(Cougar Nav)

(Team 5)

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

Software Requirements Specification Document

Version: (3) **Date: (12/11/2024)**

Table of Contents

Changelog

4.1 was updated to reflect chat box ai

4.2 updated to reflect chat box ai

4.3 was changed in the part where we were planning on using the google api integration.
6.2 was updated to add chat box ai

1.Purpose

This section describes

1) *the business background (or problem context).*
   Cougar Nav is a first of its kind campus navigation system for universities. Currently, there are no digital options for navigating through college campuses. This is problematic because with the growing reliability and usability of technology students will need a way to locate important items and destinations on campus. With campuses becoming increasingly large, it can be difficult to find classrooms, offices, restaurants, and other important destinations on campus. Cougar Nav will have the ability for students to upload their schedule which will automatically map the most efficient route to get to classrooms at scheduled times. The software will also have the ability to locate other important necessities such as atms, stairs, elevators, vending machines, restaurants, offices, student resources, and much more features on campus.
   When a student uses the software, they will have the option to use the app as a casual mapping system which will map out the entire university or they can choose to upload and link their course schedule to take maximum advantage of the software. When connecting the software to their schedule, students will have the most efficient route mapped from their current location to the classrooms they need when the class is scheduled.

2) *why a new system is needed. You should describe the existing problems, issues, or deficiencies in the business where the use of the new system can bring business values (by addressing the problems, issues, or deficiencies).*
   A new system is needed to provide the most efficient way to navigate our college campus considering students, professors and visitors. This will allow for users to plan ahead and find which routes work best for them to get to class on time and develop a better understanding of our campus and its available resources such as financial services, cougar pantry, counseling services. Currently, there is no softwares available with the ability to map out a college campus and locate important or frequently used destinations. Due to students not having a detailed map of the campus, they lose out on many features that the campus has to help students. Also, by using the new software, an easy increase in productivity is available to students which relieves stress and increases efficiency, confidence, and time management.

2.Scope

Describe the scope of the software by clearly list the desired objectives. This sets up a scope for the new system to be developed. You may choose to address only a few of the problems, issues, or deficiencies identified in the business. In particular, you should

1) *Give an appropriate name to the system (e.g., PSU Campus Map, Super Team Editor, etc.) and reference it by name in the rest of the document;*
2) *Explain goals (what the software will do);*
3) *Describe the application of the software, including potential benefits.*

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. 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.

3.User characteristics

Identifying the potential users of the product. Describe general characteristics of the intended groups of users (stakeholders) of the product, especially focusing on characteristics that may influence usability, such as educational level, experience, disabilities, and technical expertise.

3.1.Key users

They are critical to the continued success of the product. Give greater importance to requirements generated by this category of user.

- *User role responsibilities: what to do with the product.*
  - *key users are college students*
  - *upload course schedule*
  - *use map, search bar, menus to navigate campus*
  - *locate available parking*
  - *locate restrooms*
  - *provide available school resource location*

- *Subject matter experience: Summarizes the users' knowledge of the business (domain). Rate as novice, journeyman, or master.*
  - *journeyman, users have a limited knowledge about the business domain of mapping software's, however it is a simple enough concept to understand.*
- *Technological experience: Describes the users' experience with relevant technology. Rate as novice, journeyman, or master.*
  - *journeyman, this software will have a similar layout and usability as popular navigation software's such as google maps, Waze, apple maps, etc. However, it will have more educational resources for students to be educated about and find campus resources.*

- *Other user characteristics: Describe any characteristics of the users that have an effect on the requirements and eventual design of the product. For example:*
  - *Physical abilities/disabilities*
    - *disability accessible routing will be an available feature for people with physical abilities/disabilities to give more access to elevators and ramps.*
  - *Intellectual abilities/disabilities*
    - *software will be very easy to navigate for everyone*

        - *detailed labels and voice directions will be available*
    - *Attitude toward technology*
        - *easy to use software will keep users calm and positive*
    - *Education*
        - *users will be college students and well educated*
        - *high school graduates*
        - *college undergraduates*
        - *college postgraduates*
    - *Linguistic skills*
        - *primary language of the software will be english*
        - *pictured directions suitable for everyone*
    - *Age group*
        - *users will be ages 16+ but no age limit required*
    - *Gender*

        - *everyone*

3.2.Secondary users

They will use the product, but their opinion of it has no effect on its long-term success. Where conflict between secondary users' requirements and those of key users, the key users take

- *User role responsibilities: what to do with the product.*
    - *secondary users are faculty, staff, visitors*
    - *use map, search bar, menus to navigate campus*
    - *locate available parking*

