(Cougar Nav)

(Team 5)

(Sierra Norris, Shane Wasserman, Javier Gutierrez, Leonardo Beltran-Mariano)

Software Design Specification & Project Delivery Report

Version: (1) **Date: (12/11/2024)**

Table of Contents

1 Introduction

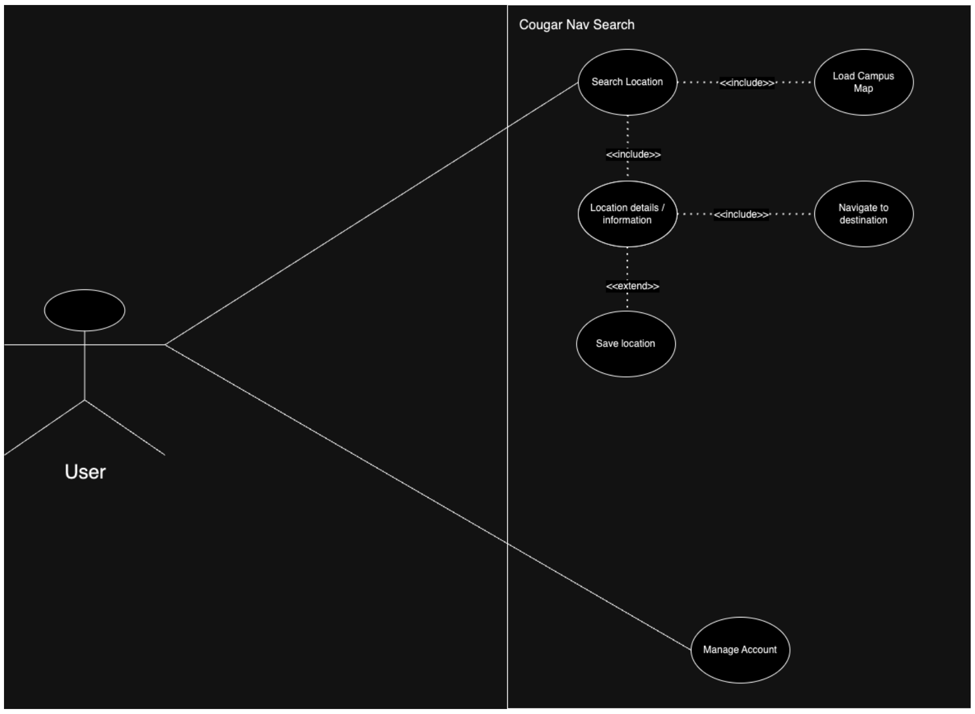
This section provides an overview of the entire design document. This document describes all data, architectural, interface and component-level design for the software.

1.1 Goals and objectives

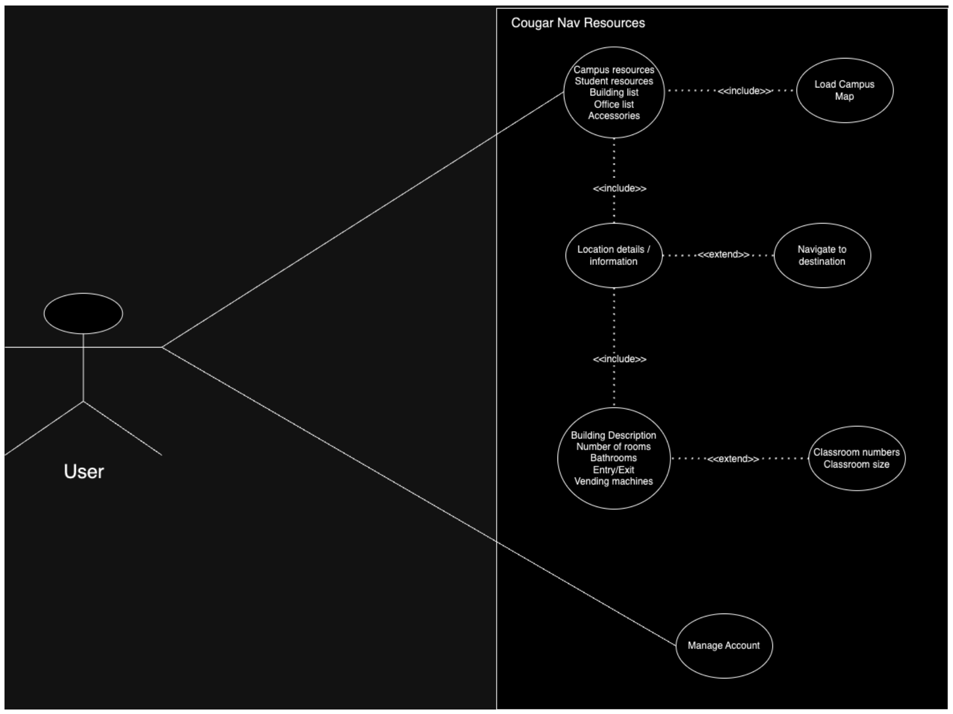
The goals of this software is to efficiently navigate students to desired locations, help achieve time punctuation, time management, eliminate confusion, and highlight helpful resources that students can benefit from. This software will also give information about items around campus and how to navigate to them. This type of software will benefit students with an increase in productivity, relieve stress, increase efficiency, increase confidence, and time management which will help students academically.

1.2 Statement of system scope

Cougar Nav is an advanced mapping software for California State University San Marcos. In this software is an included detailed map of the campus with locations of important features and destinations. These will be locations of certain buildings, classrooms, offices, parking structures, restaurants, vending machines, atms, student resources, dormitories, and other features that are offered on campuses.



