**Project Name:  Location Sharing System**

**Project Description:**

Design a system to share the location of users between known people. Location can be of home, office, place where he/she usually goes. This system is useful for people who shift from one location to another. It can also be useful for carpooling, flat renting, hanging out, to know recommended service providers, etc. Different teams may consider different usage of this system.

**Contents**

1. **Software Requirements**
2. **User Requirements**
3. **Functional Requirements (Features)**
4. **Non Functional Requirements**
5. **Use Case Diagram**
6. **Software Process Model**

**Software Requirements**

* The solution shall locate the user's position with utmost accuracy.
* The system must be resistant to spoofing and interference.
* The solution shall be available indoors.
* The system must offer on-request positioning.
* The geographical coverage of the solution must be global.
* The functioning of the system must be flawless and elegant.
* The solution shall offer the users an opportunity to provide timely feedback to make the user experience seamless.
* The solution should be voice assisted to help navigation.
* The user’s location should be kept private and sharing must be end-to-end encrypted.
* The solution must be available in multiple languages.

**User Requirements**

* Location Sharing among friends within the app. Create groups to share functions and related chat.
* Emergency SOS message that shares location instantly to your chosen friends by pressing a single button.
* Assign frequently visited locations like home, office and store them for easy sharing.
* Allow sharing recommended places to visit by friends if they have already visited that city for better experience of the visitor.
* In case a group of friends wants to meetup, the app provides shortlisted locations suitable to all members such that they are approximately equidistant.

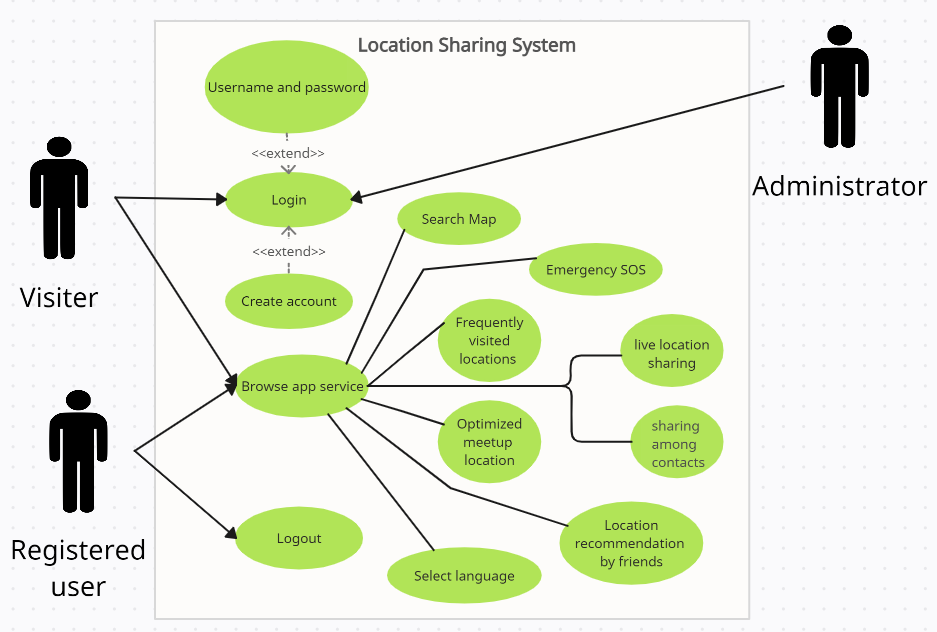
**Functional Requirements (or Features)**

* The user should be able to access his own location on the system and keep a track of it.
* The user should be able to access the contacts and friends with whom he/she wants to share their location with.
* The user should be able to use any message service of his/her choice to send his location.
* The user should be able to assign frequently visited locations like home, office, etc. and store them for easy sharing and tracking.
* The user should be able get rent options or rent his/her own property.
* The user should be able to get recommended locations or popular places to visit if he is new to any place.
* The user should be able to use an emergency SOS message to share location instantly to chosen friends.
* Allow sharing recommended places to visit by friends.
* Providing an equidistant meetup location for a group of people.

**Non-Functional Requirements**

* Security
* Privacy
* Accuracy
* Maintainability
* Reliability
* Scalability
* Performance
* Flexibility

**Use Case Diagram**



**Software Process Model**

The Process model we are using here for this system will be the “Iterative Model”.

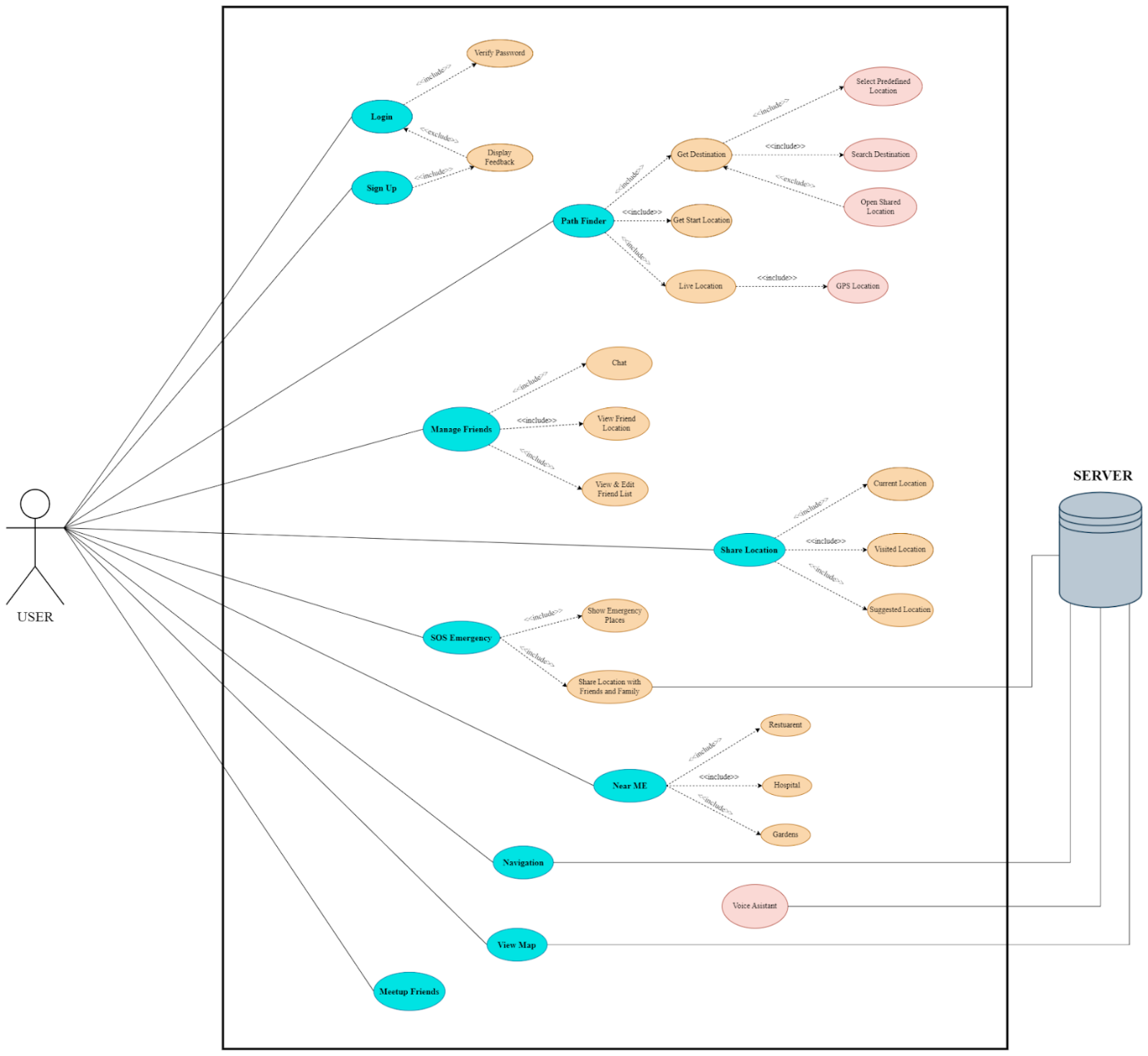
This is because firstly the requirements of our system are fixed and they won’t change. We can keep adding more features as we like but the user requirements will still remain fixed.