- *Subject matter experience: Summarizes the users' knowledge of the business (domain). Rate as novice, journeyman, or master.*
    - *journeyman, users have a limited knowledge about the business domain of mapping softwares, however it is a simple enough concept to understand.*
- *Technological experience: Describes the users' experience with relevant technology. Rate as novice, journeyman, or master.*
    - *journeyman, this software will have a similar layout and usability as popular navigation softwares such as google maps, waze, apple maps, etc.*

- *Other user characteristics: Describe any characteristics of the users that have an effect on the requirements and eventual design of the product. For example:*
    - *Physical abilities/disabilities*
        - *disability accessible routing will be an available feature for people with physical abilities/disabilities*
    - *Intellectual abilities/disabilities*
        - *software will be very easy to navigate for everyone*
        - *detailed labels and voice directions will be available*
    - *Attitude toward technology*

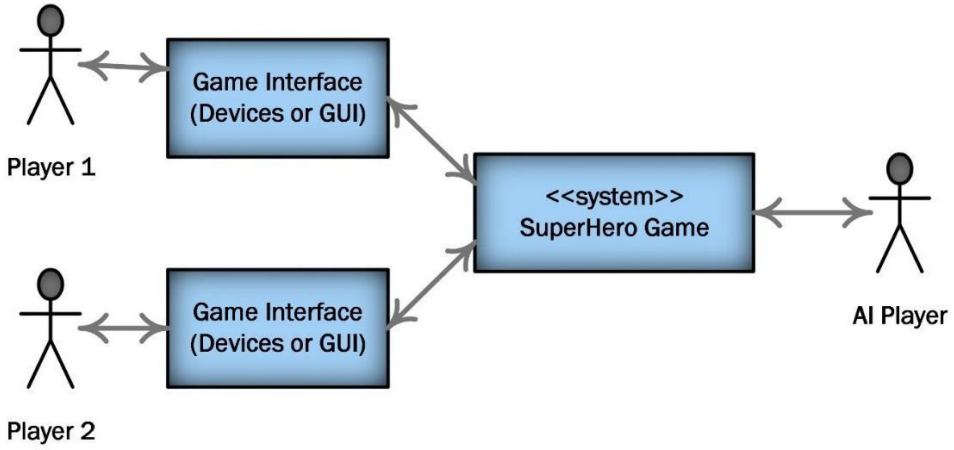
- *easy to use software will keep users calm and positive*

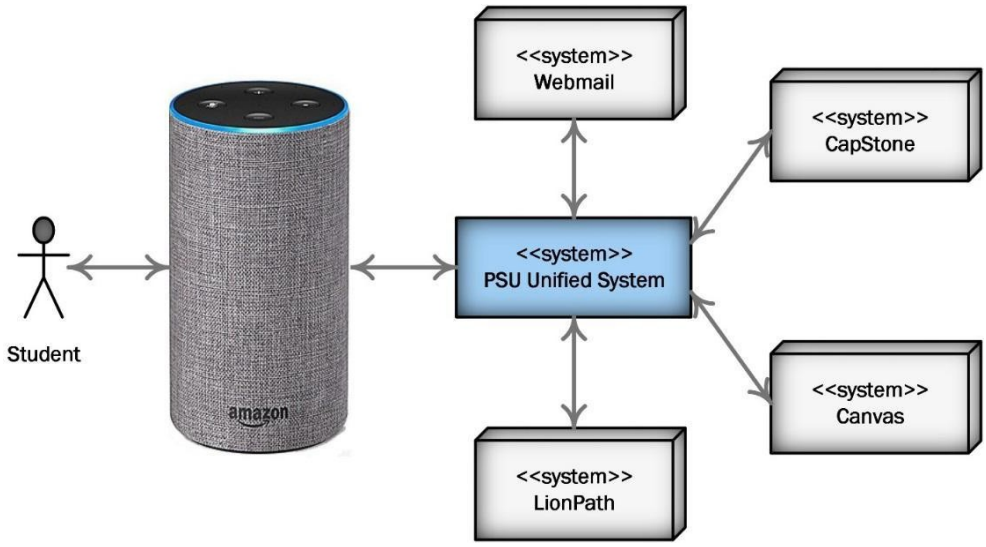
- *Education*
  - *educated college graduates*
  - *teachers/professors*
- *Linguistic skills*
  - *primary language of the software will be english*
  - *pictured directions suitable for everyone*
- *Age group*
  - *users will be ages 16+ but no age limit required*
- *Gender*
  - *everyone*