Caption



Caption

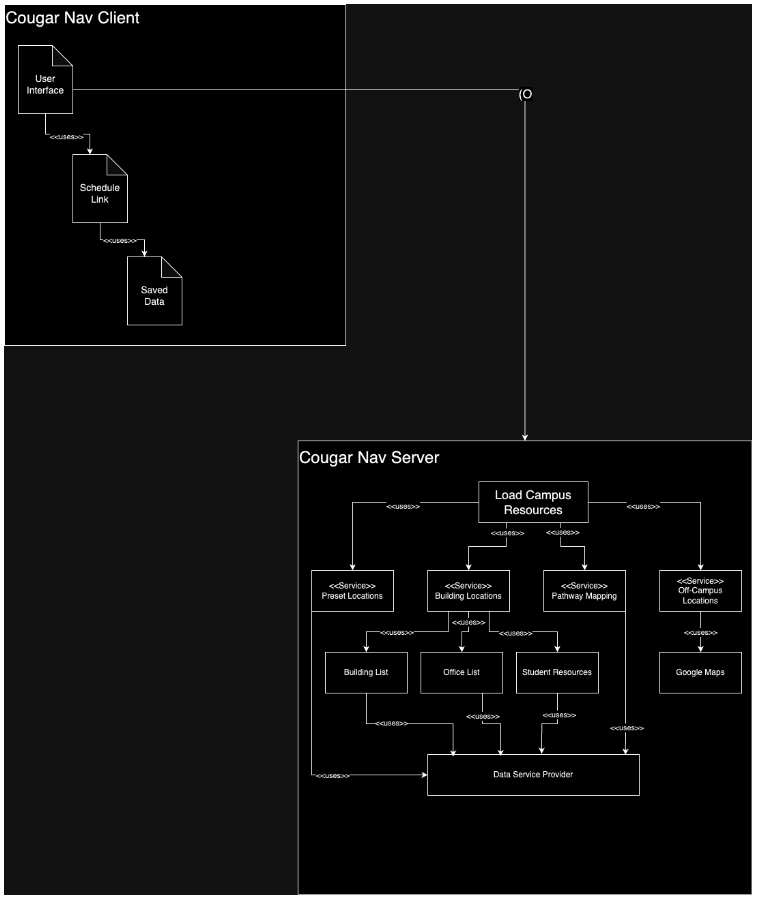
1.3 Reference Material

Google, GitHub, Geeks for Geeks, Open AI, Stack overflow,

2 Architectural design

2.1 System Architecture

The program is structured as a Client-Server architecture design. In the client package contains the users saved data (if user is logged in) and displays the user interface of the map with available resources. In the server package contains the data of the campus. Load campus resources gets all the preset locations, building locations (classrooms, offices, resources), pathway mapping, and off-campus location. The user interface in the client package calls the server and gathers all the data to accurately display whats available on campus. When the user interacts with the user interface, data is collected from the server and displayed back to the user.



Caption

2.2 Design Rational

We chose the Client-Server architecture because it best for the organized services of the software. With all the information about the campus saved in one server, individual users will access the server to get all the information that will be needed to navigate the campus. It can be used from one computer or distributed across a network. A disadvantage to this approach is that the server side can be a single point of failure. If something goes wrong on the server side, then all users are affected. We almost chose a layered architectural design because the server package is layered. The server package uses layers to get the correct information about location depending on what type of resource it is. However processing through too many layers would cause performance issues and was not as easy to implement as Client-Server.

3 Key Functionality design

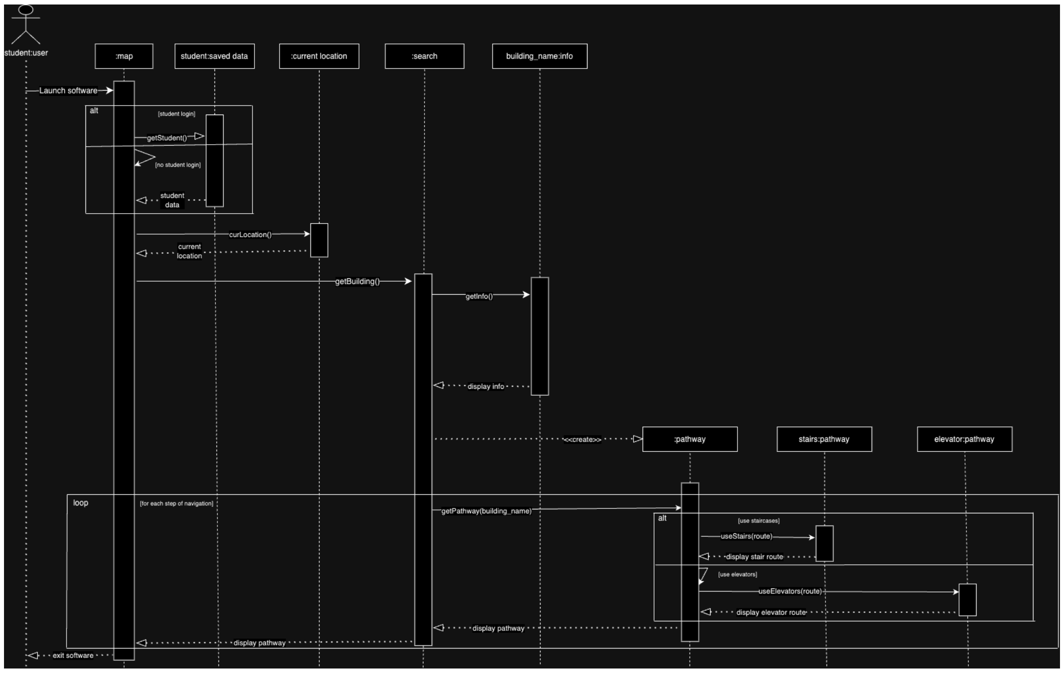
Following the given structure to describe the design of key functionality, **add a brief paragraph to describe the diagrams**.

3.1 [Function 1: Navigation]

3.1.1 [Navigation] Use Cases

[A user searches (or selects from menu) a location/resource to navigate to. Information about the location will be displayed with a directions button. The directions button will display a pathway from users current location to the selected location. The user will have the choice to take a pathway with stairs or an elevator (for users with disabilities).]

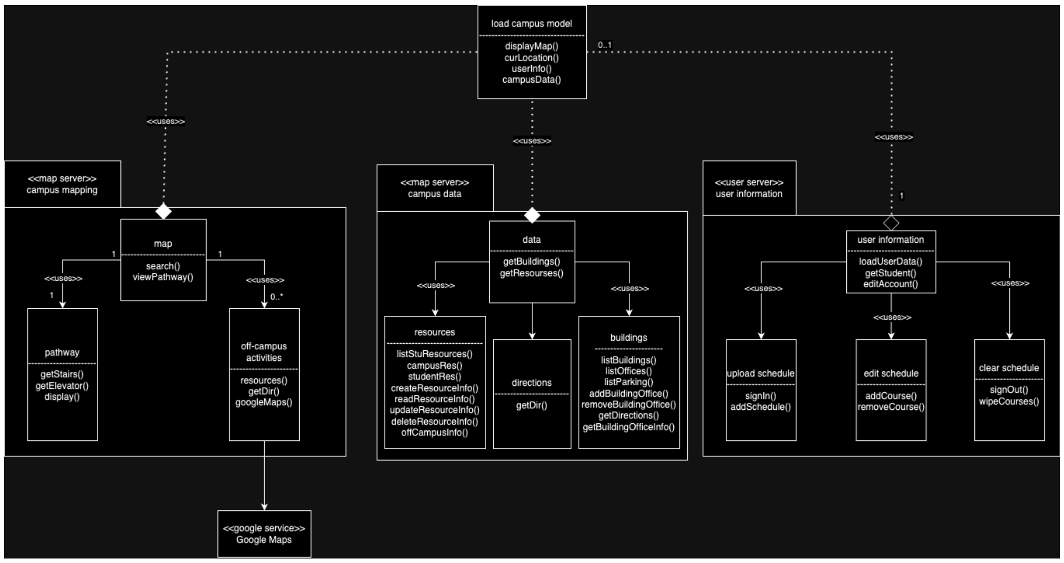
3.1.2 Processing sequence for [Navigation]



Caption

User launches software and the map is displayed. The user searches/selects a location and the building directions are displayed. The directions include a pathway, either stairs or elevator, and the displayed pathway loops to keep the updated route displayed.