Secondly, we can’t get utmost accuracy in the very first attempt of developing the application. So what we can do here is that we implement the application, then take the user feedback and then use that to improve the features in the next deployment of the application.

**Project Name:  Location Sharing System**

**Contents:**

1. **Use Case Diagram**
2. **Use Case Description**
3. **Textual Description for each Use Case**
4. **Non Functional Requirements and their justifications**

**Use Case Diagram:**

**Textual Description for each Use Case:**

In this use case diagram, the user interacts with the system to perform various actions, such as signing into the system, searching locations in map, sharing their live or static location, receiving location from others, sending and receiving an emergency SOS message and finding the most appropriate location for a meetup. The system is responsible for processing these request and providing the necessary functionality to the users:

|  |  |
| --- | --- |
| **Use case** | Log in |
| **Description** | Verify login credentials from the user to access services. |
| **Actors** | User, location sharing system |
| **Precondition** | 1. Users must be connected to the internet. 2. Input interface should be available to the user. |
| **Post conditions** | 1. Users should be able to access the services. |
| **Flow** | 1. User enters his credentials. 2. The system verifies the credentials with the database. 3. The user is allowed to access the system if the credentials are verified. |
| **Alternate flow 1** | In step 3, if the credentials are not verified,   1. The system asks the user to re enter credentials. 2. System offers an option of “forget password” where users can edit their current password by providing alternate methods of verification. |

|  |  |
| --- | --- |
| **Use case** | Search location |
| **Description** | The user will be shown a map based interface of his/her nearby locations to browse for different places to visit. |
| **Actors** | User, location sharing system |
| **Precondition** | 1. Users must be connected to the internet. |
| **Post conditions** | 1. Users should be able to access the services. 2. Guest users should only be able to access search maps. |
| **Flow** | 1. User locates the search bar. 2. Enter the name/coordinates of the place of visit. 3. The user is allowed to manipulate the map to explore more options. |

|  |  |
| --- | --- |
| **Use case** | Manage friend |
| **Description** | The user will be provided with a functionality to chat and add people to his/her friend list. |
| **Actors** | User, location sharing system |
| **Precondition** | 1. Users must be connected to the internet. |
| **Post conditions** | 1. Users should be able to access the services. 2. Guest users should only be able to access search maps. |
| **Flow** | 1. User clicks on “Add people” functionality. 2. Select people from his/her contact list or by specifying the email address to add them to the user's friend list. 3. The recipient will be notified about the new connection. 4. The system provides the user with the functionality to text people present in his/her friend list and share their location. |

|  |  |
| --- | --- |
| **Use case** | Share live location |
| **Description** | The user will be able to share his/her live location with the selected recipient. |
| **Actors** | User, location sharing system |
| **Precondition** | 1. Users must be connected to the internet. 2. Input interface should be available to the user. |
| **Post conditions** | 1. Users should be able to access the services. 2. Guest users should only be able to access search maps. |
| **Flow** | 1. User clicks on “share live location” functionality. 2. Select the recipient to share the location with, from the user’s edit list. 3. Click on the ”share” button. |

|  |  |
| --- | --- |
| **Use case** | Emergency SOS message |
| **Description** | It’s a safety feature that allows the user to activate a SOS button that the user can press during an emergency to notify their chosen recipients (friends or family). |
| **Actors** | User, location sharing system, recipients |
| **Precondition** | 1. Users must be connected to the internet. |
| **Post conditions** | 1. A button must be activated on the homescreen of the user’s device. |
| **Flow** | 1. User activates the emergency SOS message feature from the application. 2. User selects the recipients from the list of friends to whom he may want to notify. Users can also have a default list, so that he doesn’t have to select the recipients everytime he activates the feature. 3. System activates and displays a SOS button on the homescreen of the user’s device. 4. If the user presses the button, the system sends an alert notification and the user’s live location to all the recipients. |
| **Alternate flow 1** | 1. Users can also select an expected time he is anticipating to reach his destination. In case the user is unable to reach the destination 1 hour past the anticipated time, the SOS message is automatically sent to the recipients. |

|  |  |
| --- | --- |
| **Use case** | Meetup with friends |
| **Description** | A group of users can find a common meeting point that is suitable to all the users in the group. |
| **Actors** | Multiple Users, location sharing system |
| **Precondition** | 1. Users must be connected to the internet. 2. All the users must be in a group. 3. The location sharing service should be activated by all the users of the group. |
| **Post conditions** | 1. The system should provide a list of places for meeting to all the users. |
| **Flow** | 1. A user from the group selects the option “meetup with friends”. 2. The system takes the locations of all the users in the group and provides a list of recreational places that are suitable to all the users. |

|  |  |
| --- | --- |
| **Use case** | Nearby me |
| **Description** | The system displays nearby recreational places like restaurants, parks, cinemas, etc. to the user. This also includes the places that are recommended by the user's friends. |
| **Actors** | Multiple Users, location sharing system |
| **Precondition** | 1. Users must be connected to the internet. 2. The user’s location sharing feature must be activated. |
| **Post conditions** | 1. The system should provide the locations of the suggested places on the user’s map. |
| **Flow** | 1. User activates the “nearby me” feature in the application. 2. The system takes the locations of the user and displays all the nearby recreational places that are recommended by his friends. |

**Non Functional Requirements and their justifications:**

**Security:** The location sharing system must have high levels of security so that  unauthorized users can’t access user location information. This feature is quite  crucial for securing user privacy and preventing data leakages.

**Performance:** The system should be efficient enough to handle enormous amounts of location data at the same time.  As a result of this multiple users would be able to access and share the location quickly.

**Reliability:** The system should be capable enough to provide guarantee of reliability that users can access and share their location data at any time & also it should be completely secure.The system should be robust to system failures and there should be minimal downtime.

**Scalability:** System should be able to accommodate the growing user base and the enormous location data. This will make sure that the system will be able to accommodate the user base’s requirements as it grows.

**Usability:** The system should have an easy and user-friendly interface. As a result of which users won’t be facing challenges while using the user interface.

**Compatibility:** The system should be made to operate smoothly with a variety of hardware, software, and operating systems. By doing this, users would be able to access and share their location information from any device or platform of their choice.

**Availability:** Users must always have access to the system, with little to no downtime. Users will always be able to access and share their location information.

**Accessibility:** Any user should be able to access and share the location data regardless of their ability. Even a person having hearing or vision impairments or any other disabilities should be able to utilize the system.