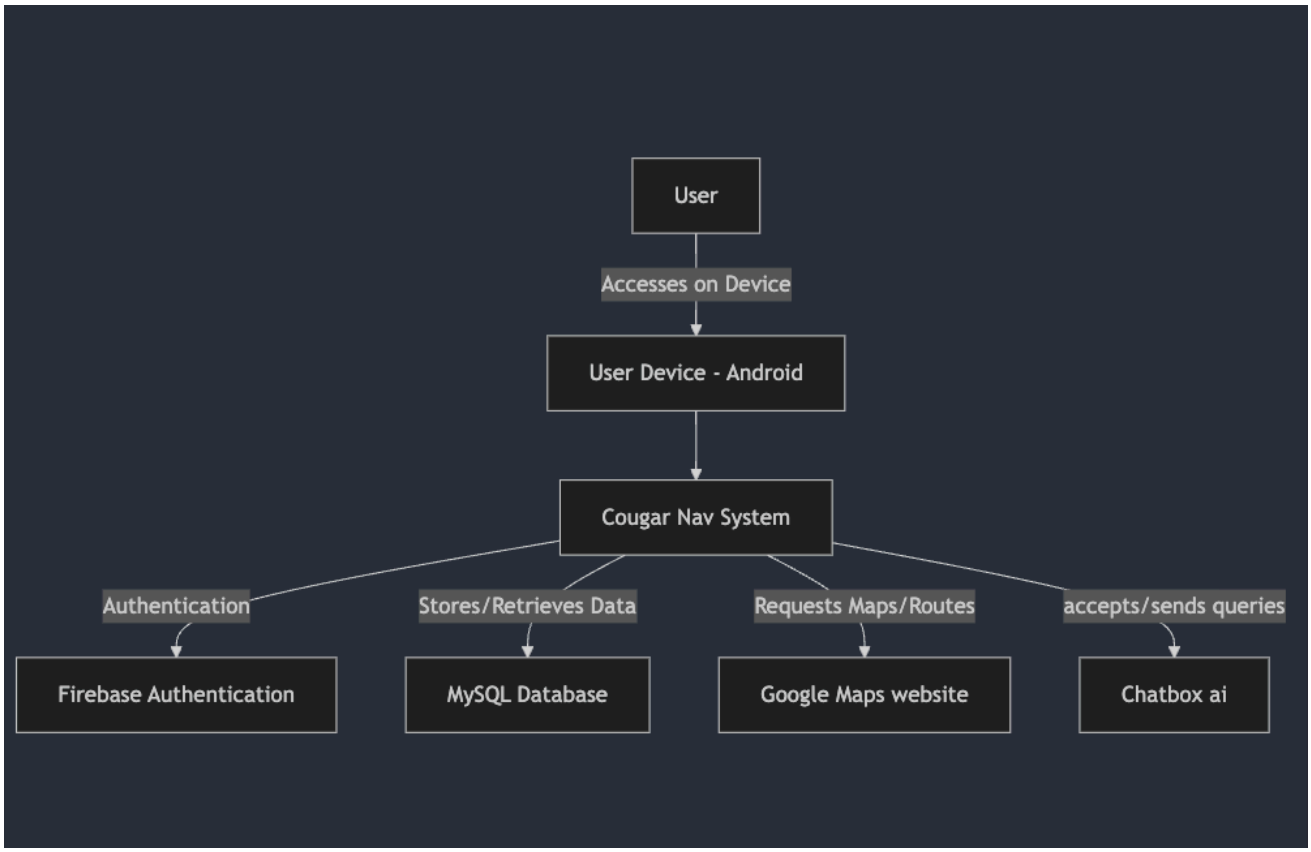
4.Product perspective

4.1.System Context

Define the system's relationship to users or other related systems. If the system is an element of a larger system, then identify the interfaces between the system covered by this SRS and the larger system. A block diagram showing the major elements of the larger system, interconnections, and external interfaces can be helpful. Below are two example block diagrams (with the system to be developed highlighted):







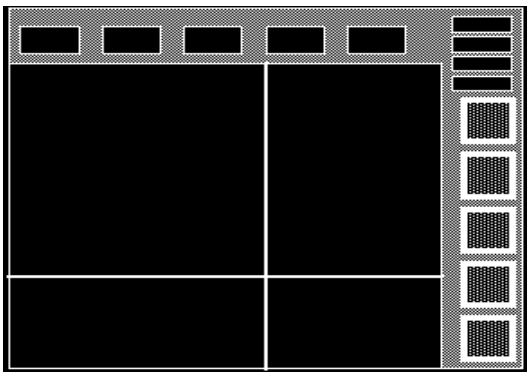
4.2.User interfaces

Specify the required characteristics of each interface between the software product and its users. Keep in mind that we are specifying requirements, so we here only care about the requirements on interfaces (this is NOT the place to put user interface designs by the software developers). For each requirement on an interface, it may include:

