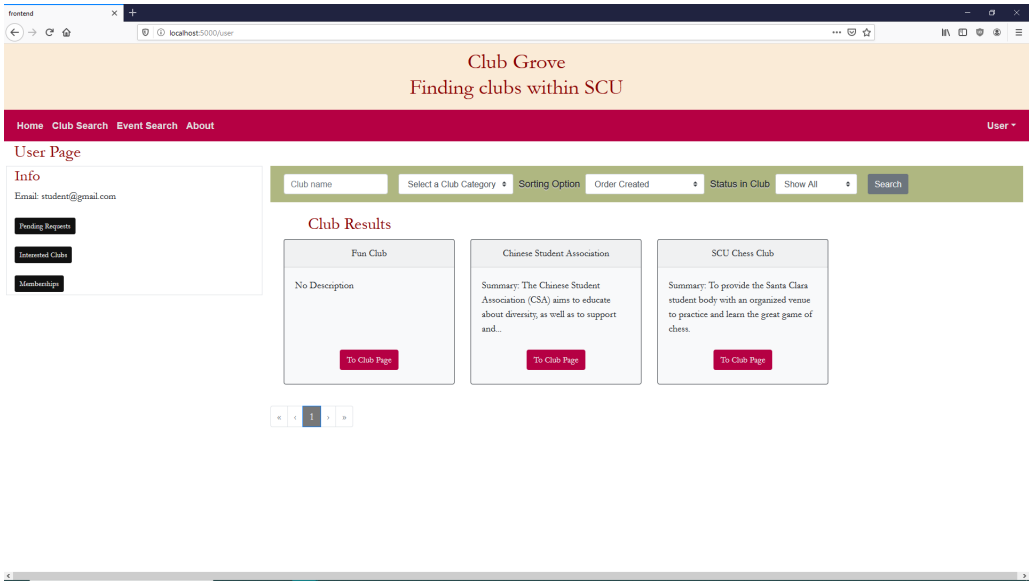


Figure 8.9: User Page

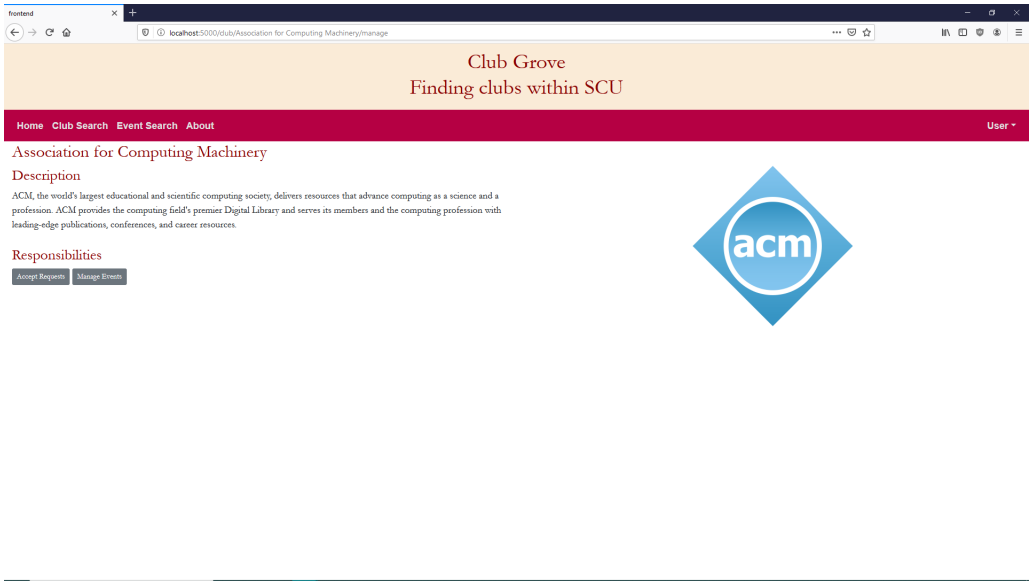


Figure 8.10: Manage Club Page

8.3.3 Manage Club Events Page

In this page, club leaders can create events. To add an event the leader has to click the add event button. From there, a module will pop up prompting for information. After the club leader has entered and submit the information, the event will be added to the list of events and their club events list. Figure 8.11 shows what it looks like.

