CS/SE-3354 Software Engineering

Final Project Deliverable 1

Where2Next

Group Members: Oluwadara Morakinyo, Charlie Alpert, Lennon Crow, Ariel Ong, Nuria Habib, John Kasperbauer, Joseph Farghal, Manh Ngo

**Final Project Draft Description, Professor feedback and Response to Feedback**

Project Description:

Our project, “where2next,” is a google maps extension that allows the user to easily look for new and exciting outings and restaurants through an adjustable radius feature. With filters for certain user-nominated tags like “peaceful,” “vegan,” or “adrenaline-inducing,” the user can find activities that will match the intended atmosphere of the outing. For example, if the user likes moving and socializing, the user can use the filter for tags like “lively” and “active” to get recommendations. Similar tags are used for food-oriented activities as well. Unlike google maps, where2next keeps a chronological log of the past locations you’ve been to and what you did there. This feature is especially handy if you’re prone to forgetting what you last ordered and whether you liked it or not.

Professor Feedback:

Interesting and promising to be a useful topic.

In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how.

Fair delegation of tasks.

Please share this feedback with your group members.

You are good to go. Have fun with the project and hope everyone enjoys the collaboration.

Response:

Similar applications to our Where2Next software include Google Maps and Yelp. Since Where2Next is an extension of Google Maps, it works with it. Where2Next is different from Google Maps and Yelp specifically because of its tagging system and user history tracking. Our tagging system is a feature to help businesses expose their products or services and categorizes the businesses to help our users better find what they are looking for. Some tag examples include “birthday,” “relaxing,” or “gluten-free.” Additionally, Where2Next keeps the user’s history and ratings to help find more places similar to what they enjoy and avoid places where they previously had a bad experience with.

**Project Scope**

1.1 Search

1.1.1 Search Bar: User Types what they want

1.1.2 Tag Pop Ups: Tag options are given the user can click when searching

Ex: "lively" "quiet" "classy"

1.1.3 General Option Buttons: Buttons User can click for broad genres

Ex: "restaurants" "gyms"

1.2 Location

1.2.1 Set User Current or Desired Location

1.2.2 Set Distance Radius

1.2.3 Distance Slider from 0-50 miles

1.3 History

1.3.1 Store user past activity

1.3.2 Activity User Info: Rating on Enjoyment, Option to add tags, Things bought/ordered at activity

1.3.3 Allow user to make activity public

1.4 Create Activity / Location

1.4.1 Add public event

1.4.2 Temporary events do not require approval

1.4.3 Permanant Activities require admin approval

Ex: new restaurant added

**URL to Group GitHub**

<https://github.com/josephfarghal/3354-where2next.git>

**Delegation of tasks**

Charlie Alpert

* Attached the Final Project draft description while also addressing the instructor's feedback.
* Made a commit on the repository titled “project scope” with the descriptions of what our project would do.

Joseph Farghal

* Set up the GitHub repository while also providing the Layered Architecture pattern for our project.

Oluwadara Morakinyo

* Made the first commit to the repository with the README file as well as created the functional and non-functional requirements.

John Kasperbauer

* Created the use case diagram for the project.

Manh Ngo

* Created the sequence diagram for each case use.

Nuria Habib

* Created the class diagram showing the cardinalities and relationship types between the classes.
* Nuria also made sure to represent the attributes and methods of each class as well.

Ariel Ong

* Wrote the delegation of tasks and the justification behind choosing the spiral model.

Lennon Crow

* Kept the group on schedule and led group meetings while also reviewing finished portions of the deliverable.

**Software Process**

Spiral Model

* We will be employing spiral model because “where2next” has multiple features and heavily relies on user interaction, it is necessary to come up with multiple prototypes before slowly perfecting the product.
* The spiral model offers the perfect division for this.
* We chose the spiral model over the prototyping evolutionary model because our product deals with a lot of data.
* As developers, we have a responsibility to protect our user’s privacy, hence with each prototype created, risk must be analyzed to ensure data is not vulnerable.
* The spiral model allows developers to move cyclically, repeating some steps to ensure we are able to meet requirements while developing a secure product that our customer is comfortable using.
* The cycle of determining objectives and constraints before evaluating and resolving risk in order to develop product before planning the next phase offers an organic feel to the development process.
* If we are to use the spiral model in conjunction of beta testing, the product should be able to satisfy user needs in a secure manner.