- *Required screen formats (text only or multi-media),*
- *Required page or window layouts,*
- *Required content of any reports, menus, error messages,*
- *Required backup and recovery operations, or*
- *Required user-initiated operations or function*

keys. Some examples:

1) *It is required that the system provides the option of long and short error messages.*
2) *It is required that the system allows users to save work-in-progress and load previous works*
3) *It is required that the system interfaces have the following screen layout:*



1) The interface must support multimedia, including text, images, and icons for building landmarks, pathways, and campus resources.
2) Interactive maps will be displayed with zoom-in/out capabilities and options to switch between different views (e.g., street view, satellite view).
3) The app must provide detailed error messages with options for long and short formats, explaining issues like incorrect search terms or unavailable resources.
4) Menus must display resource categories (e.g., clubs, restrooms) with descriptive icons and short text labels.
5) Users must be able to click on a building or location on the map to view detailed information (e.g., available resources, room numbers).
6) Shortcut keys (e.g., swipe gestures or double-tap) must allow users to set the starting point and destination quickly.
7) System will have an integrated chat ai section for queries.

4.3.Software interfaces

Specify the use of other required software systems (e.g., a data management system, an operating system, or a mathematical package), and interfaces with other software systems.

*[Note: This is only for **customer-specified systems** that you **have** to interact with. Choosing SQL Server 7 as a DB without a customer requirement is a Design choice, not a requirement. This is a subtle but important point to writing good requirements and not over-constraining the design.]*

- *For each required software product, if possible, specify: a) Software name; b) Software specification number; c) Software version number; d) Source.*
- *For each interface specify the required message formats. If the interface is well-documented elsewhere, provide a reference to the document defining the interface.*

For instance if your customer uses SQL Server 7 DB and you are required to use that, then you need to specify

- *The system must use SQL Server 7 as its database component.*
- *Communication with the DB is through ODBC connections.*

- **Database Management System**:
  - **a)** Software Name: MySQL
  - **b)** Software Specification Number: Open-source database management system
  - **c)** Software Version Number: Version 8.0
  - **d)** Source: MySQL Community Edition
  - **Message Format**: Communication with the MySQL database will use SQL queries via JDBC (Java Database Connectivity) connections.
- **Operating System**:
  - **a)** Software Name: Android OS
  - **b)** Software Specification Number: Mobile Operating System
  - **c)** Software Version Number: Android 10+
  - **d)** Source: Google
  - **Message Format**: The app will interact with the Android OS using Java APIs for accessing GPS, camera, and notification services.
- **Mapping and Navigation API**:
- **Instead we used a website in which we were able to create our own map of the campus which the user is taken to this website a pound clicking on the map of the csusm in the app**.
  - **a)** Software Name: Google Maps API
  - **b)** Software Specification Number: Google Maps Basic API
  - **c)** Software Version Number: Version 2.0
  - **d)** Source: Google
  - **Message Format**: The app will communicate with Google Maps through simple RESTful API calls using HTTPS, with JSON data for routes and location information.
- **Authentication System**:
  - **a)** Software Name: Firebase Authentication
  - **b)** Software Specification Number: Firebase Authentication Service
  - **c)** Software Version Number: Latest Version
  - **d)** Source: Firebase (by Google)
  - **Message Format**: User authentication will occur using JSON Web Tokens (JWT) for login and registration processes.

4.4.Deployment requirements

Specification of the deployment environment that is required for the installation and operation of the software.

If any modifications to the customer's work area would be required by your system, then document that here. Any equipment the customer would need to buy or any software setup that needs to be done so that your system will install and operate correctly should be documented here.

This could be hardware-specific, For instance, "A 100Kw backup generator and 10000 BTU air conditioning system must be installed at the user site prior to software installation".

This could also be software-specific like, "New data tables created for this system must be installed on the company's existing DB server and populated prior to system activation."

1. Deployment Environment Requirements:

- *The app must be deployed on Android smartphones with a minimum version of Android 10 or higher.*
- *The device must have at least 2GB of RAM and 200MB of free storage space for installation and operation.*
- *GPS functionality must be enabled on the smartphone for accurate location tracking.*

2. Network Requirements:

- *A stable internet connection (Wi-Fi or mobile data) is required for accessing real-time map data and resources from Google Maps API.*
- *The app must have permission to access the internet and location services during installation.*

3. Software Setup Requirements:

- *Google Maps API keys must be obtained and configured in the app settings before deployment to ensure proper map functionality.*
- *Firebase Authentication must be set up with valid API keys to allow user login and registration.*
- *The MySQL database should be set up on the school's existing server, with the necessary tables created and populated before system activation.*

4. Modifications to the Customer's Work Area:

- *If the system is intended for use on public campus help desks, these devices must be equipped with touch-screen capabilities and a secure internet connection.*
- *Any additional hardware requirements, such as the installation of campus Wi-Fi hotspots to provide internet access in outdoor areas, must be completed before app deployment.*

5.Assumptions and Dependencies

List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but any changes to these factors can affect the requirements in the SRS.