Figure 8.11: Manage Club Events Page

8.3.4 Admin Page

The Admin Page can only be accessed by the admin. The admin has the option to delete or create clubs. To delete a club, the admin will just press the button ”delete club”, which will be associated with the club name. To add a club, the admin will press the ”Add Club” button which displays a popup. The popup shows the necessary information to add a club. After the admin submits the club information, the club will be added to the system. This is shown in figure 8.12.

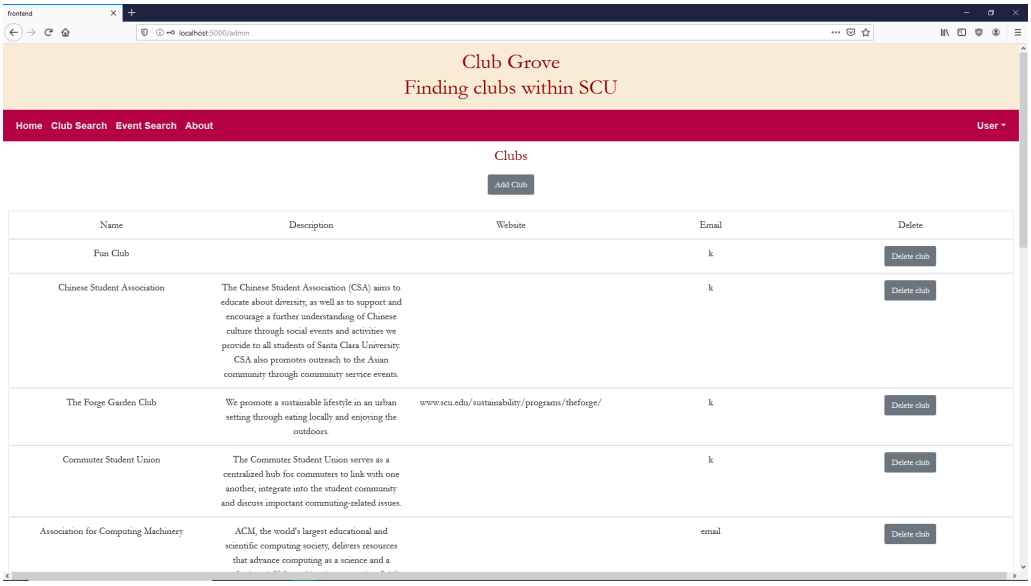


Figure 8.12: Admin Page

Chapter 9

Difficulties Encountered

9.1 Introduction

The table below indicates the difficulties we encountered to our project. The table is sorted by impact in descending order. Impact is the product of probability and severity. Probability is the likelihood that the risk could happen, and severity is how damaging that risk could be to our project.

Table 9.1: Risk Analysis

Risk	Consequences	Probability	Severity	Impact	Mitigation Strategies
Bugs	System not does not work; May increase production time	0.99	5	4.95	Test early, create testing programs, purposely add bugs
Incompatible Software	Product is delayed, and must use other software	0.8	6	4.8	Check compatibility with other technologies, test before using technologies
Group member gets sick	More work for other group members	0.6	4	2.4	Stay healthy, work while sick if possible
Incomplete System	Product is delayed or not done, lacking some or all required functionalities	0.2	10	2	Manage time well and limit scope creep. Work on most vital components first.
Major pandemic occurs	Product significantly harder to work on, less communication between group	0.1	10	1	Communicate via online format, Use Git to share code, Schedule meetings online

Table 9.1: Risk Analysis

Risk	Consequences	Probability	Severity	Impact	Mitigation Strategies
Database systems incompatible with environment	Delays in production, testing & maintenance	0.2	3	0.6	Find similarities between database systems; Use abstraction to make conversion to new database easier

Chapter 10

Test Plan

10.1 Introduction

Through the development of this project, we had a plan to test for bugs, security issues and other types of problems. This section lists what we did during the testing phase of our project.

10.2 Testing process

We had to test our website individually through unit and functional testing. This means each page was to be tested on its own before being connected to other pages. The following process was done to test the components on its own.

For the web pages that did not require access to our back-end server, we simply loaded the web page in a browser to see if it displayed the correct information in the right format.

For the web pages that needed to access our back-end server, we first tested if the SQL query gave proper results or had intended side effects. Then, we tested our back-end server link to make sure it showed the message that our front end will receive and had the intended side effects. Lastly, the web page was tested to make sure it displays the information in the correct format.