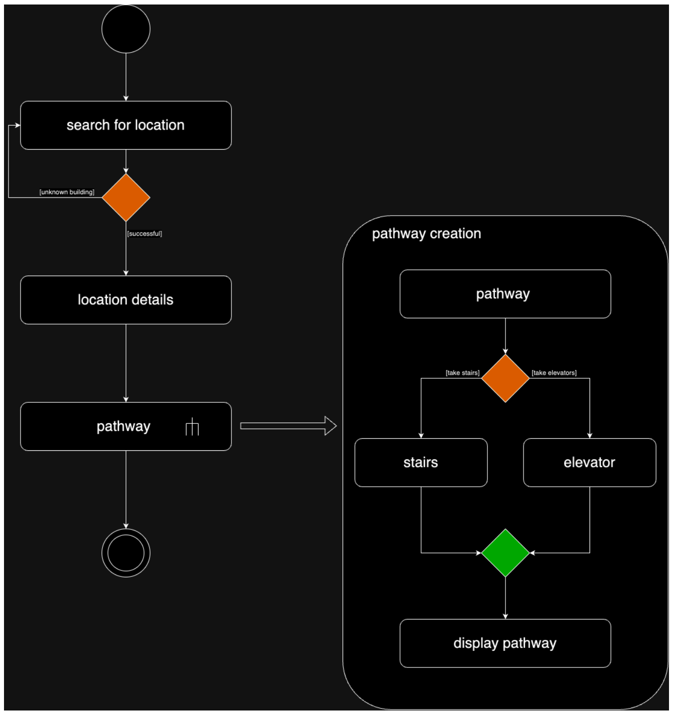
3.1.3 Structural Design for [Navigation]



Caption

The campus model is loaded and the campus map is displayed along with the users current location. To get navigation, campus mapping and campus data is called to get all the related information to the search/selected building.

3.1.4 Key Activities



Caption

User searches for a location, if the location isn't valid the user will have to enter a new location. If the location is valid, location details and information will appear with a directions button. When the directions button is pressed a pathway will appear. The pathway has multiple options for users. They can use stairs or elevators to navigate the campus. The user chooses which pathway they want and the corrected path will appear.

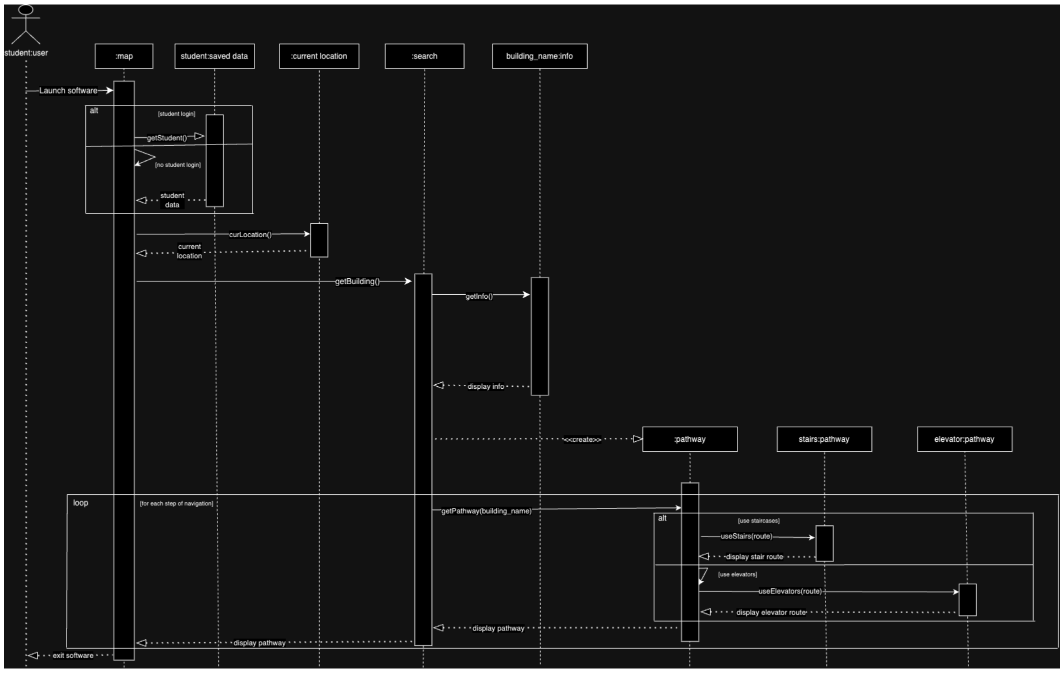
3.1.5 Software Interface to other components

3.2 [Function 2: Information]

3.2.1 [Information] Use Cases

[A user searches (or selects from menu) a location/resource to navigate to. Information about the location will be displayed with a directions button. The information includes phone numbers for help, hours of operation, and other needed information.]

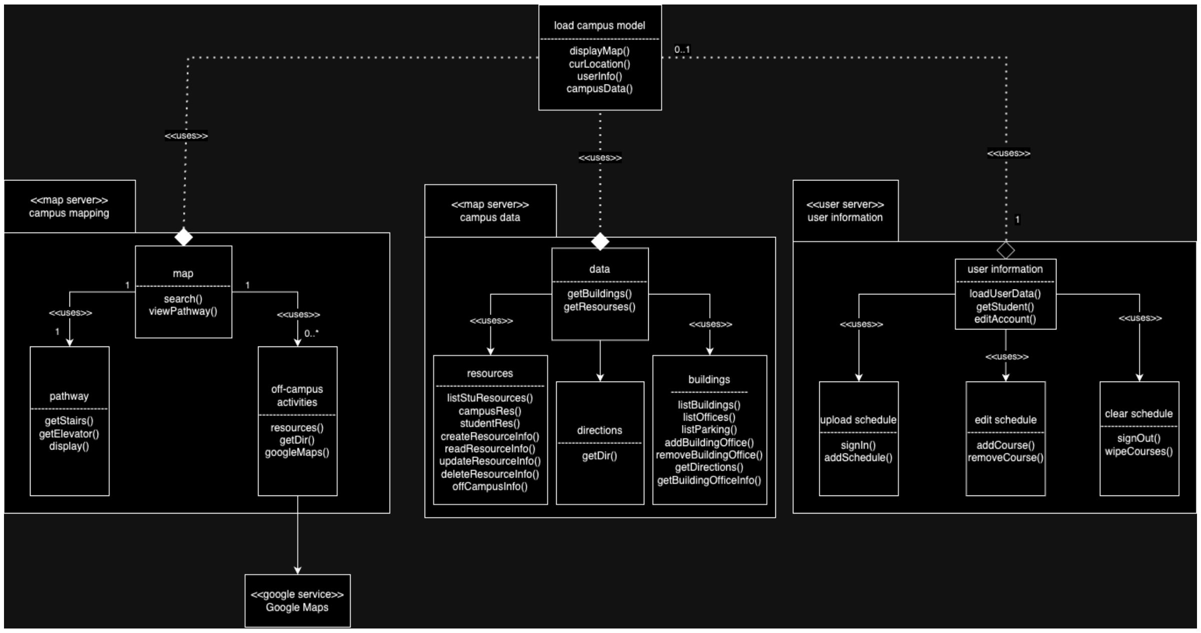
3.2.2 Processing sequence for [Information]