For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

Assumptions:

1. **Operating System Availability**: Users will have devices running Android 10 or higher. If not, the app's compatibility requirements may need to be adjusted.
2. **GPS and Internet Connectivity**: Users have functional GPS and internet access. Lack of these features could impair location services.
3. **Google Maps API Access**: The app assumes continued access to Google Maps API for navigation. If unavailable, alternative mapping services would be needed.
4. **Campus Wi-Fi Coverage**: Adequate Wi-Fi coverage is assumed across the campus. Areas without coverage may require offline map support.
5. **Firebase Authentication**: The system assumes Firebase services will be available. Changes may require a switch to another authentication method.
6. **Database Server Accessibility**: Continuous access to the MySQL server is assumed. Interruptions may necessitate data handling adjustments.

Dependencies:

1. ***API Keys and Licensing****: The app depends on valid API keys for Google Maps and /or Firebase. Changes in terms could require system modifications.*
2. ***Device Hardware****: User devices must meet minimum hardware requirements (e.g., GPS). Variation could impact performance.*
3. ***Campus Map Accuracy****: The app relies on accurate and up-to-date campus maps. Changes in infrastructure may affect route calculations.*

6.Specific requirements

This is a specification, a designer should be able to read this spec and build the system without bothering the customer again.

- *Specify all of the software requirements to a level of detail sufficient to enable designers to design a software system to satisfy those requirements.*
- *Specify all of the software requirements to a level of detail sufficient to enable testers to test that the software system satisfies those requirements.*
- *Each requirement should be uniquely identifiable for traceability. Usually, they are numbered R.1, R.1.1,*
  R.1.2.1 etc. so that each can be cross-referenced in other documents.
- *Each requirement should also be testable. Avoid imprecise statements like, "The system shall be easy to use." What is easy to use?*

Use proper terminology:

- *A required, must have feature: The system shall…*
- *An optional, nice-to-have feature that may never make it to implementation: The system may…*

Functional Requirements

- **FR.1:** The system will run on Android 10+.

- ***FR.2****:* The system will provide an interactive map with zoom and pan features, showing indoor and outdoor routes, including elevators and stairways.

- **FR.3**: The system will allow users to search for campus locations (e.g., buildings, restrooms, clubs) and provide the shortest walking route.
- **FR.4**: The system will provide a “Locate Me” feature that centers the map on the user’s current location.
- **FR.5**: The system will enable user authentication using Firebase, with features to save frequently accessed routes and preferences.
- **FR.6**: The system will store user data and map details in a MySQL database, updating dynamically to reflect campus changes.

Non-Functional Requirements

- ***NFR.1****:* The system should load the map interface within 3 seconds after opening the app.
- **NFR.2**: The system should update the user’s position every 5 seconds during navigation.
- **NFR.3**: The system should function offline for cached areas of the map, providing limited navigation capabilities.
- **NFR.4**: The system should encrypt all user data stored in the database to ensure privacy and security.
- **NFR.5**: The system should log users out automatically after 30 minutes of inactivity to enhance security.

Avoid over-constraining your design. Do not require specific software packages, etc., unless the customer specifically requires them.

Avoid examples: Don’t say things like, “The system shall accept configuration information such as name and address.” The designer doesn’t know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.

6.1.System Functional Requirements

Provide a list of the major functions that the software will perform. For example, an SRS for an accounting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those functions requires.

As mentioned in lecture, you should try the best to use the User Story Template to document system functional requirements.

1. **User Location Tracking**
- As a student, I want to see my current location on the map, so that I can orient myself and find directions quickly.
2. **Campus Navigation**
- As a student, I want to search for a building, classroom, or campus resource, so that I can get the shortest walking route to my destination.
3. **Resource Information Display**
- As a student, I want to click on buildings or resources on the map, so that I can view details such as hours of operation and available facilities.
4. **Route Planning**

- As a student, I want to plan my route based on indoor and outdoor pathways, so that I can navigate efficiently and access elevators or stairways as needed.

5. **Offline Map Access**

- As a student, I want to access previously saved or cached map areas offline, so that I can navigate even without internet access.

6. **User Authentication and Profile Management**

- As a user, I want to log in and create an account using Firebase Authentication, so that I can save my preferred routes and access them later.
- As a user, I want to save frequently visited locations, so that I can quickly access directions for these locations in the future.