For the components that required the use of Session variables in the server, there was not a systematic way to test. As a result, we ran those components after the necessary parts in both front-end and back-end server was done to see if the actions relating to the session variables gave intended results.

When everything works properly, we tested if the modules work together.

Chapter 11

Development Timeline

11.1 Development Timeline

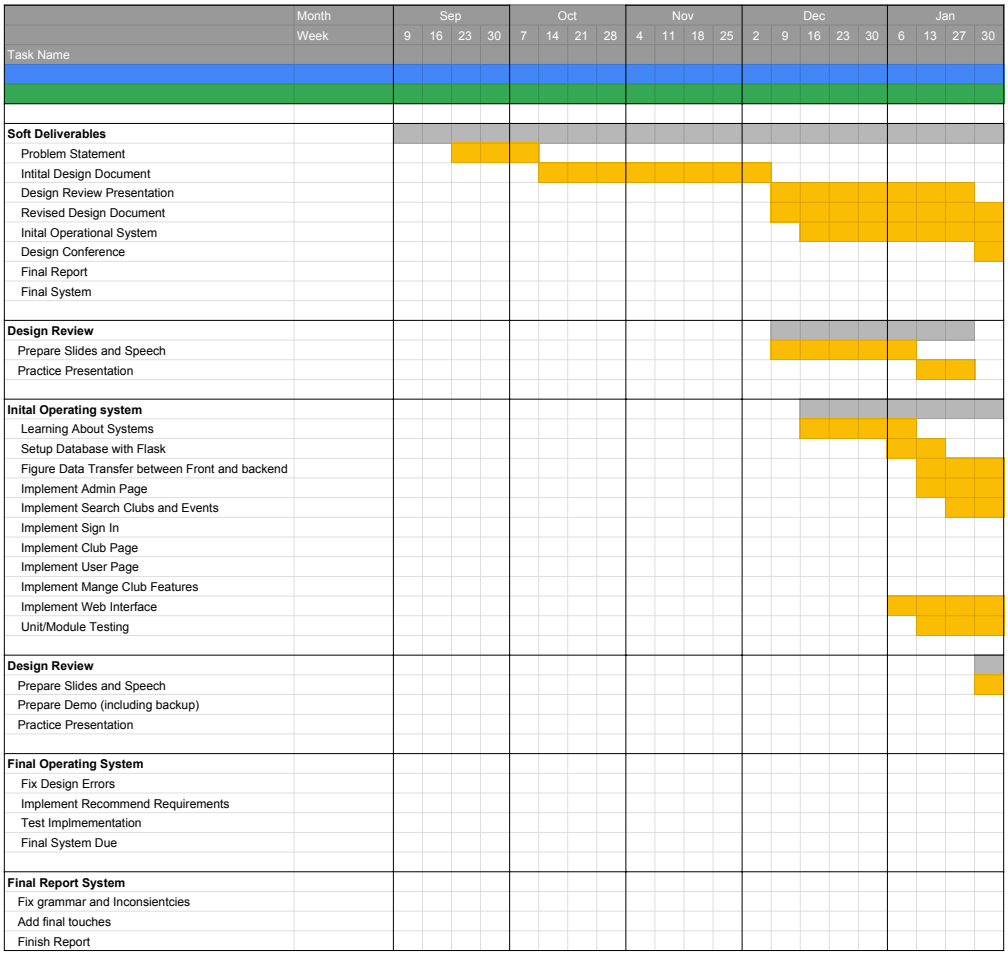


Figure 11.1: Development Timeline, 1st Half

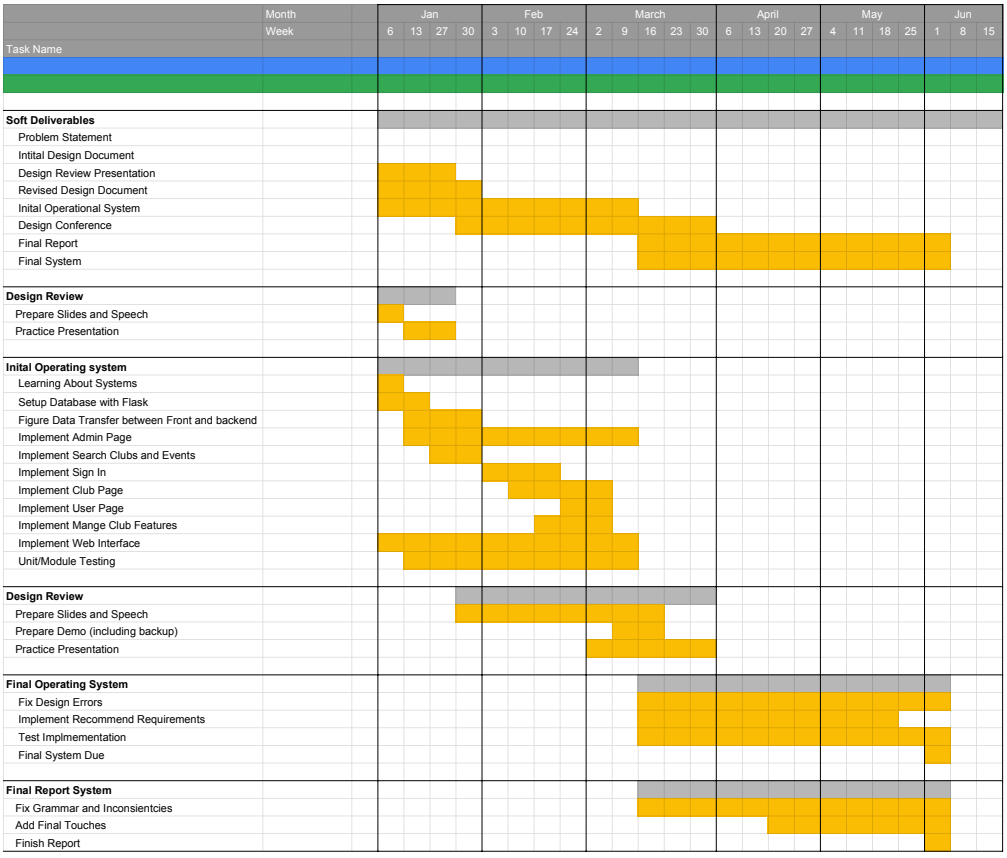


Figure 11.2: Development Timeline, 2nd Half

Chapter 12

Ethical Concerns

Throughout the process of building this system, we strive to keep in mind our ethics. Mainly, we will be focusing on why is ethically good, what ethics are involved in the experience of making this application, and what are the implications our application can have. Most of the ethics are based off of the ACM code of ethics[1].

12.1 Why build this application

The main reason for building this application is to improve the overall experience of all SCU students. According to the ACM's ethical consideration, doing the public good is in line of its ethics[1]. Helping students look for clubs will make students a little more active and overall improve the SCU lifestyle. In addition, the building of this application may encourage others to build systems to improve SCU. Trying to build this program fosters others to improve this application and other areas of the school.

12.2 The ethics required in our project

When working on this project, there were a few things we had to keep in mind. One thing was to accept and provide professional review. To check if our codes worked with each other, we met up every once in a while. This also enabled us to keep a high standard of work and keep track of our responsibilities. The other thing we had to keep in mind while doing this project was maintain a working schedule. According to the ACM code of ethics one must "Manage personnel and resources to enhance the quality of working life"[1]. We tried to manage our time well by dedicating time to work on the project.

12.3 Implication of application