Caption

User launches software and the map is displayed. The user searches/selects a location and the building information is displayed. The information includes phone numbers, hours of operations, and other necessary information related to the location.

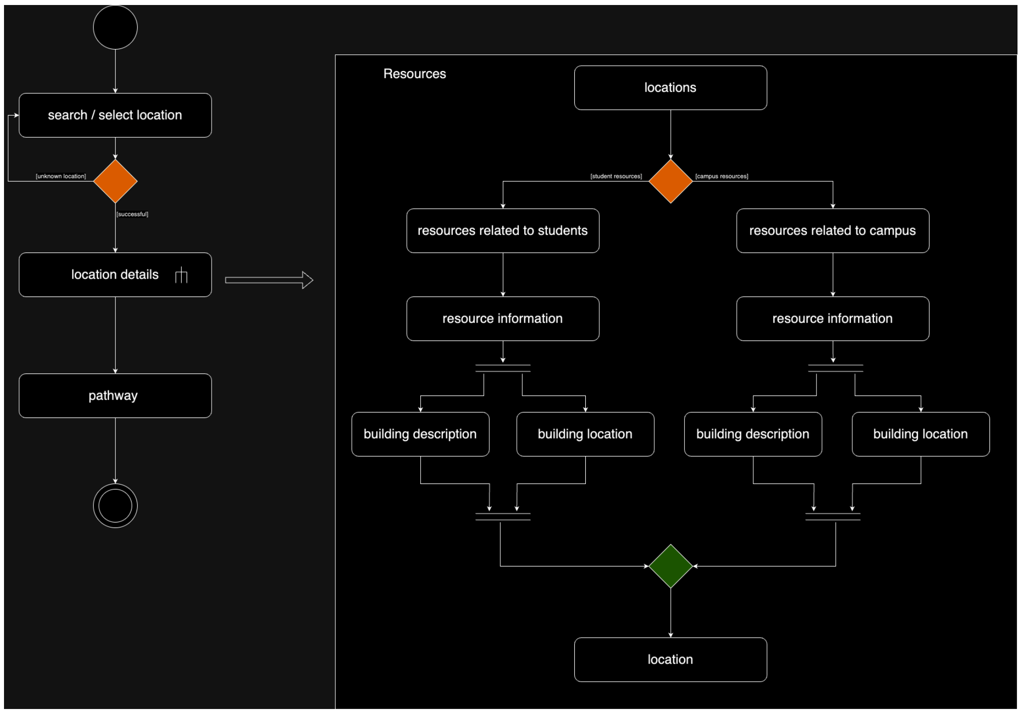
3.2.3 Structural Design for [Information]



Caption

The campus model is loaded and the campus map is displayed along with the users current location. To get information, campus data is called to get all the related information to the search/selected building. Information includes phone numbers, hours of operations, and other related details for the location.

3.2.4 Key Activities



Caption

User searches for a location, if the location isn't valid the user will have to enter a new location. If the location is valid, location details and information will appear. Based on the location, either student resources or campus resources will be called. Information regarding the selected resource will be displayed including a description and location. All the information for the selected location will be displayed.

3.2.5 Software Interface to other components

4 User interface design

4.1 Interface design rules

The following rules were established to ensure a consistent and user-friendly design for the Cougar Nav system:

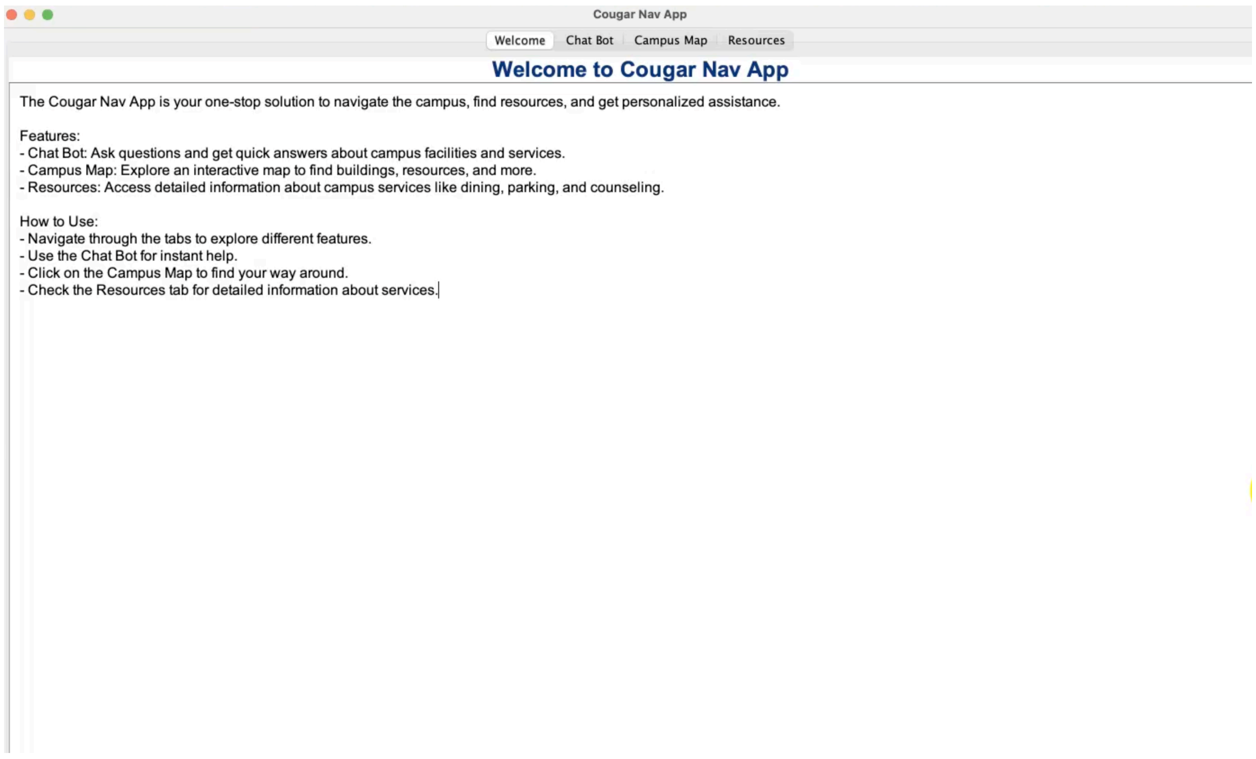
1. **Color Scheme**: Adhere to CSUSM branding colors for consistency with university materials.
2. **Navigation Consistency**: The navigation menu must remain accessible across all pages.
3. **Transitions:** from each page are quick and efficient to maintain high standards for the consumer.