**Project Name:  Location Sharing System**

**Contents:**

1. **Tools, Technologies and Frameworks**
2. **Effort of Project**

**Tools, Technologies and Frameworks**

**Tools : Android Studio, mapbox / Google maps API**

**Technology : Android, iOS**

**Frameworks : Flutter**

**Android studio:**

Android Studio is the official Integrated Development Environment (IDE) for Android and App development. It is developed and maintained by Google. So there are many resources available for it. It includes a large community of developers so that one can get help whenever required. It also contains a wide range of plugins and supports different types of languages like Java, Kotlin, C++, etc. It also contains an emulator so one can run and debug their application.

**Mapbox :**

Mapbox API is for custom maps. Mapbox is a platform for designing and delivering custom maps, geocoding and routing API’s and location-based services. It provides developers with a suite of tools for building applications that visualize, analyze, and interact with geographic data. It also supports various languages and frameworks. It provides SDKs for both android and iOS.

**Google maps API :**

Google Maps API is a service provided by Google that allows developers to integrate maps and location-based features into their applications. The API includes a range of tools and services for creating and customizing interactive maps, geocoding, routing, and location-based services. It also supports various languages and provides SDKs for both android and iOS. It is widely used by developers and has quite good documentations.

**Flutter :**

It is an open-source app development framework made by Google. It is easy to  build high-performance, visually attractive, and natively compiled applications using it. It supports all mobile, web, and desktop platforms from a single codebase. Flutter has a widget-based architecture, which allows developers to create complex user interfaces easily. It uses the Dart programming language to add logic in your

application.

**Data Base : MongoDB**

MongoDB is a NoSQL database that can handle large volumes of geospatial data. It offers geospatial indexing and querying capabilities, allowing for fast retrieval of location-based data. MongoDB can also be easily integrated with various mapping libraries and frameworks. It also includes built-in authentication service for secure access. It also supports storing data on the cloud.

**Effort of Project**

**Step 1 : Calculate Unadjusted Use-Case Points**

* **Determine Unadjusted Use-Case Weight**

|  |  |  |
| --- | --- | --- |
| **Use case** | **Transactions** | **Complexity** |
| Log in | 3 | Simple |
| Search location | 3 | Simple |
| Manage friend | 4 | Average |
| Share live location | 3 | Simple |
| Emergency SOS | 4 | Average |
| Meetup with friends | 2 | Simple |
| Nearby me | 2 | Simple |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use-Case Complexity** | **Use-Case Weight** | **Number of Use-Cases** | **Product** |
| Simple | 5 | 5 | 25 |
| Average | 10 | 2 | 20 |
| Complex | 15 | 0 | 0 |
| **Unadjusted Use case weight (UUCW)** | | | **45** |

* **Determine Unadjusted Actor Weight**

|  |  |
| --- | --- |
| **Actors** | **Complexity** |
| Users | Complex |
| Developer | Average |
| System | Simple |

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor Complexity** | **Actor Weight** | **Number of Actors** | **Product** |
| Simple | 1 | 1 | 1 |
| Average | 2 | 1 | 2 |
| Complex | 3 | 1 | 3 |
| **Unadjusted Actor Weight (UAW)** | | | **6** |

* **Calculate Unadjusted Use-Case Points**

**Unadjusted Use-Case Points (UUCP) = UUCW + UAW = 45 + 6 = 51**

**Step 2: Adjust For Technical Complexity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Factor** | **Description** | **Weight (W)** | **Rated Value (0 to 5) (RV)** | **Impact (I = W × RV)** |
| **T1** | Distributed System | 2.0 | 0 | 0 |
| **T2** | Response time or throughput performance objectives | 1.0 | 4 | 4 |
| **T3** | End user efficiency | 1.0 | 4 | 4 |
| **T4** | Complex internal processing | 1.0 | 2 | 2 |
| **T5** | Code must be reusable | 1.0 | 3 | 3 |
| **T6** | Easy to install | .5 | 4 | 2 |
| **T7** | Easy to use | .5 | 5 | 2.5 |
| **T8** | Portable | 2.0 | 3 | 6 |
| **T9** | Easy to change | 1.0 | 3 | 3 |
| **T10** | Concurrent | 1.0 | 2 | 2 |
| **T11** | Includes special security objectives | 1.0 | 3 | 3 |
| **T12** | Provides direct access for third parties | 1.0 | 1 | 1 |
| **T13** | Special user training facilities are required | 1.0 | 0 | 0 |
| **Total Technical Factor (TFactor)** | | | | **32.5** |

**TCF = 0.6 + (TF / 100)**

**=  0.6 + (32.5 / 100)**

**=  0.925**

**Step 3: Adjust For Environmental Complexity**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Factor** | **Description** | **Weight (W)** | **Rated Value (0 to 5) (RV)** | **Impact (I = W × RV)** |
| **F1** | Familiar with the project model that is used | 1.5 | 4 | 6 |
| **F2** | Application experience | .5 | 4 | 2 |
| **F3** | Object-oriented experience | 1.0 | 3 | 3 |
| **F4** | Lead analyst capability | .5 | 5 | 2.5 |
| **F5** | Motivation | 1.0 | 0 | 0 |
| **F6** | Stable requirements | 2.0 | 2 | 4 |
| **F7** | Part-time staff | -1.0 | 0 | 0 |
| **F8** | Difficult programming language | -1.0 | 4 | -4 |
| **Total Environment Factor (EFactor)** | | | | **13.5** |

**Calculate the Environmental Factor (EF) = 1.4 + (-0.03 × EFactor)**

**= 1.4 + (-0.03 × 13.5)**

**= 0.995**

**Step 4: Calculate Adjusted Use-Case Points (UCP)**

**Calculate Adjusted Use-Case Points (UCP)  = UUCP × TCF × EF**

**= 51 x 0.925 x 0.995**

**= 46.93**

**Here, We take 6 man hours per UCP**

**Estimated Efforts = UCP × Hours / UCP**

**= 46.93 × 6**

**= 281.58 hours**

**Here, We consider 40 man hours per week.**

**Thus, Estimated week will be around 7.**

**Project Name:  Location Sharing System**

**Contents:**

1. **Domain Analysis Model**
2. **Boundary, Entity and Control Object**
3. **Sequence and Class Diagram**
4. **High Level System Design**

**Domain Analysis Model:**

A domain analysis model for the “Location Sharing System” would typically include the following elements:

* **User Roles:** There are different roles for a user in a system such as end-users,administrators, moderators, etc . Each of them having different permissions and functionalities.
* **Location Data**: The system would define the type of location data that it will collect and process, such as GPS coordinates, street addresses, or landmarks.
* **Privacy and Security:** The system would provide privacy and security to the users such as data encryption, user authentication and access control.
* **Communication Protocols:** The system should define the communication protocols that it will use to receive, store and retrieve location data, such as RESTful APIs or MQTT.
* **Integration:** The model should define the integration points. The system will use an integration of systems such as mapping services, social networks, Emergency SOS, etc.
* **User Interface:** The model should define the user interface that the system will provide to end-users, i.e. a mobile application here, and the features that will be available, such as real-time location tracking, notifications, alerts, etc.
* **Performance and Scalability:** The model should define the performance and scalability requirements of the system, such as response times, data throughput, and user concurrency.
* **Data Analytics:** The model should define the data analytics capabilities of the system, such as location-based insights, user behavior, and trends.