7. **"Locate Me" Feature**

- As a student, I want to tap a button to quickly find my location, so that I can center the map on my position and get immediate directions.

6.2.Logical Database Requirements

Specify the logical requirements for any information that is to be placed into a database, including data entities and their relationships, and Integrity constraints.

If the customer provided you with data models, those can be presented here. ER diagrams (or static class

can be useful here to show entity relationships. Remember a diagram is worth a thousand words of

1. Entities and Attributes

1. **Users**
   - Attributes:
     - `user_id` (Primary Key): Unique identifier for each user.
     - `username`: The name chosen by the user.
     - `email`: User's email for authentication.
     - `password_hash`: Hashed password for secure login.
     - `saved_locations`: List of locations frequently saved by the user.
     - `preferences`: User settings like preferred walking speed.
2. **Buildings**
   - Attributes:
     - `building_id` (Primary Key): Unique identifier for each building.
     - `building_name`: Name of the building.
     - `location_coordinates`: GPS coordinates (latitude, longitude).
     - `hours_of_operation`: Operating hours of the building.
     - `facilities`: List of available facilities (e.g., restrooms, elevators).
3. **Routes**
   - Attributes:
     - `route_id` (Primary Key): Unique identifier for each route.
     - `start_location`: Start location's GPS coordinates or building ID.
     - `end_location`: End location's GPS coordinates or building ID.
     - `route_details`: Path details (indoor/outdoor pathways, distance, estimated time).
4. **Campus Resources**
   - Attributes:

- `resource_id` (Primary Key): Unique identifier for each resource (e.g., club, restroom).
- `resource_type`: Type of resource (e.g., club, restroom, library).
- `location_coordinates`: GPS coordinates (latitude, longitude).
- `associated_building_id`: Foreign Key linking to the building where the resource is located.

Chat Box AI

- *Attributes:*
  - *ai_id (Primary Key): Unique identifier for the AI.*
  - *knowledge_base: Data source containing information about campus.*
  - *response_logic: Algorithms for generating responses.*
  - *user_history: Log of user interactions with the AI.*

2. Entity Relationships

- A User can save multiple Buildings and Routes as favorites, establishing a one-to-many relationship between Users and both Buildings and Routes.
- A Building may contain multiple Campus Resources, establishing a one-to-many relationship between Buildings and Campus Resources.
- Routes connect different Buildings, forming associations between Buildings based on the start and end locations of each Route.

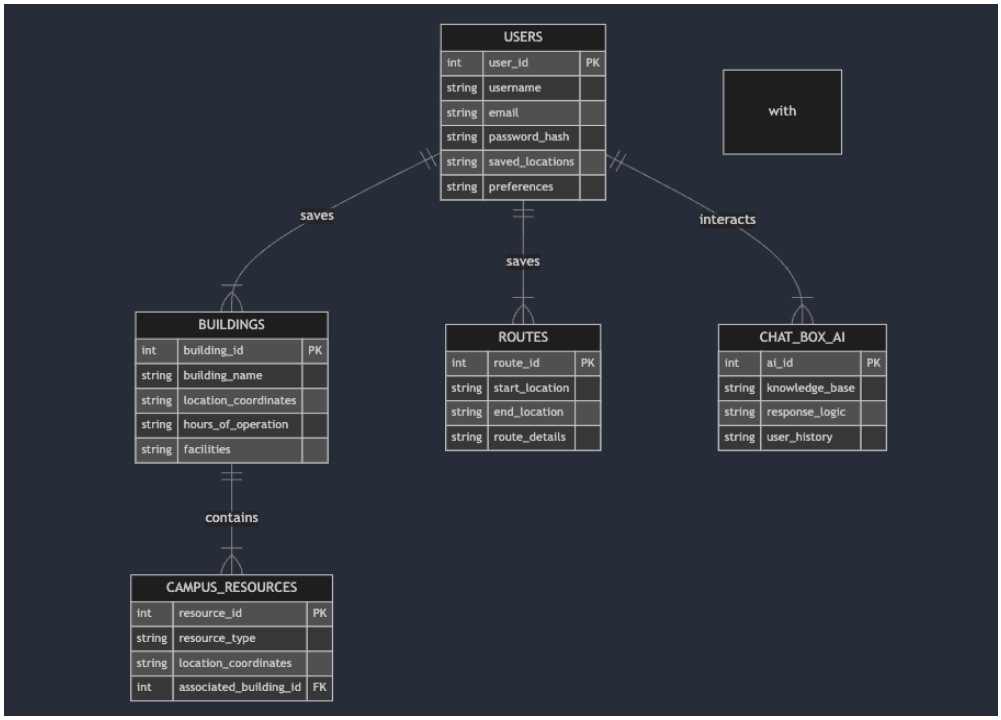
3. Integrity Constraints

- **Primary Keys**: Each entity must have a primary key (`user_id`, `building_id`, `route_id`, `resource_id`) to uniquely identify records.
- **Foreign Keys**:
  - `associated_building_id` in the Campus Resources table must reference an existing `building_id` in the Buildings table.
- **Data Consistency**:
  - `location_coordinates` should be in a standard format (latitude, longitude) to ensure consistency.
  - `hours_of_operation` should follow a standard format (e.g., 08:00 - 17:00) for consistency across entries.
- **Referential Integrity**:
  - Deleting a building should also remove associated resources and invalidate routes that use that building as a start or end point.