**Software Requirements**

Functional Requirements

1. The user shall be able to search for activities based on adjectives such as “lively” (filter) or specific key-words (vegan, epilepsy-friendly)
2. The user shall be able to set the distance radius of the results they receive from their current location.
3. The application shall keep a chronological log of past activities the user has attended.

4. The application shall keep a note for each past activity which includes information on what the user did while there.

- For every past activity done, user should be able to append adjectives from application bank to describe the activity

5. The application shall have a secondary mode that enables users to add a public event/activity location.

6. The user shall be able to share activities from the application to external applications (e.g. social media)

Non-functional Requirements

[Product Requirements]

1. After completing the 1-minute application walk-through video, users should be able to identify the buttons for specific tasks

(e.g., searching a location, accessing activity logs) within 30 seconds.

* This requirement is derived from the usability requirement leaf from figure 4.3

1. Each user transaction should take no more than 5 mseconds

* This requirement is derived from the performance requirements leaf from figure 4.3

1. Each user will be allocated 10 GB in data to store location history and additional notes

* If 10GB is passed, then a paywall will appear.
* This requirement is derived from the space requirement leaf from figure 4.3

1. The application should not have more than three, 30 minute down-times within a 30-day period.

* Every downtime must be addressed and closed within 30 minutes.
* This requirement is derived from the dependability requirement leaf from figure 4.3

1. To protect against fake/spam accounts, the user will have to use some sort of third-party authentication to verify their identity.

* This requirement is derived from the security requirement leaf derived from the product requirement node from figure 4.3.

[Organizational Requirements]

1. The application must be run on a physical computer device that is between 32 to 95 degrees Fahrenheit in temperature for the specified product requirements to be met.

* This requirement is derived from the environmental requirement leaf from figure 4.3.

1. The application must be supported on Android and IOS operating systems per Google Maps available usage.

* This requirement is derived from the operational requirements leaf from figure 4.3.

1. Application must be written in a secure and reliable language.

* Secure in the sense that the developers will prioritize language security over memory efficientness.
* This requirement is derived from the development requirements derived from figure 4.3

[External Requirements]

1. Updates on software should be done during the hours servers receive the least traffic and should take no more than 6 hours

* This requirement is derived from the regulatory requirements leaf from figure 4.3.

1. The application will screen out activities that contain illegal actions and/or hateful/discriminatory language.

(e.g., drugs, prostitution, white-supremacist groups)

* This requirement is derived from the ethical requirements leaf from figure 4.3.

1. Application profits must be reviewed every fiscal quarter and discrepancies in profits and operational costs are to be addressed within 72 hours.

* This requirement is derived from the accounting requirement leaf from figure 4.3.