Overall, the domain analysis model for a location sharing system should be comprehensive and cover all the important aspects of the system, from user roles and privacy to performance and analytics.

**Identify Boundary, Entity and Control Object:**

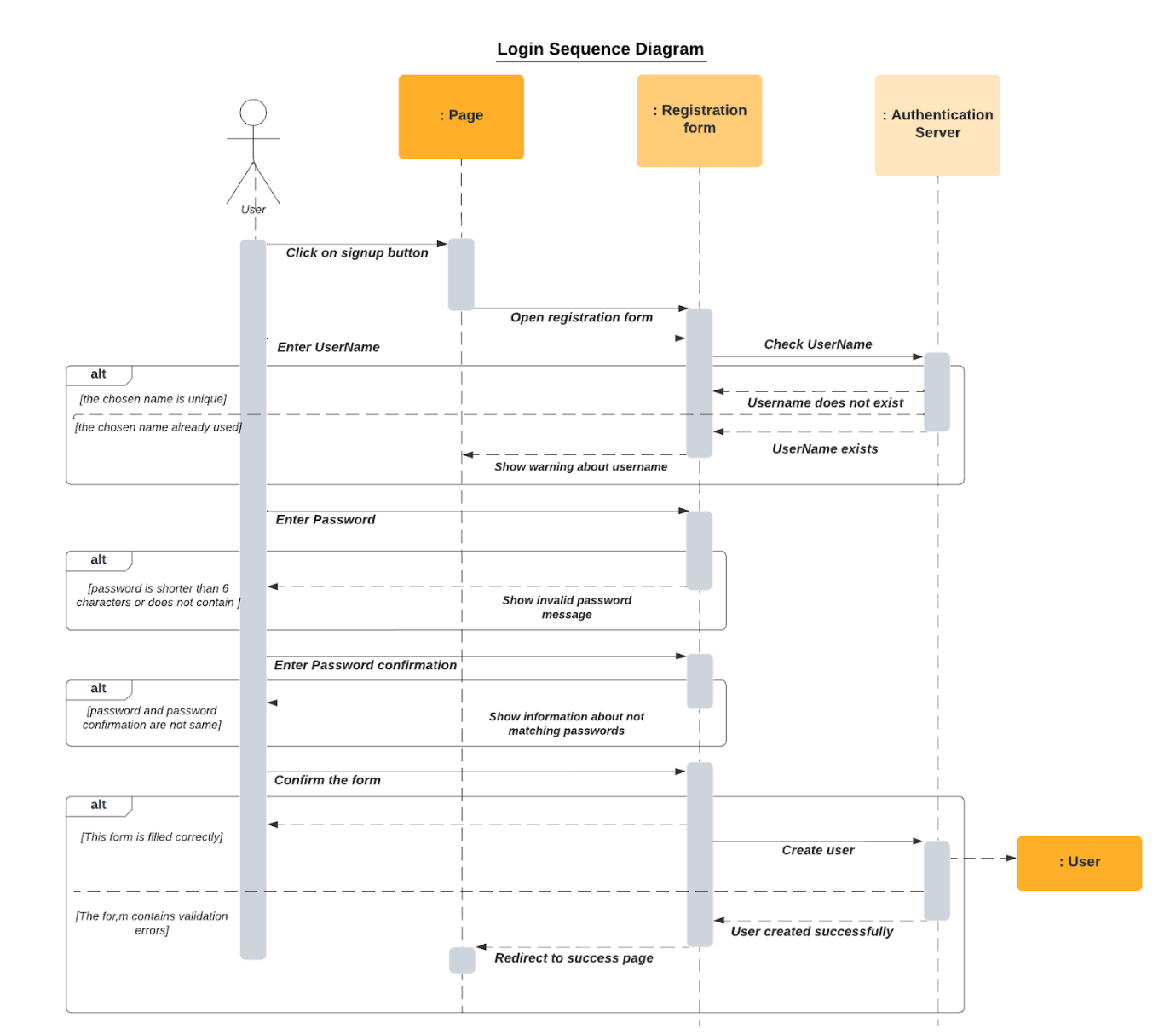
1. **Boundary Object:**
   1. Boundary for the application would be the interface of mobile application which separates the system and the user.
   2. It would include different features in the application such as friends meetup, navigation, live location sharing etc.

1. **Entity Object:**
   1. There are several entities associated with the application. This includes databases of saved locations, restaurants, hotel, sos places.
   2. It would also include database of friends and family, database of sos contacts and api of google maps
   3. Additionally the entities would include the hardware and software supporting the system.

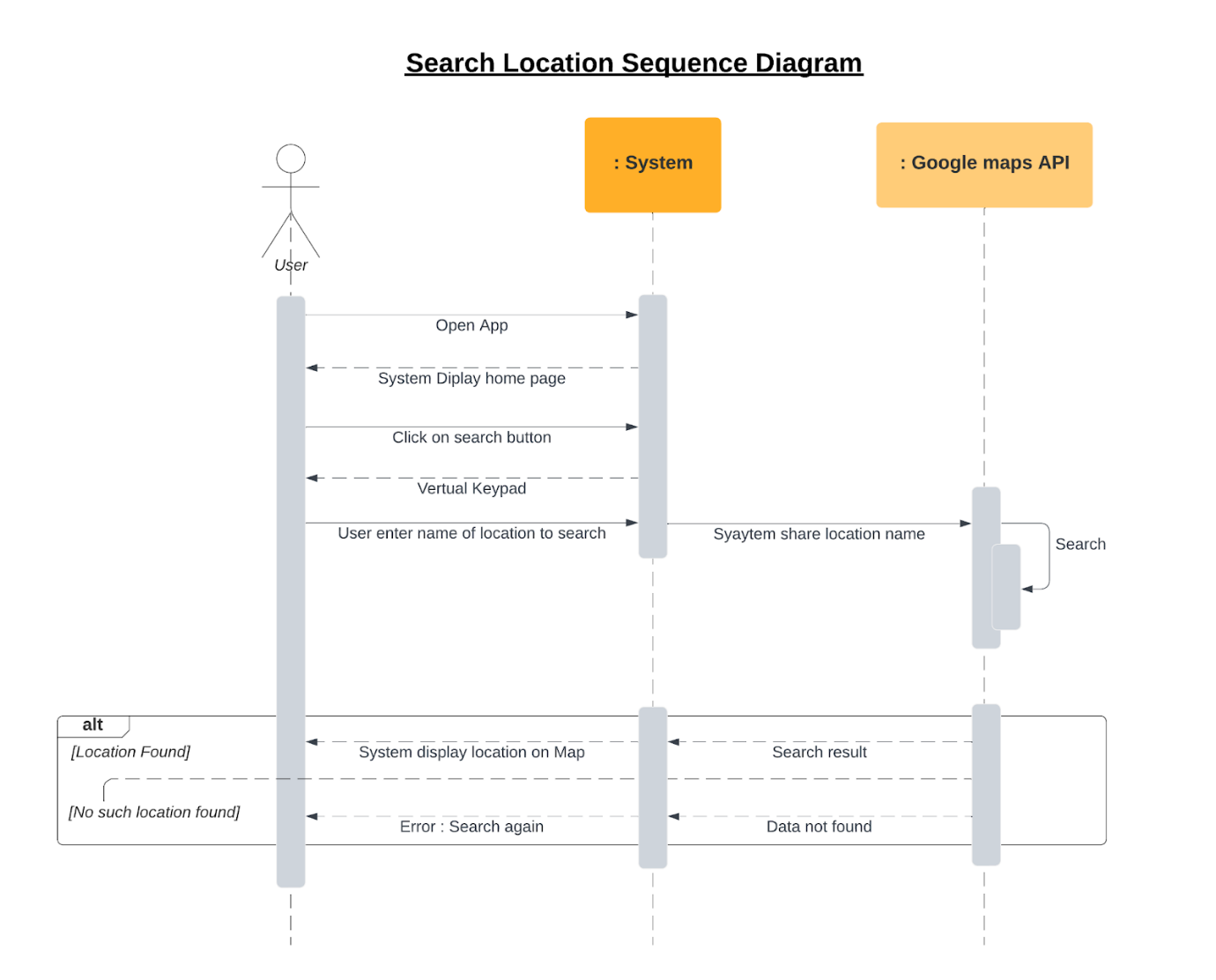
1. **Control Object:**
   1. Control objects would be the options and features that generate data or control the database features.
   2. Database for friends and family contacts will be accessible through phones contacts.