4. ER Diagram

To visually represent these entities and their relationships, an Entity-Relationship (ER) diagram can be used, showing:

- Entities (`Users`, `Buildings`, `Routes`, `Campus Resources`).
- Relationships (`User` saves `Buildings` and `Routes`; `Buildings` contain `Campus Resources`).
- Attributes within each entity and primary/foreign key associations.



6.3.Software System Attributes

Specify non-functional requirements, that is, the required attributes of the software product.

For each requirements in this section, make sure it is testable

1. Performance Requirements

- NFR.1: The system should load the main map interface within 3 seconds of launching the application.
    - Testable: Measure the time taken from app launch to map display under normal operating conditions.
- NFR.2: The system should update the user's position on the map every 5 seconds when the GPS signal is available.
    - Testable: Track and verify that the location updates on the map are performed every 5 seconds during active navigation.

2. Reliability Requirements

- NFR.3: The system should have an uptime of 99.5%, ensuring continuous access for users.
    - Testable: Monitor system logs over a defined period to check if uptime meets the specified percentage.
- NFR.4: The system should cache map data and frequently accessed locations to allow for offline navigation in previously viewed areas.
    - Testable: Disconnect the device from the internet and verify that cached maps are accessible.

3. Usability Requirements

- NFR.5: The system should allow users to access the "Locate Me" feature with a single tap from the main navigation screen.
    - Testable: Perform usability testing to verify that users can activate the feature with one tap.
- NFR.6: The system shall provide clear error messages for failed operations, such as "Location not found" or "GPS not available," within 2 seconds of the issue occurring.
    - Testable: Trigger each error condition and measure the response time and clarity of the message displayed.

4. Security Requirements

- NFR.7: The system should encrypt all user data stored in the database using AES-256 encryption.
    - Testable: Inspect database storage to verify that data is encrypted according to the specified standard.
- NFR.8: The system should log out inactive users automatically after 30 minutes of inactivity to protect user privacy.
    - Testable: Simulate user inactivity and verify that the session is terminated after the specified time period.

5. Compatibility Requirements

- NFR.9: The system should be compatible with Android devices running version 10 and above.
    - Testable: Test the application on multiple Android versions (10, 11, 12, etc.) to verify compatibility.
- NFR.10: The system should function correctly on devices with a minimum of 2GB RAM and 200MB free storage.
    - Testable: Perform testing on devices that meet these specifications and verify the system's performance and functionality.

6. Maintainability Requirements

- NFR.11: The system should be modular, allowing for updates to the map interface without affecting the user authentication module.
    - Testable: Implement a change in the map interface and ensure the authentication module remains unaffected through regression testing.

6.3.1.Usability

Usability requirements for the software system include measurable effectiveness and satisfaction criteria in specific contexts of user interactions.

Search Functionality
UR.1: The system should enable users to find a building or campus resource through the search bar within 3 taps or fewer.
***Testable**: Conduct usability testing sessions where users perform searches, ensuring the task can be completed within the specified number of taps.*

Interactive Map Efficiency
UR.2: The interactive map should respond to user gestures (zoom, pan, tap) within 1 second to maintain a smooth user experience.
***Testable**: Measure the response time of the map when users interact with it under normal conditions.*

Locate Me Feature
*UR.3: The "Locate Me" button should accurately display the user's current location on the map 95% of the time. **Testable**: Test this feature in various locations across campus to verify that the user's position is displayed accurately at least 95% of the time.*

Route Planning Simplicity
UR.4: The system should allow users to plan a route from their current location to a chosen destination within 5 steps or fewer.
***Testable**: Perform user tests where users navigate to different locations, ensuring they can complete the task within the specified number of steps.*

Error Message Clarity
UR.5: The system should provide clear and understandable error messages for all common issues (e.g., GPS not available) within 2 seconds of the issue occurring.
***Testable**: Simulate error scenarios, such as turning off GPS, to verify that the appropriate error message appears promptly and is easy to understand.*

User Satisfaction Rating
UR.6: The system should achieve an average user satisfaction score of at least 4 out of 5 during usability testing sessions.
***Testable**: Collect user feedback through surveys after test sessions and calculate the average score based on user satisfaction with navigation, ease of use, and interface responsiveness.*

Onboarding and Help System
*UR.7: The system should offer an onboarding tutorial for first-time users that can be completed in 3 minutes or less. **Testable**: Measure the time it takes for new users to complete the tutorial, ensuring it stays within the specified time limit.*

6.3.2.Performance

Specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole.