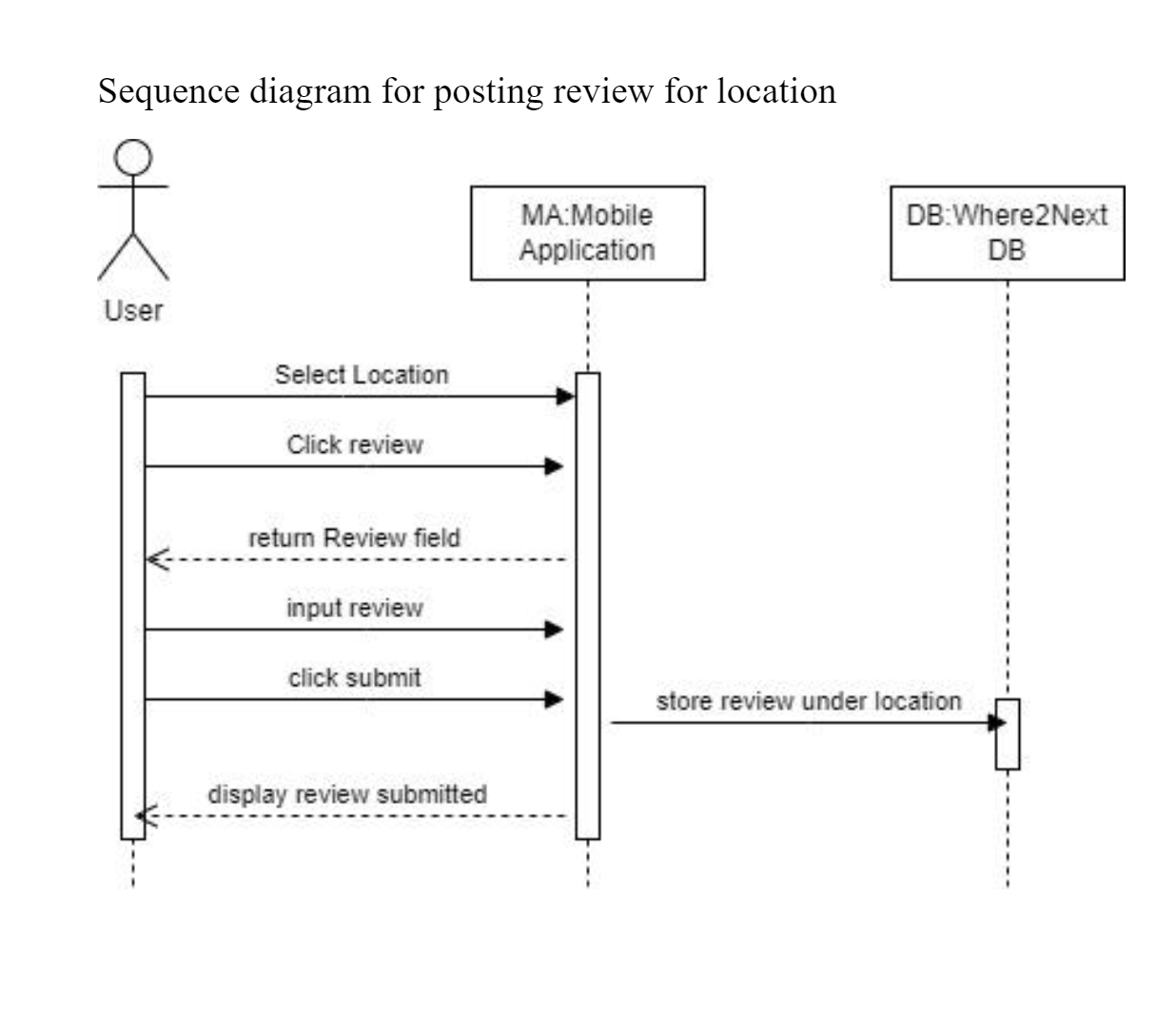