There are few ethical implications that could happen because of the application. The application may expose some user data to the general public. However, the public data gathered by the website may not be damaging to the user. This information can range from emails to what clubs a particular person belongs to. Another concern is that the security of this project is ill managed. This system does have some security features like hashing passwords. The few commands that are left in the program could potentially delete data. However, in the full release, these commands will likely be removed.

Chapter 13

Conclusion

13.1 Suggested Changes

Even though we have completed our requirements, there are several features that we wished to have included. Those include providing the ability for students to sign up as a user, the ability for admins to sign up, being able to store date and time for various club events, allowing events to be recurring as many clubs have events that happens every week, allowing admins and club leaders to put pictures in their club page like webmasters are able to, and allowing admins to assign students as a club leader. There are also several bugs that causes some current functionality to not work that would need to be fixed before the website is ready for public consumption. Additionally, we can always improve upon the user interface and user experience. An example is using the pagination system across the entire web site instead of a select few pages.

13.2 Lessons Learned

Most of the lessons we learned have to do with our plan to implement this system. Even though we had set a development timeline, we could have improved it. If we had planned to finish more of the requirements earlier than we did, we may have been able to implement some of the suggested changes above. Additionally, we had treated the timeline as a guideline instead of a strict deadline for features implemented, which had caused the implementation of the relatively more difficult features to lag behind the timeline. The combination created a situation where we found ourselves barely finishing the minimal requirements on time. If we could do this project again, we would make those changes, and likely would finish more of the suggested changes above.