4.2 Description of the user interface

The Cougar Nav system consists of the following key interface pages:

4.2.1 Welcome Page

- **Functionality**:
  The Welcome Page serves as the entry point for the application. It provides an overview of the system's features, including navigation, resource information, and the integrated chatbot.
- **Screen Image**: Displays a welcome message with options to navigate to other features.
- **Objects and Actions**:
  - **Navigation Menu**: Allows users to switch between the Chat Bot, Campus Map, and Resources tabs.



Welcome to Cougar Nav App

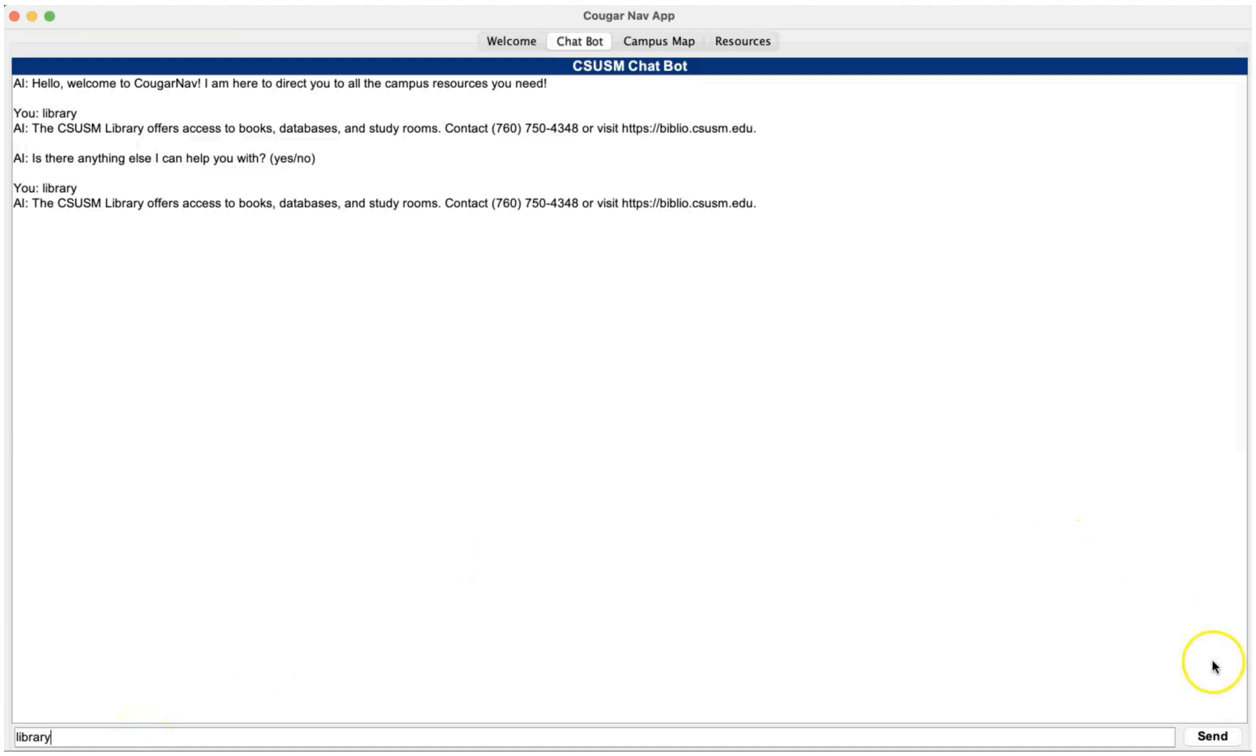
The Cougar Nav App is your one-stop solution to navigate the campus, find resources, and get personalized assistance.

Features:
- Chat Bot: Ask questions and get quick answers about campus facilities and services.
- Campus Map: Explore an interactive map to find buildings, resources, and more.
- Resources: Access detailed information about campus services like dining, parking, and counseling.

How to Use:
- Navigate through the tabs to explore different features.
- Use the Chat Bot for instant help.
- Click on the Campus Map to find your way around.
- Check the Resources tab for detailed information about services.

4.2.1 Chat box Page

- **Functionality**:
  Provides users with a conversational interface to query information about campus resources.
- **Screen Image**: Features an AI chat window where users can type queries.
- **Objects and Actions**:
  - **Input Box**: Users type their questions (e.g., "library").
  - **Submit Button**: Sends the query to the AI chatbot.
  - **Response Area**: Displays AI-generated responses with links, short information and contact details.



```
AI: Hello, welcome to CougarNav! I am here to direct you to all the campus resources you need!

You: library
AI: The CSUSM Library offers access to books, databases, and study rooms. Contact (760) 750-4348 or visit https://biblio.csusm.edu.

AI: Is there anything else I can help you with? (yes/no)

You: library
AI: The CSUSM Library offers access to books, databases, and study rooms. Contact (760) 750-4348 or visit https://biblio.csusm.edu.
```