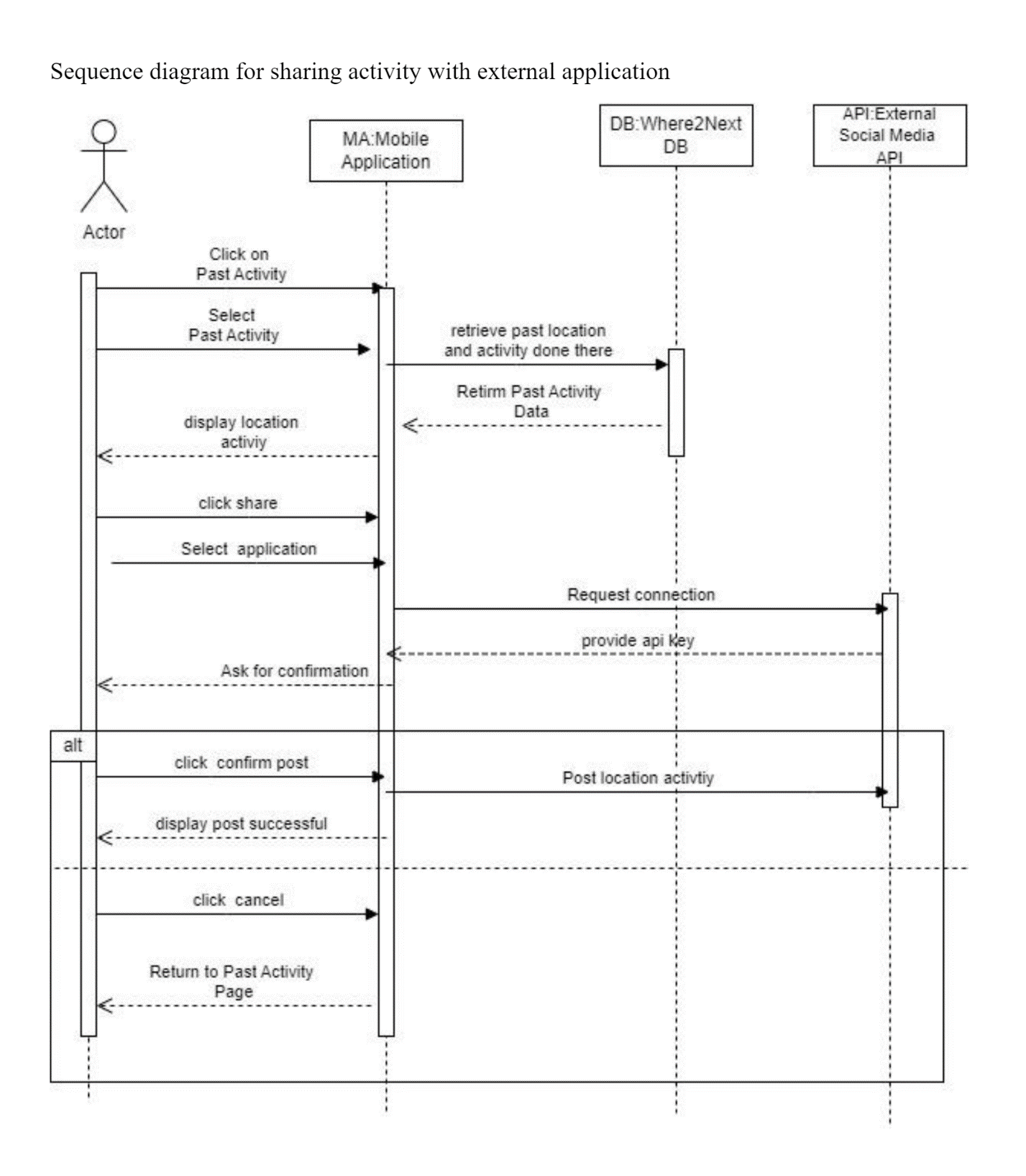