**Sequence Diagram:**

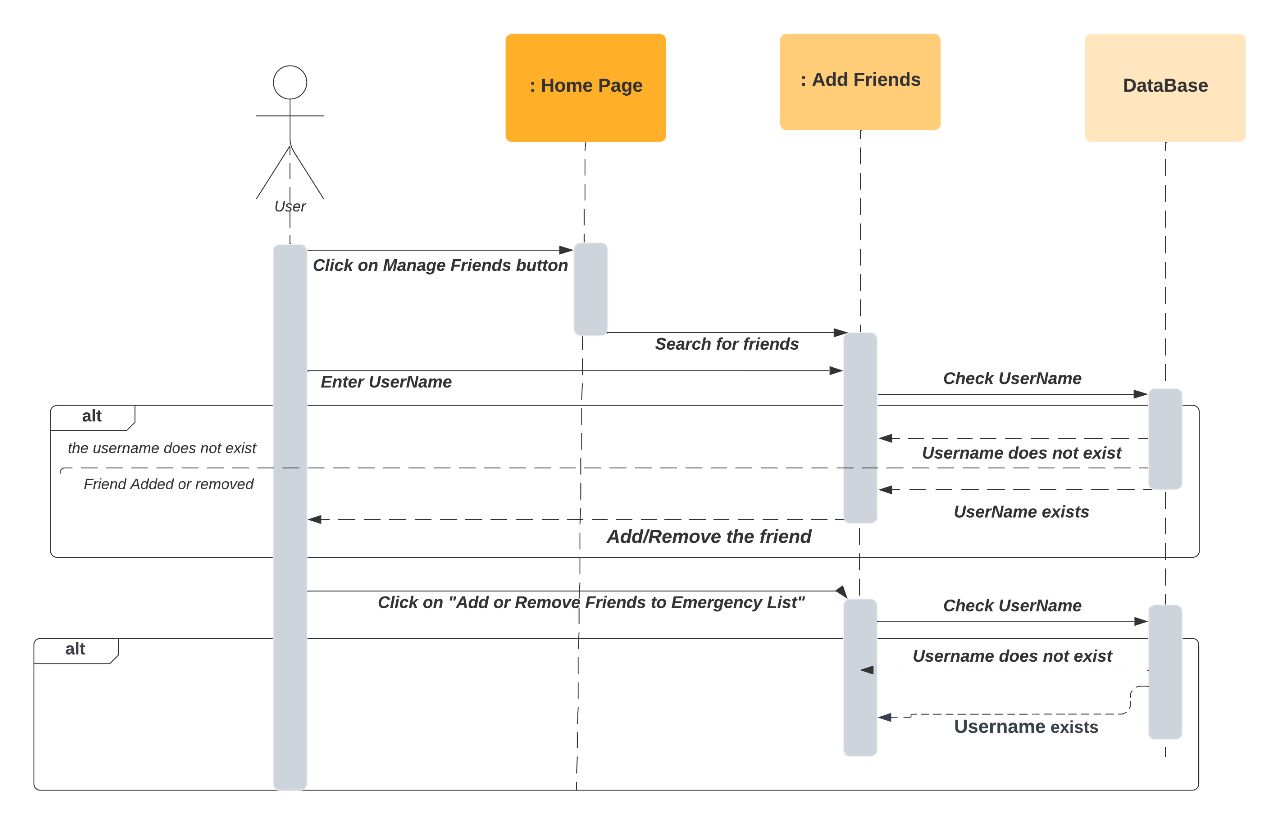
1. **Login**



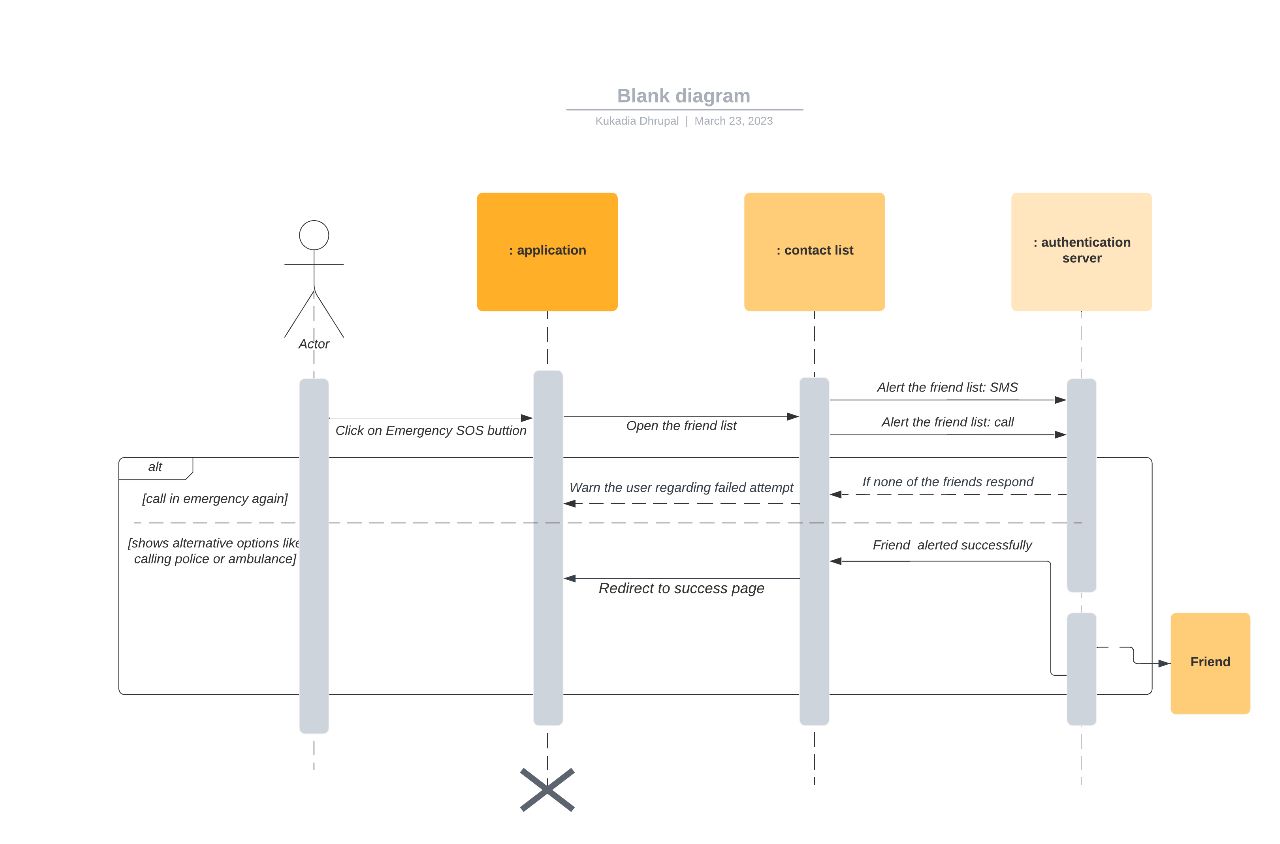
1. **Search Location**



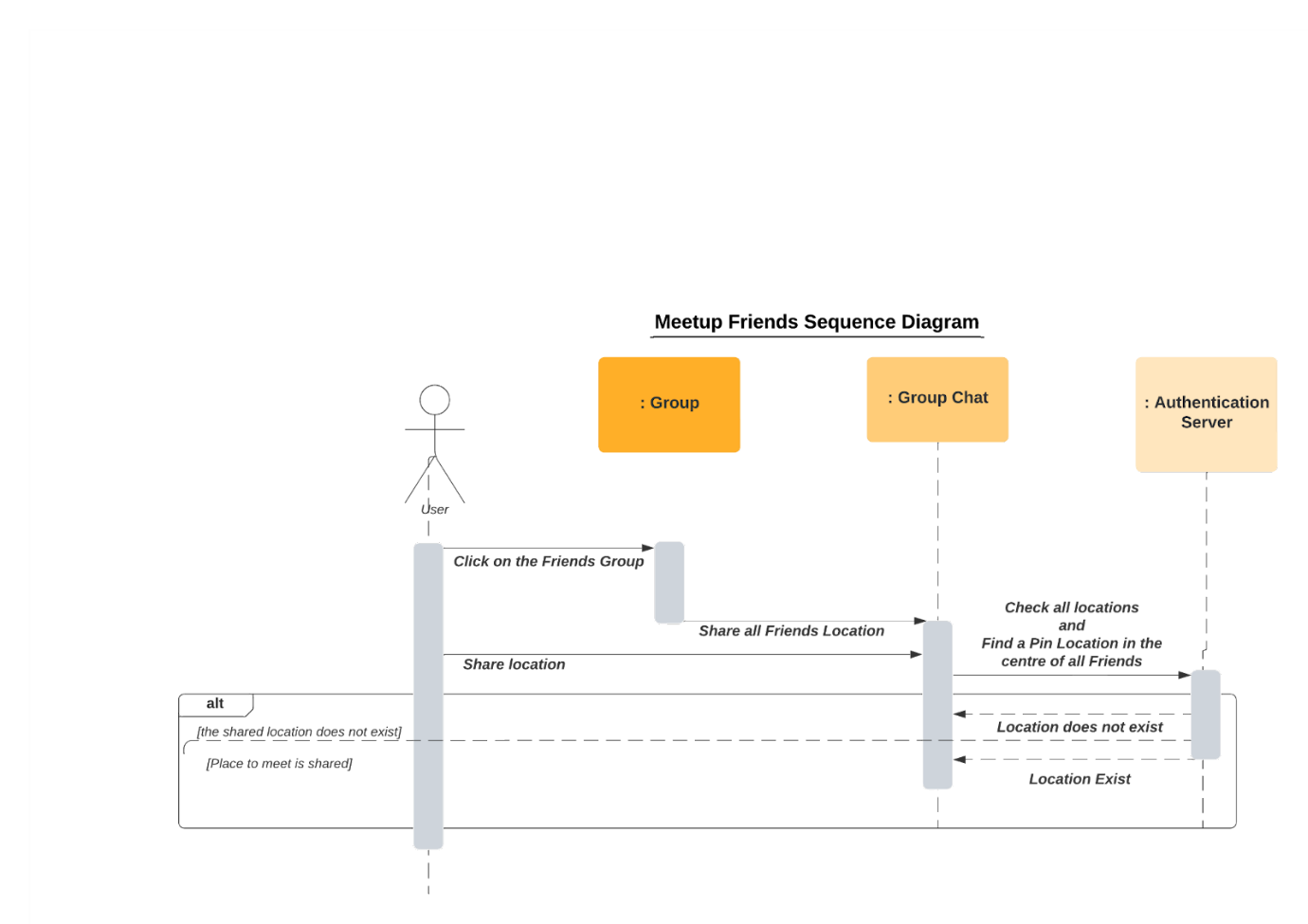
1. **Manage Friend**

****

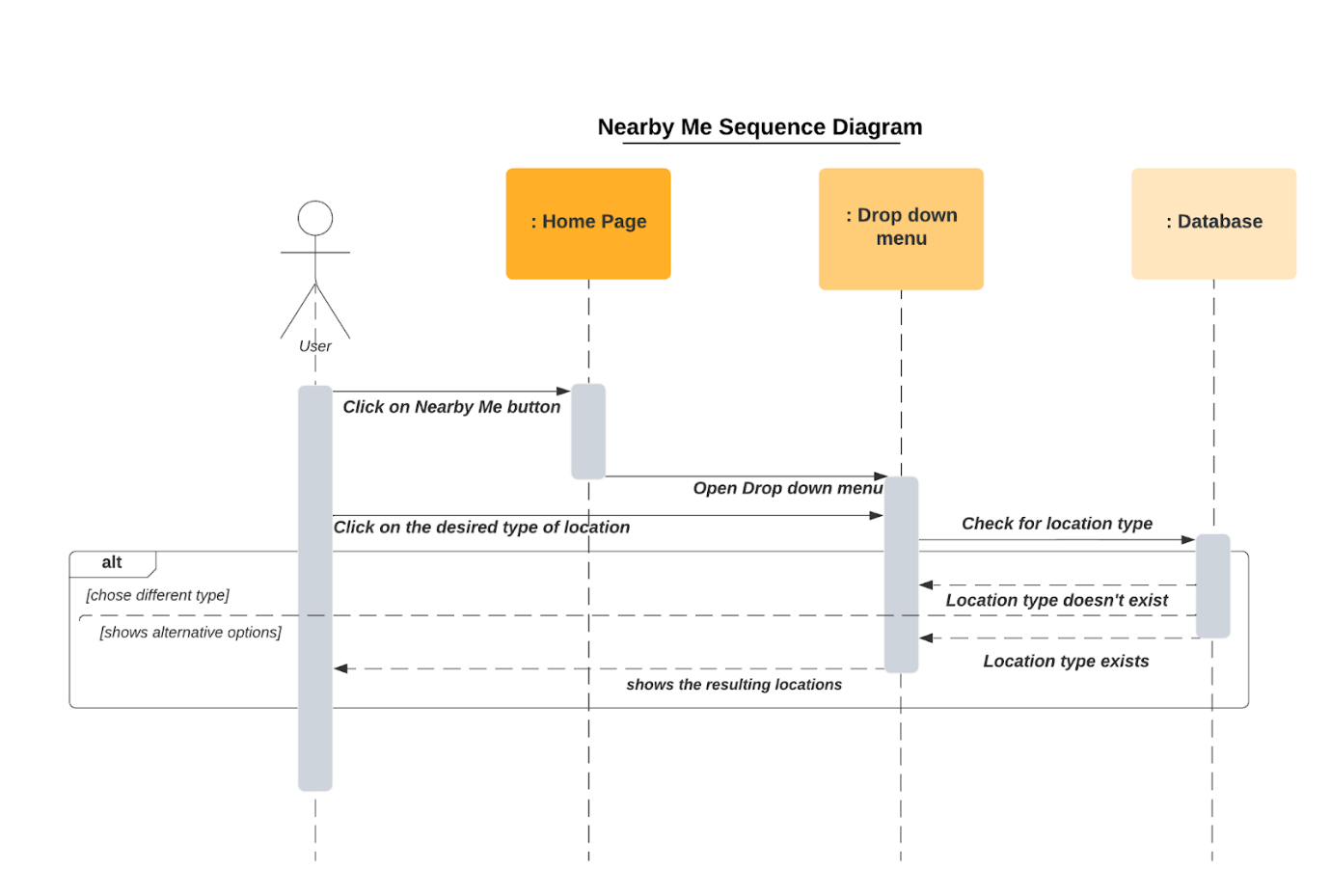
1. **Emergency SOS**

****

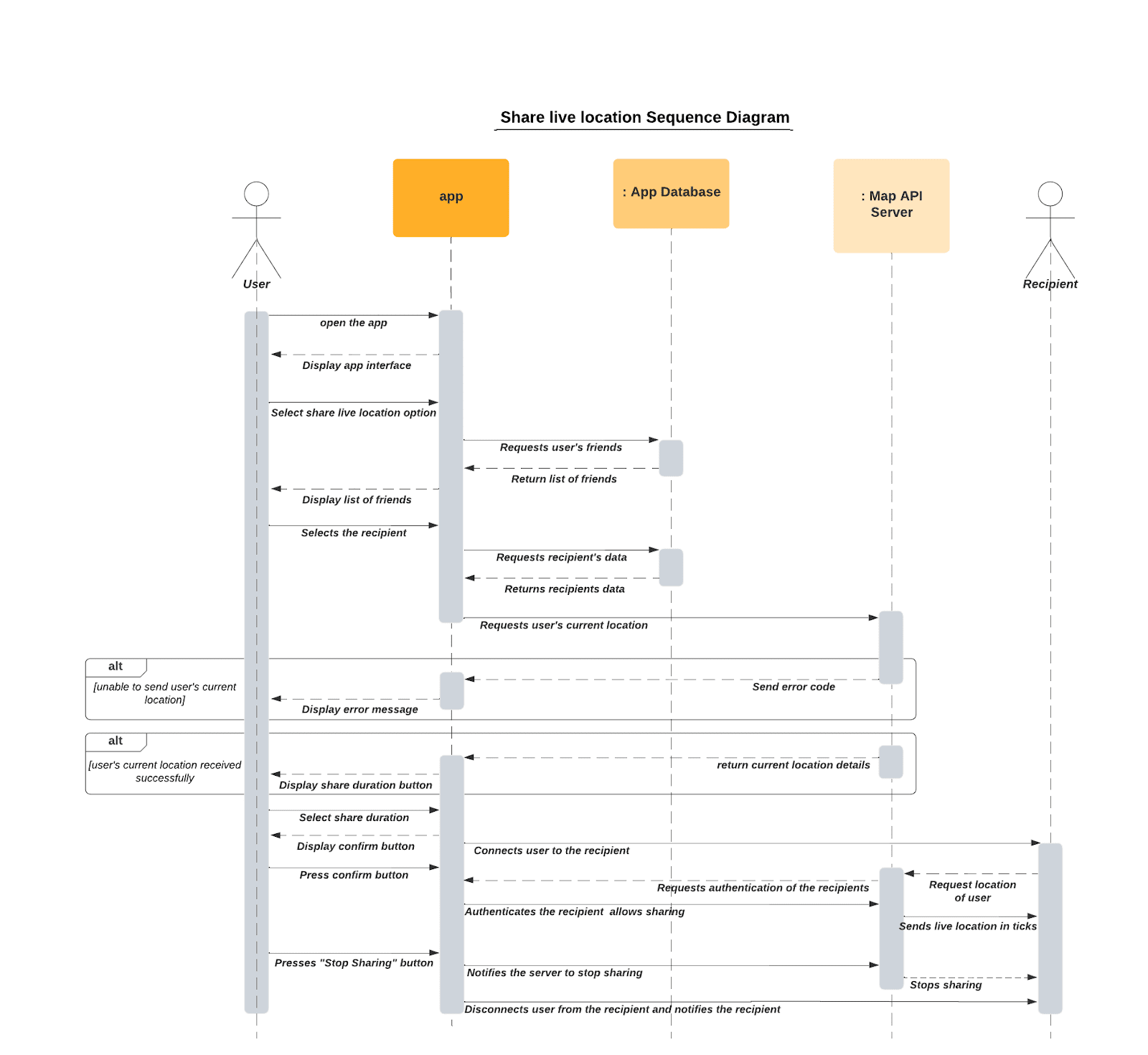
1. **Meetup Friends**

****

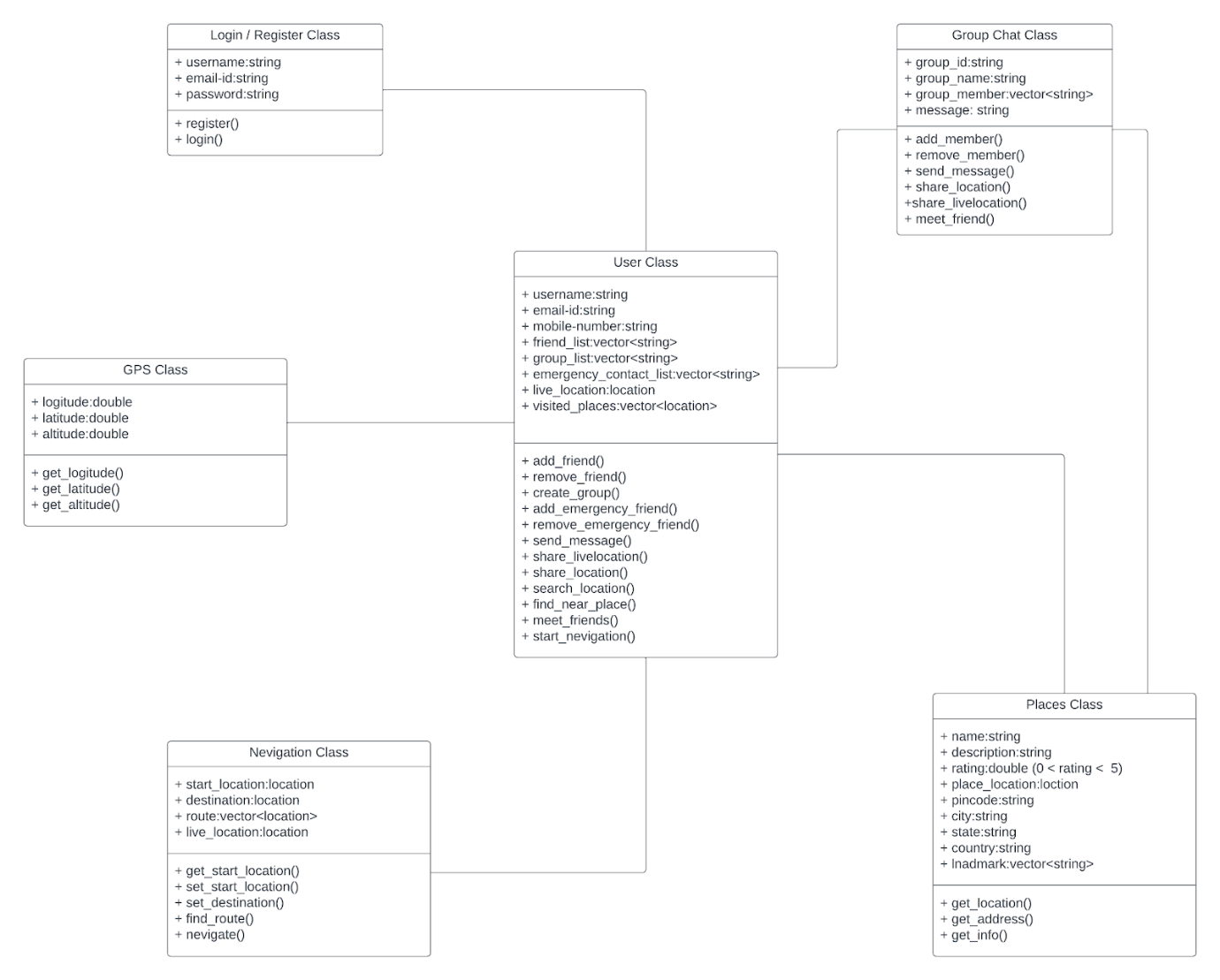
1. **Nearby me**



1. **Share Live Location**

****

**Class Diagram**



**High Level System Design**

**Architecture:**

A client-server architecture would be suitable for this project. The client would be the mobile application that users interact with, and the server would be a centralized location where user data is stored and processed. The client application would send location updates and other data to the server, and the server would process this data and send it back to the appropriate users. The communication between the client and server would be through RESTful API or similar web services. The use of cloud-based technologies such as AWS or Google Cloud Platform could provide the necessary resources and scalability for the project. Overall, this high-level design provides a solid foundation for a location sharing project that is scalable, secure, and user-friendly.

This architecture consists of a presentation tier, an application tier, and a data storage tier.

**Presentation Tier:**

It is responsible for handling user interface and user input. This tier includes the following subsystems:

* **User Interface:** This subsystem includes the app interface that allow users to interact with the app.
* **Client-side Logic:** This subsystem includes the Dart and other client-side code that handles user input validation and other user interactions.

**Application Tier:**

The application tier is responsible for handling the business logic and processing user input. This tier includes the following subsystems:

* **Controller:** This subsystem is responsible for managing the flow of data between the user interface and the data storage tier.