4.2.1 Campus Map Page

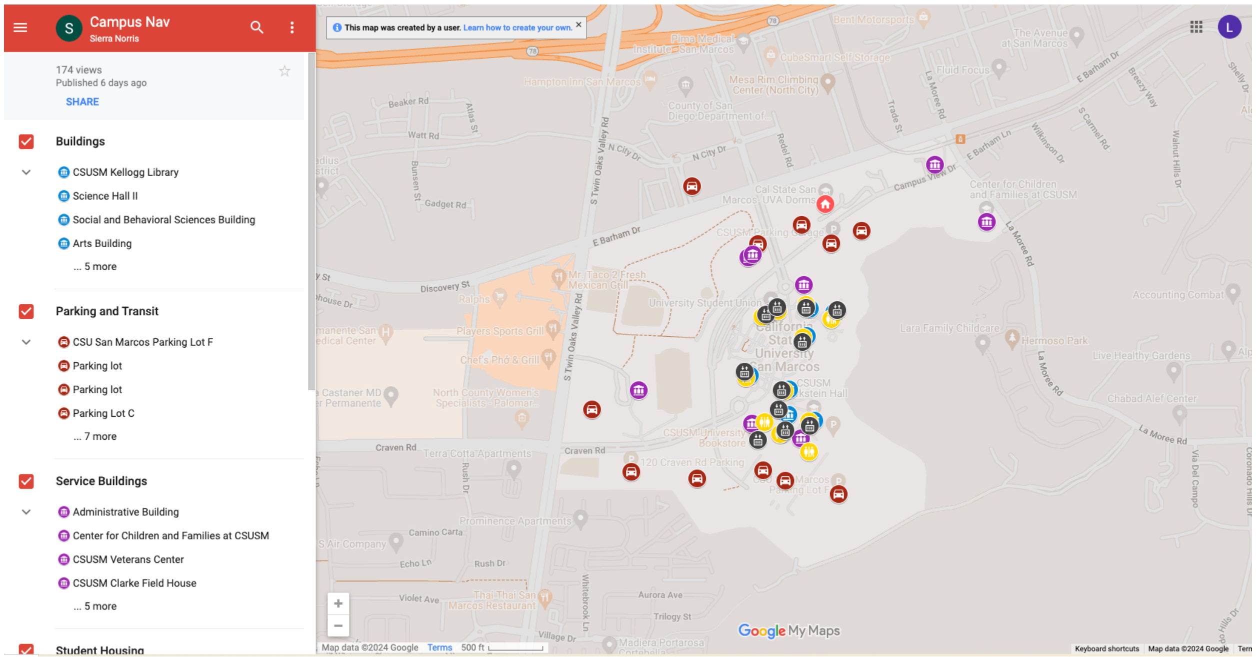
Functionality:

An interactive map showcasing key campus locations, including buildings, parking, and services.

Screen Image: Displays a map with labeled markers and filters for navigation.

Objects and Actions:

- **Map Markers**: Highlight buildings and resources. Clicking a marker displays additional information.
- **Search Bar**: Allows users to search for specific locations.
- **Filter Options**: Users can filter results based on categories like "Parking" or "Student Services."



Campus Nav
Sierra Norris

174 views
Published 6 days ago
SHARE

Buildings
- CSUSM Kellogg Library
- Science Hall II
- Social and Behavioral Sciences Building
- Arts Building
- ... 5 more

Parking and Transit
- CSU San Marcos Parking Lot F
- Parking lot
- Parking lot
- Parking Lot C
- ... 7 more

Service Buildings
- Administrative Building
- Center for Children and Families at CSUSM
- CSUSM Veterans Center
- CSUSM Clarke Field House
- ... 5 more

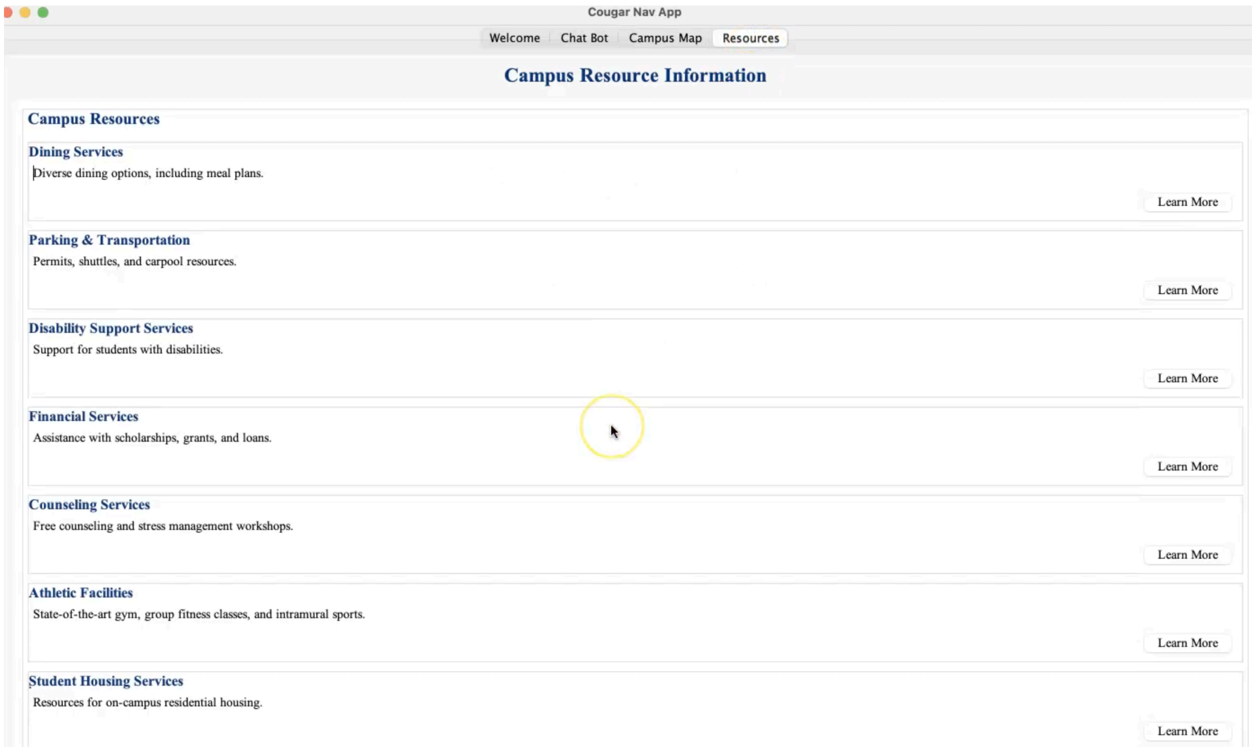
Student Housing

This map was created by a user. Learn how to create your own.

4.2.1 Resource Page

Functionality:
Provides detailed information about campus services, such as dining, counseling, and financial aid.
Screen Image: A list of categorized resources with "Learn More" buttons.
Objects and Actions:

- **Resource List**: Categories such as Dining Services, Financial Aid, and Disability Support.
- **Learn More Buttons**: Displays additional details about the selected resource.



Campus Resource Information

Campus Resources

Dining Services
Diverse dining options, including meal plans.
Learn More

Parking & Transportation
Permits, shuttles, and carpool resources.
Learn More

Disability Support Services
Support for students with disabilities.
Learn More

Financial Services
Assistance with scholarships, grants, and loans.
Learn More

Counseling Services
Free counseling and stress management workshops.
Learn More

Athletic Facilities
State-of-the-art gym, group fitness classes, and intramural sports.
Learn More

Student Housing Services
Resources for on-campus residential housing.
Learn More

Figure 3 - Main Menu

4.2.1.2 Objects and Actions

The Welcome Page serves as the central navigation hub for the Cougar Nav application, providing users with easy access to the Welcome, Chat Bot, Campus Map, and Resources tabs through a consistent navigation menu visible across all pages. Each tab allows users to interact with specific features, such as chatting with the integrated AI in the Chat Bot tab to ask questions about campus resources, exploring the interactive map in the Campus Map tab to view building locations and pathways, or accessing detailed resource information in the Resources tab. The interface ensures simplicity and efficiency, enabling users to seamlessly switch between features and navigate the CSUSM campus effectively.