Appendices

Appendix A

Installation guide

To install this system, you need to have these:

- Python 3.8 or higher
- Git
- PostgreSQL
- Pip
- npm
- vue/cli

A.1 Start Back-end locally

To clone our Git repository, go to this link https://github.com/Lleeshen/ClubGrove and clone the directory to your local system. Alternatively, go to a folder you want to put the git file in and call the command:

git clone https://github.com/Lleeshen/ClubGrove.git

Next, go into the directory ClubGrove by calling these commands:

cd ClubGrove

Afterwards, you need to start a Python environment and installing the necessary libraries. This is done by calling these commands in linux:

python3 −m venv venv
source venv/bin/activate
pip install −r backend/requirements.txt

In Windows, the commands are different. Instead, call these commands:

python −m venv venv
source venv/Scripts/activate
python −m pip install −r backend/requirements.txt

Afterwards, every time you start a terminal you either need to call this command on Linux:

source venv/bin/activate

and call this command on Windows:

source venv/Scripts/activate

To start the python server, call this command:

python start.py

This should start the server. To go to the server, go to http://localhost:5000. The application should exist but there are currently no pages on it.

A.2 Assigning a Database

To assign a database to the system you need to go to the folder labeled ”backend.” Then go to the the ”model” folder and edit the dbfile.ini. This is done by calling the following commands:

cd backend
cd model

In the dbfile.ini, edit the information so it contains your username, password, host, and database password. Now, the application now has a database.

A.3 Generating the front-end locally

While your back end is running, go to the ”frontend” folder. If you are in the base directory, call this command:

cd frontend

Next, call this command to install the necessary modules:

npm install −g @vue/cli @vue/cli−service−global
npm install axios bootstrap−vue vue−router

To start it, call this command:

npm run serve

If that does not work go to the src folder and call this:

vue serve

The front-end should be running on local-host:8080

A.4 Apply Front-end to Back-end(optional)

If you just want everything from the front-end to go to the ”backend” folder do these instructions: First, in the frontend folder, edit the vue.config.js file. The specific edit is to un-comment the line that has outputDir and assetDir. The comment that has assetDir will not allow npm run serve to work properly. However, it will allow npm run build to work properly. After wards call this command:

npm run build

This will allow the back-end to obtain all the information from the front-end.

Appendix B

User Manual

B.1 Start-up User Guide

These are the commands that deal with generating and deleting a database. So far these are used for testing purpose and will be deleted on a full release.

localhost:5000/db/init
localhost:5000/db/add
localhost:5000/db/reset
localhost:5000/db/clear

- Init: generates the tables
- Add: adds the values
- Reset: deletes the tables
- Clear: clear all values from tables

B.2 Guest Guide

1. When you are at the application, you can choose whether to go to the club search page or the event search page.
2. When you arrive at either page, you can search for more items or navigate through the pages. The system only shows a couple of items per page.
3. Assuming that you are at the club search, you can click the red buttons to go to a specific club page. From there, you can go see a list of club events.

B.3 Student Guide

1. First, you have to login in to the system, which will be given to you before you come to the application. After you login in, you will be given the same options as the guest and more.
2. To join a club or put on interested list go to the club page and click on the respective button.

3. To log out or go to the user page, click the user options button and afterwards click the option that suits your needs.

4. To remove a club from interested list or pending requests, click the side buttons that has the requested list and remove your preferred option.

B.4 Club Leader Guide

1. You have the same login process as a student.

2. To manage club info, go to the page you are a leader as and click the manage club options. Doing so will give you options to move pending memberships to memberships and edit/add club events

3. To edit/add club events, go to the manage event page which can be accessed from the club page or the manage club page. Afterwards, click on the button that says add events which will prompt a popup to type in information. After typing the information, just add the event and it will show up as an event.

B.5 Admin Guide

1. You have the same login process as a student and club leader.

2. To manage club information, go to the user options and click admin options. From there you will be able to delete or add clubs.

3. To add clubs, click the add club button, enter the necessary information and press add club.

4. To delete a club, just press the delete club button. However, clubs cannot be deleted if they have events.

Bibliography

[1] Association for Computer Machinery. ACM code of ethics and professional conduct. https://www.acm.org/code-of-ethics, 2018.