- *Static numerical requirements (capacity) may include the following the number of simultaneous users to be supported; amount and type of information to be handled.*
- *Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.*

The performance requirements should be stated in measurable terms. For example, 95 % of the transactions shall be processed in less than 1 second.

1. Static Numerical Requirements (Capacity)

- ***PR.1**: The system should support up to 1,000 simultaneous users without any noticeable performance drop. **PR.2**: The system should manage up to 10,000 campus locations and 100,000 routes while keeping response times consistent.*

2. Dynamic Numerical Requirements

- ***PR.3**: The system should process 95% of location updates within 1 second when GPS is available. **PR.4**: The system should display calculated routes within 2 seconds after the destination is selected. **PR.5**: The system should handle up to 10,000 transactions per hour (including location updates, route calculations, and searches) without significant lag.*

3. Data Processing Requirements

- ***PR.6**: The system should retrieve cached map data in under 500 milliseconds for repeated requests. **PR.7**: The system should sync user preferences and saved routes with the server within 5 seconds.*

6.3.3.Reliability/Dependability

Specify the factors required to establish the required reliability/dependability of the software system at time of delivery.

System Uptime

- ***RD.1**: The system shall maintain an uptime of **99.5%**, ensuring continuous availability for users.*
  - *Testable: Monitor system logs over a period to verify that downtime does not exceed the specified limit.*

Error Handling and Recovery

RD.2*: The system should automatically recover from GPS signal loss and reconnect within 10 seconds when the signal is restored.*
Testable*: Simulate GPS signal loss and verify that the system reconnects and resumes functionality within the specified time frame.*

RD.3*: The system should log and report any critical errors, ensuring these logs are accessible for review and debugging.*
Testable*: Induce various error scenarios and check that errors are logged correctly.*

Data Integrity

RD.4*: The system should perform daily backups of user data and map information to prevent data loss in case of a failure.*
Testable*: Verify that the backup system is operational and that data can be restored accurately from the backups.*

Service Recovery Time

RD.5*: In the event of a system crash, the system should restore full functionality within 5 minutes to minimize user disruption.*
Testable*: Simulate a system failure and measure the time taken to restore the system, confirming that all functions are operational.*

6.3.4.Security

Specify the requirements to protect the software from accidental or malicious access, modification, or destruction. Specific requirements in this area could include the need to:

1) *Utilize certain cryptographic techniques;*
   - *certain cryptographic techniques will be utilized to keep students uploaded schedules protected*
   - *certain cryptographic techniques will be utilized to keep students searches protected*
   - *certain cryptographic techniques will be utilized to keep saved routes protected*
2) *Keep specific log or history data sets;*
   - *specific logs or history data sets of frequently visited or searched locations will be kept and protected*
   - *specific logs or history data sets of students course schedules will be kept and protected*
3) *Authenticate system users;*
   - *user authentication will be utilized to keep unauthenticated users out of protected information*
   - *CSUSM student login will be required to access saved information for students*
4) *Check data integrity for critical variables;*

- *use user data to find vulnerabilities in the software for revision and updates*

5) Assure data privacy.

- *we assure data shared between user and software is private*

6.3.5.Maintainability

Specify attributes of software that relate to the ease of maintenance of the software itself. These may include requirements for certain modularity, interfaces, or complexity limitations. Requirements should not be placed here just because they are thought to be good design practices.

Modular Design

***MT.1**: The system should be organized into separate modules for the user interface, navigation, and data management, allowing for independent updates.*
***Testable**: Verify that each module can be modified or tested without affecting other parts of the system.*

Code Readability

***MT.2**: The system should follow standard coding practices (e.g., consistent naming conventions) to enhance code readability for future developers.*
***Testable**: Perform code reviews to ensure consistency and clarity throughout the codebase.*

Version Control

***MT.3**: The system should use a version control system (e.g., Git) to manage changes and enable easy rollback to previous versions if needed.*
***Testable**: Test the version rollback capabilities by reverting recent changes to confirm proper operation.*

Basic Automated Testing

***MT.4**: The system should include basic automated tests for core functions to ensure that updates do not break existing features.*
***Testable**: Verify that automated tests run and pass successfully after system updates.*