5 Restrictions, limitations, and constraints

- *Java language only*
- *Google Maps API key*
- *Firebase Authentication*
- *User location settings*
- *Android 10 or higher*
- *2GB of RAM*
- *200MB free storage*
- *Java 17 or 18*
- *JavaFx-17*
- *Sqlite*

6 Testing Issues (SLO #2.v)

Test strategy and preliminary test case specification are presented in this section. Additional test cases are located in the Appendix section in spreadsheet format.

6.1 Types of tests

You may consider the following types of tests:

1. **Performance Test** – for example, to ensure that the response time for information retrieval is within an acceptable range. You typically should provide a specific performance bounds. For example, the search process should not take longer than 30 seconds.
2. **Accuracy Test** – for example, to determine if queries return the expected results.
3. **User Interface Test** – for example, to make sure the user interface is clear and easy to use with all types of users. Unfamiliar user can use the interface with minimal instruction and achieve the desired results.
4. **Security test** – for example, to ensure that users can only perform the tasks specified for their user group
5. **Repeatability Test** – for example, the software returns the same result for repeated queries.

We used Performance testing, Accuracy testing, User Interface testing, and Repeatability testing. We did not use Security testing because there are no features that are used for specific groups.

6.2 List of Test Cases

You should document each test case in the following format:

Test Type	**Accuracy Test**
Testing range	Resource searching
Testing Input	Science Hall 1
Testing procedure	Search for Science Hall 1 in the search bar
Expected Test Result	Building information for Science Hall 1
Testers	Shane Wasserman
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Resource searching
Testing Input	Science Hall 2
Testing procedure	Search for Science Hall 2 in the search bar
Expected Test Result	Building information for Science Hall 2
Testers	Shane Wasserman
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Resource searching
Testing Input	University Hall
Testing procedure	Search for University Hall in the search bar
Expected Test Result	Building information for University Hall
Testers	Shane Wasserman
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Resource searching
Testing Input	Academic Hall
Testing procedure	Search for Academic Hall in the search bar
Expected Test Result	Building information for Academic Hall
Testers	Shane Wasserman
Test result	Passed

Test Type	**Accuracy Test**

Testing range	Resource searching
Testing Input	Pleasant Park
Testing procedure	Search for Pleasant Park in the search bar
Expected Test Result	Unknown building
Testers	Shane Wasserman
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Location mapping
Testing Input	PS1
Testing procedure	Search for PS1 in the search bar and get pathway to Kellogg Library from current location
Expected Test Result	Display pathway to PS1 from current location
Testers	Shane Wasserman
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Location mapping
Testing Input	Kellogg Library
Testing procedure	Search for Kellogg Library in the search bar and get pathway to Kellogg Library from current location
Expected Test Result	Display pathway to Kellogg Library from current location
Testers	Shane Wasserman
Test result	Passed

Test Type	**Accuracy Test**
Testing range	Location mapping
Testing Input	Kellogg Library w/ elevators
Testing procedure	Search for Kellogg Library in the search bar and use elevator pathway
Expected Test Result	Display pathway to Kellogg Library using elevator route only
Testers	Shane Wasserman
Test result	Failed

Test Type	**Accuracy Test**
Testing range	Resource searching

Testing Input	Kellogg Library w/ elevators (retry)
Testing procedure	Search for Kellogg Library in the search bar and use elevator pathway
Expected Test Result	Display pathway to Kellogg Library using elevator route only
Testers	Shane Wasserman
Test result	Passed

Test Type	**Performance Test**
Testing range	Location mapping
Testing Input	Science Hall 1
Testing procedure	Search for Science Hall 1 in the search bar and get pathway to display in under 10 seconds
Expected Test Result	Display pathway to Science Hall 1 in under 10 seconds
Testers	Sierra Norris
Test result	Passed

Test Type	**Performance Test**
Testing range	Resource Information
Testing Input	SBSB information
Testing procedure	Search for SBSB in the search bar and have related information be displayed in under 5 seconds
Expected Test Result	Display SBSB information in under 10 seconds
Testers	Sierra Norris
Test result	Passed

Test Type	**User Interface Test**
Testing range	User Interface
Testing Input	
Testing procedure	Test all functions to make sure they are easily understandable and working
Expected Test Result	Easy to use and understand user interface
Testers	Javier Gutierrez
Test result	Passed

Test Type	**Repeatability Test**
Testing range	All previous tests
Testing Input	All previous tests

Testing procedure	Retry all tests to make sure they passed and work as intended
Expected Test Result	All tests come back as passed
Testers	Leo Beltran-Mariano
Test result	Passed

Note: you may have more than one test cases for each type of tests. If the tester and test results information is not available, you may add it in the final submission.

6.3 Test Coverage

For each of the functional requirements, describe if it has been achieved by the system or not.

For each of the non-functional requirements, describe if it has been achieved by the system or not.

All of the functional and non-functional requirements have been achieved.

Functional:

- Search function works as intended
- Resource information is correctly displayed
- Chatbot responds with correct answers
- Stair and Elevator route properly display
- Login to account

Non-Functional:

- Pathway is displayed in under 10 seconds
- Resource information is displayed in under 10 seconds
- Chatbot responds under 5 seconds

7 Appendices

7.1 Packaging and installation issues

How to install and prepare the system to run

For example, many teams use a database for your system. Some teams use FireBase, some use Google services, others use customized servers (located in dorm). Whatever techniques you are using, you need to describe how you set up the DB and how to create DB connections (showing some code snippet would help).

This part will be used by the instructor to evaluate your self-learning ability. When you describe a technique (even it is a small tool like Postman), provide sufficient information such that it can be used as a tutorial for a novice to quick get it up and running.

To prepare to run the program, users need to have Java versions 17 or 18, JavaFx-17, and Sqlite for Java installed. Once these three downloads are successfully installed, users can run our programs code and the software will display.

To download Java version 17:
Go to Oracle website: https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

Download the correct version based on users set up

To download JavaFx-17:

Go to JavaFx website: https://openjfx.io/

Download the Scene Builder and TestFx

To download sqlite for java:

Go to sqlite website: https://www.sqlitetutorial.net/sqlite-java/

7.2 User Manual

Launching the Application: Open Cougar Nav on your preferred device. The Welcome Page will greet you with an overview of the system and its features.
Chat Bot: Navigate to the Chat Bot tab to ask questions about campus services or facilities. Type your query (e.g., "library" or "parking") into the input box, and the AI will provide a response with relevant details.
Campus Map: Click on the Campus Map tab to view an interactive map of the CSUSM campus. Use the map to locate buildings, parking areas, and other resources. Click on any map marker to see additional information about that location.
Resources: Access the Resources tab to browse detailed information about campus services, such as dining, counseling, and financial aid. Click on "Learn More" to view extended descriptions and contact details.
Navigating Between Pages: Use the navigation menu at the top of the application to switch between features seamlessly.