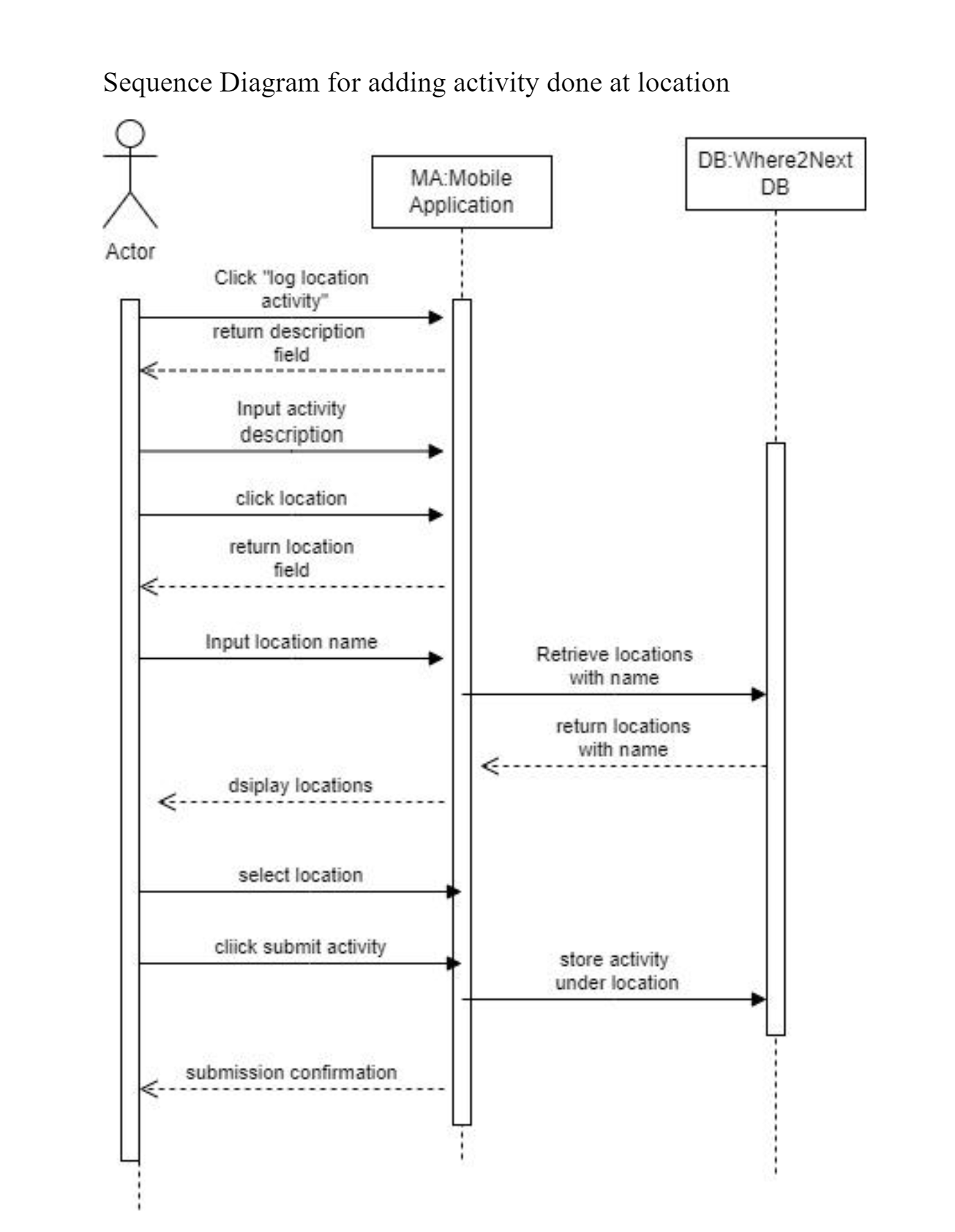
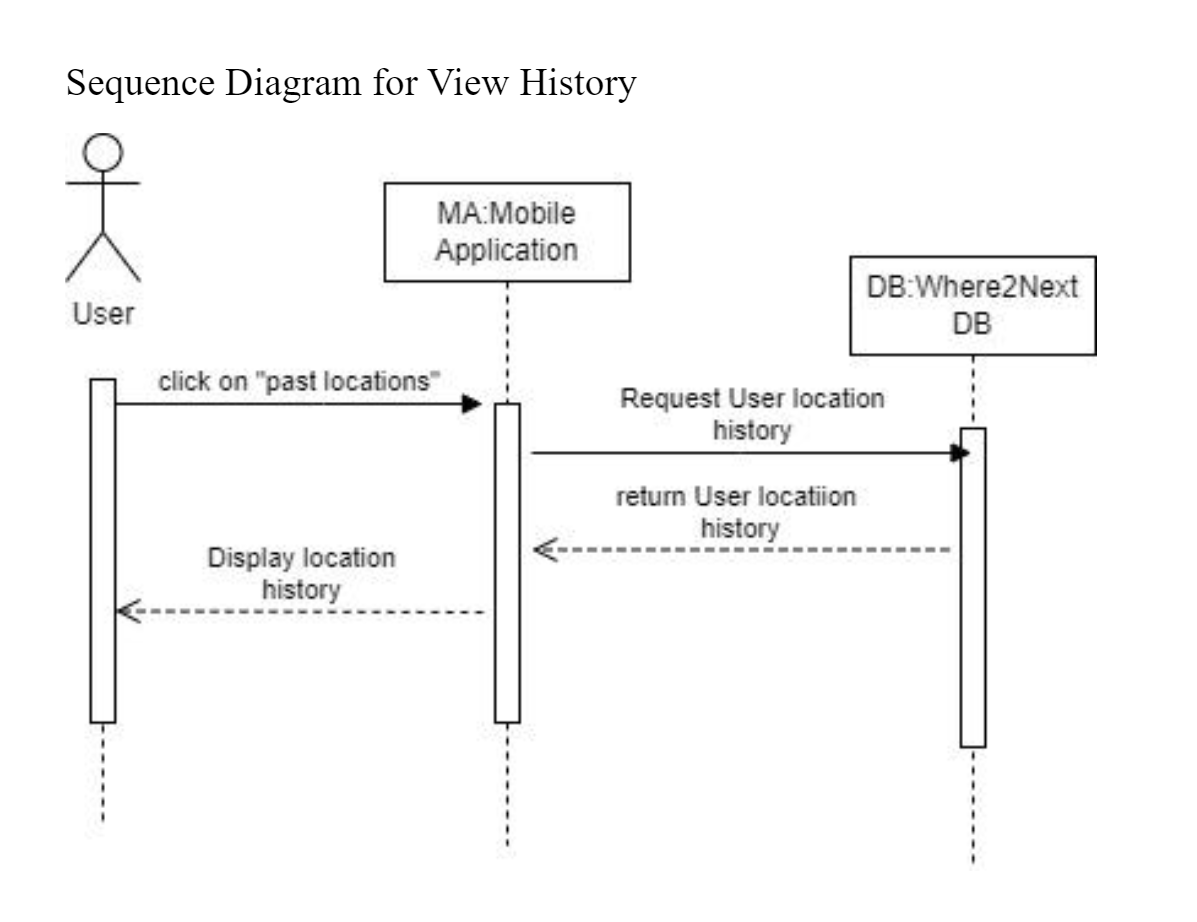