* **Business Logic:** This subsystem is responsible for implementing the business logic of the app, such as user registration, friends list, and managing sos.
* **Security:** This subsystem is responsible for implementing security features, such as user authentication and authorization.

**Data Storage Tier:**

The data storage tier is responsible for storing and managing data. This tier includes the following subsystems:

* **Data Access Layer:** This subsystem is responsible for accessing and retrieving data from the database.
* **Database:** This subsystem is responsible for storing all of the website's data, such as user information, forum data, topic data, post data, tag data, and ranking data.
* **Security:** The system should be secure and protect sensitive employee data. Access to the system should be restricted based on user roles and permissions. The system should also be backed up regularly to prevent data loss in case of a system failure.
* **Integration:** The system should be able to integrate with other HR systems and tools, such as applicant tracking systems, performance management software, and benefits administration platforms.

**Subsystems:**

1. **User Interface (UI) subsystem:**

This subsystem would consist of the user-facing components of the application, including the login and registration screens, the home screen where users can select contacts to share their location with, and the settings screen where users can customize their preferences. The UI subsystem would be responsible for handling user input and presenting information in a clear and intuitive manner.

1. **Location Tracking subsystem:**

This subsystem would be responsible for tracking the user's location using GPS or other location-based technologies. The subsystem would need to be efficient and accurate, as well as capable of handling location data from multiple users. The location data would be stored in a database and presented to the user in a meaningful way, such as on a map or in a list of recent locations.

1. **Communication subsystem:**