7.3 Open Issues

Live Route Tracking: The ability to track a user's real-time location on the map is not yet implemented.
Multilingual Support: Adding language options for non-English speakers to improve accessibility.
Offline Mode: The system does not yet support offline access to campus maps and resources.
Student Link: The ability for students to upload their schedule to the software to automatically fill with classroom locations and times is not yet available. We wanted to have students be able to log in into their student accounts so the app can access saved data for each student user like class schedule and times but we did not have enough time to complete log in page.

7.4 Lessons Learned

7.4.1 Project Management & Task Allocations (SLO #2.i)

Make sure your response covers the following aspects (refer to the Project Evaluation Rubrics to see how this part will be evaluated).

- How your team allocate tasks and responsibilities and where could you improve in the future?
- How your team do project planning activities?
- How much effort do your team perform risk analysis along the process?
- How did your team update each other’s progress and adjust your plans?
- How did your team track changes and respond to changes occurred?

Our team allocated tasks based on each member’s strengths and expertise, dividing responsibilities into subsystems like navigation, resource management, and UI development. While this approach was effective, future improvements could include more detailed documentation of individual responsibilities to avoid overlaps. We planned the project using weekly meetings to set milestones and deadlines, ensuring steady progress. Risk analysis was performed at key stages, such as during the initial system design and after each sprint, allowing us to mitigate potential delays. Regular updates were shared via group chats and collaborative tools, and plans were adjusted dynamically based on progress. We used version control (GitHub) to track changes, ensuring that all modifications were documented and reversible if necessary.

7.4.2 Implementation (SLO #2.iv)

Make sure your response covers the following aspects (refer to the Project Evaluation Rubrics to see how this part will be evaluated).

- How did your team performed code review and refactoring to improve code quality? Provide a few code snippets (screenshots) that can best represent the team's coding quality.

Our team performed code reviews and refactoring over in-person meetings as well as zoom meetings. We would meet to discuss testing and code reviews. If a test did not pass, as a team we would examine the code and debug to find issues within our code. By using this method we were able to refactor our code and rerun failed tests until we passed.

- How much is your implementation consistent with your system design?

Our implementation is pretty close to the system design. We are not 100% consistent due to busy schedules and not enough time to fix all minor imperfections. We tried to incorporate student login but did not have enough time to further dial it in.

7.4.3 Design Patterns

What are the design patterns used and why?

We used the **Façade Pattern** to simplify interactions between the user interface and the subsystems, making the system easier to maintain and extend. Additionally, the **DAO Pattern** was implemented to abstract database operations, ensuring separation of concerns and improved scalability. These patterns were chosen for their ability to modularize the system and enhance maintainability.

7.4.4 Team Communications

How team communications were conducted and where could you improve in the future?

Team communications were conducted through a combination of weekly in-person meetings, online zoom meetings, and daily online updates via messaging platforms such as discord. In the future, more structured communication tools, such as task boards or progress tracking software, could improve efficiency and ensure no details are overlooked.

7.4.4 Technologies Practiced (SLO #7)

Describe all new technologies/skills you would not have used/learned if you were not given the opportunity to work on the course project.

If we were not given the opportunity to work on the course project then we would have missed out on learning IntelliJ, Google Maps API, JavaFx, Trello, GitHub.

Also, we would not have learned methods for working such as agile approach, scrums, and various testing styles (user interface, accuracy, performance, repeatability).

7.4.5 Desirable Changes

Assume that you have another month to work on the project, what aspects of the system you would like to improve? What are the additional features you want to add to the system? [Each student should use a separate paragraph to respond to the questions]

Shane Wasserman: I would focus on implementing real-time user location tracking and adding an advanced search feature to improve navigation efficiency. Being able to get a closer visual of the pathway and improved live feedback would make our software even more usable. Also, being able to update and reroute users based on their current location would make navigating towards the location a lot more precise.

Sierra Norris: I would like to enhance the Campus Map by including more detailed overlays, such as live parking availability and building information. If I had more time I would have liked to dive deeper into the parking situation on campus. With more students taking up parking, it would be convenient to include live parking updates and spot finder to help students park more efficiently. If I had more time, I would also like to include more information for each building. I wold like to include the classroom numbers in each building, number of floors, and other details about certain buildings that would help students or visitors who are not familiar with the campus.

Javier Gutierrez: I would improve the AI Chat Bot's functionality by integrating a more comprehensive knowledge base to answer a broader range of user queries. With more time I would like to improve the chatbot we included to be more helpful. I would have the AI give more detailed descriptions as well as be able to answer more advanced questions that users might have about the campus or software.

Leonardo Beltran-Mariano: Adding multilingual support and offline mode would greatly enhance the system's accessibility for a wider range of users. Having more time would allow me to figure out how to get the software in more languages for abroad students and visitors. Having the ability to choose a language would help students who travel use the software. An offline mode would also be a good idea to add if I had more time. If students loose connection to the internet or campus wifi, then they would not lose the ability to navigate with the software because offline support would keep them connected.

7.4.6 Challenges Faced

Among requirements specification, system design, and system implementation, which one you think is the hardest task? Why? [Each student should use a separate paragraph to respond to the questions]

Shane Wasserman: System design was the hardest task due to the need to ensure all components would interact seamlessly without creating bottlenecks. Getting all the code to correctly call functions without interfering with other system functions was a challenge and required a lot of attention. Making sure the changes we made to one function did not affect the results of another one proved to challenging.

Sierra Norris: Requirements specification was the most challenging because it required balancing user needs with technical feasibility, especially with the large scope of the campus map. Also, revising previous diagrams and work was another time consuming tasks to make sure they matched new work and all requirements. Having to make large changes to codes or diagrams to match would take a lot of time and was difficult to make them both include all requirements.
Javier Gutierrez: System implementation posed the greatest difficulty, particularly when integrating subsystems, as unforeseen bugs and compatibility issues arose frequently. Making changes to certain system functions would affect the results of other functions and getting them to all work together without interference was hard. With large amounts of code, getting everything to be implemented correctly was stressful and challenging.
Leonardo Beltran-Mariano: I found system design to be the hardest because translating abstract ideas into concrete architectural components required extensive planning and foresight. Coming up with ideas was easy but including them into the code was hard. If there was a new idea we wanted to add, we had to make sure it was able to be implemented into our design and not mess up already existing functions. Getting the overall design to work with all the new additions and revisions that we being made was stressful and difficult.