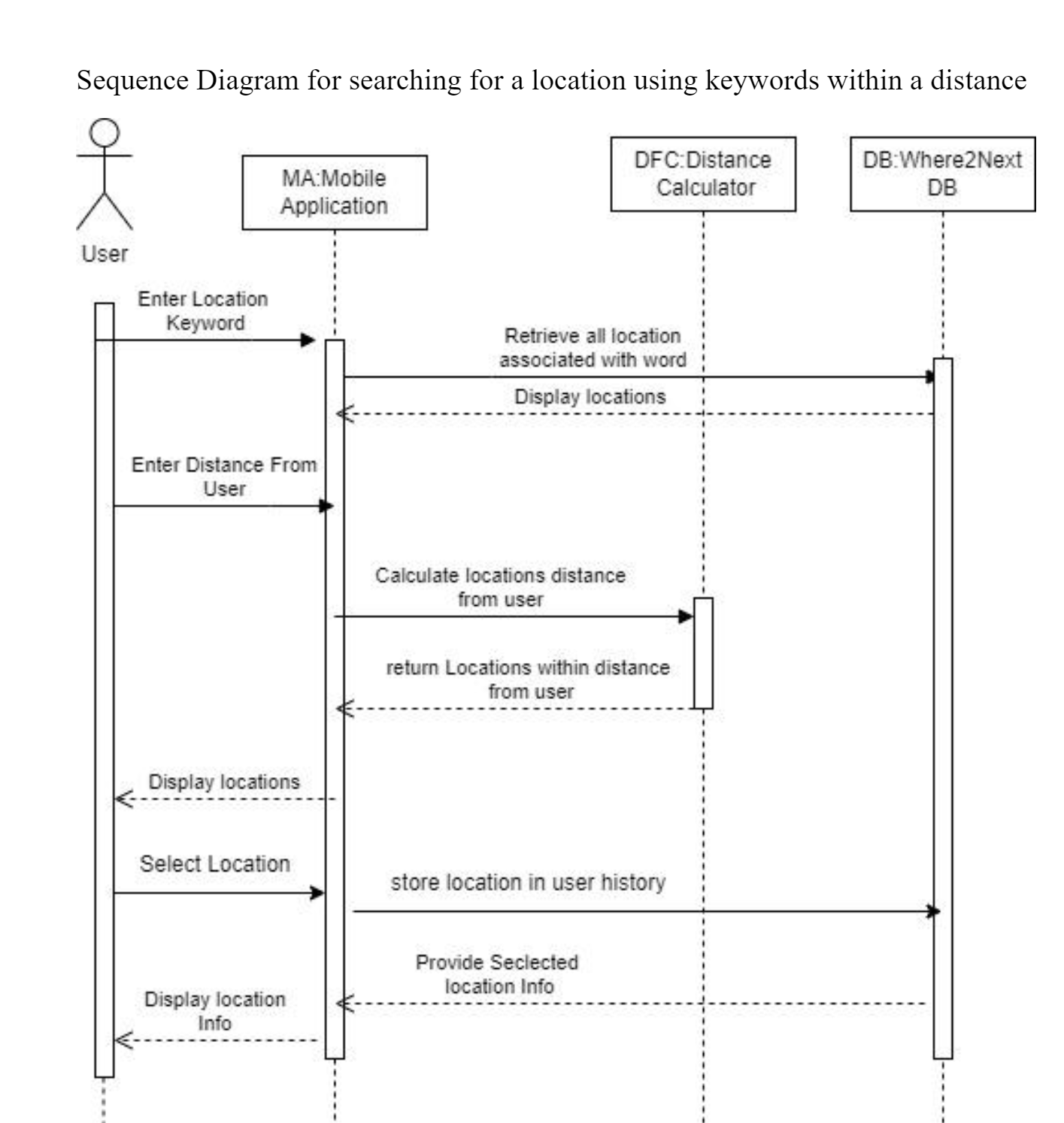