This subsystem would be responsible for enabling communication between users in the application. It would handle sending and receiving location updates and other messages between users. The communication subsystem would need to be fast and reliable, as well as secure to prevent unauthorized access to user data.

1. **Database subsystem:**

This subsystem would be responsible for storing and managing user data, including location data, contact information, and preferences. The database would need to be scalable to handle large amounts of data and support fast retrieval and updating of data. The data would need to be secured using encryption and other security measures to prevent unauthorized access.

1. **Security subsystem:**

This subsystem would be responsible for ensuring the security and privacy of user data. It would include measures such as authentication and encryption to prevent unauthorized access to user data. The security subsystem would need to be constantly updated and monitored to protect against new security threats.

A location-sharing system is an Android application allowing users to share their real-time locations with others. This Android application uses GPS technology to determine a user's location and share it with others through social media or a messaging system. This application allows users to create groups and share their location with their family and friends. This application can be accessed through a variety of devices such as smartphones, tablets, and laptops.

**Benefits**

* Increased safety and security : For example, parents may use this application to keep track of their children's whereabouts. Similar to this, relatives and friends can use location sharing to keep tabs on each other while travelling or participating in outdoor activities.
* Navigation purposes When traveling to unfamiliar places by sharing their location, individuals can easily guide others to their destination or receive assistance from others when needed.
* Recommendations : This application suggests nearby places, such as restaurants, ATMs, hotels, hospitals, etc., to users. This application also has an emergency SOS feature that offers emergency service, allowing users to call for help quickly in case of an emergency.

location-sharing system provide a path to stay connected and share their location with others. In summary, this android application allows individuals to share their real-time location with others. These systems offer a range of benefits, including increased safety and security, navigation assistance. However, users should be mindful of privacy and security concerns when using this application, and only share their location with trusted individuals.

**Tools and Technologies:**

**Flutter :** Flutter is Google's free, open-source software development kit (SDK) for cross-platform mobile application development. Using a single platform-agnostic codebase, Flutter helps developers build high-performance, scalable applications with attractive and functional user interfaces for Android or IOS. Dart is the programming language used to code Flutter apps.

**MongoDB -** MongoDB is an open-source document-oriented database that is designed to store a large scale of data and also allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables.

**Node.js -** Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

**Implementation:**

* 1. When you’re creating an app, you might want to create a login function. We have use Flutter as the app, Node.js as a backend, and at least MongoDB for storing user info.
  2. Map API is loaded in the database. Mongodb is connected to Node.js server. And then it is loaded from Mongodb using Node.js when signed in.
  3. Now sign in and sign up page are implemented in flutter.
  4. When the app is launched, it will show a splash screen and then Signin page is displayed. Splash screen is to be merged with login page. A splash screen is a graphical control element consisting of a window containing an image, a logo, and the current version of the software.
  5. When signed in, it will display the basic map structure. So, Login page is merged with map homepage.
  6. Some functionalities are to be implemeneted in map home page. These are represented using buttons.
* Navigation
* SOS
* ShareLiveLocation
* Contacts
* Friends

Map homepage is merged with all these functionalities.