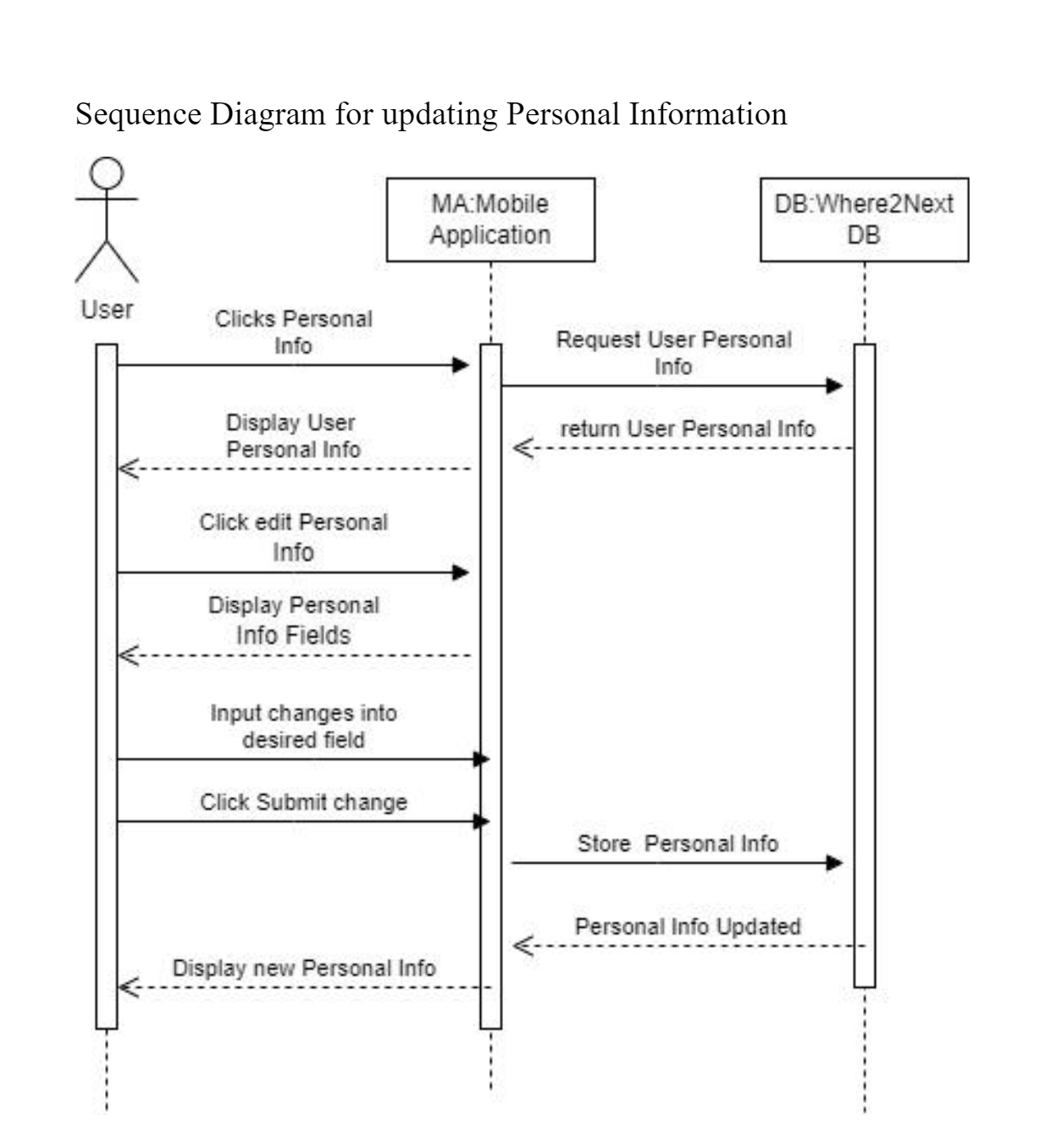
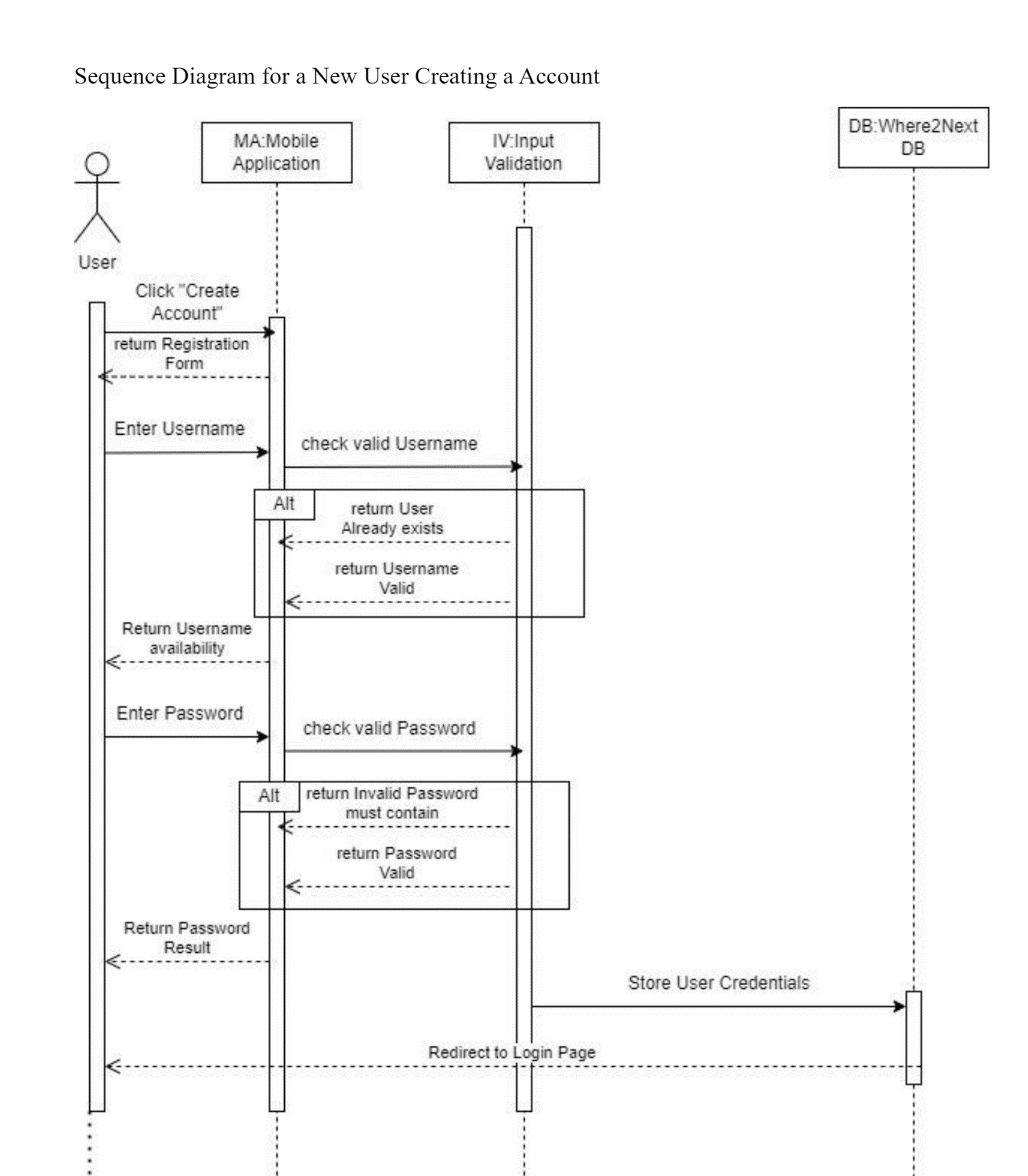
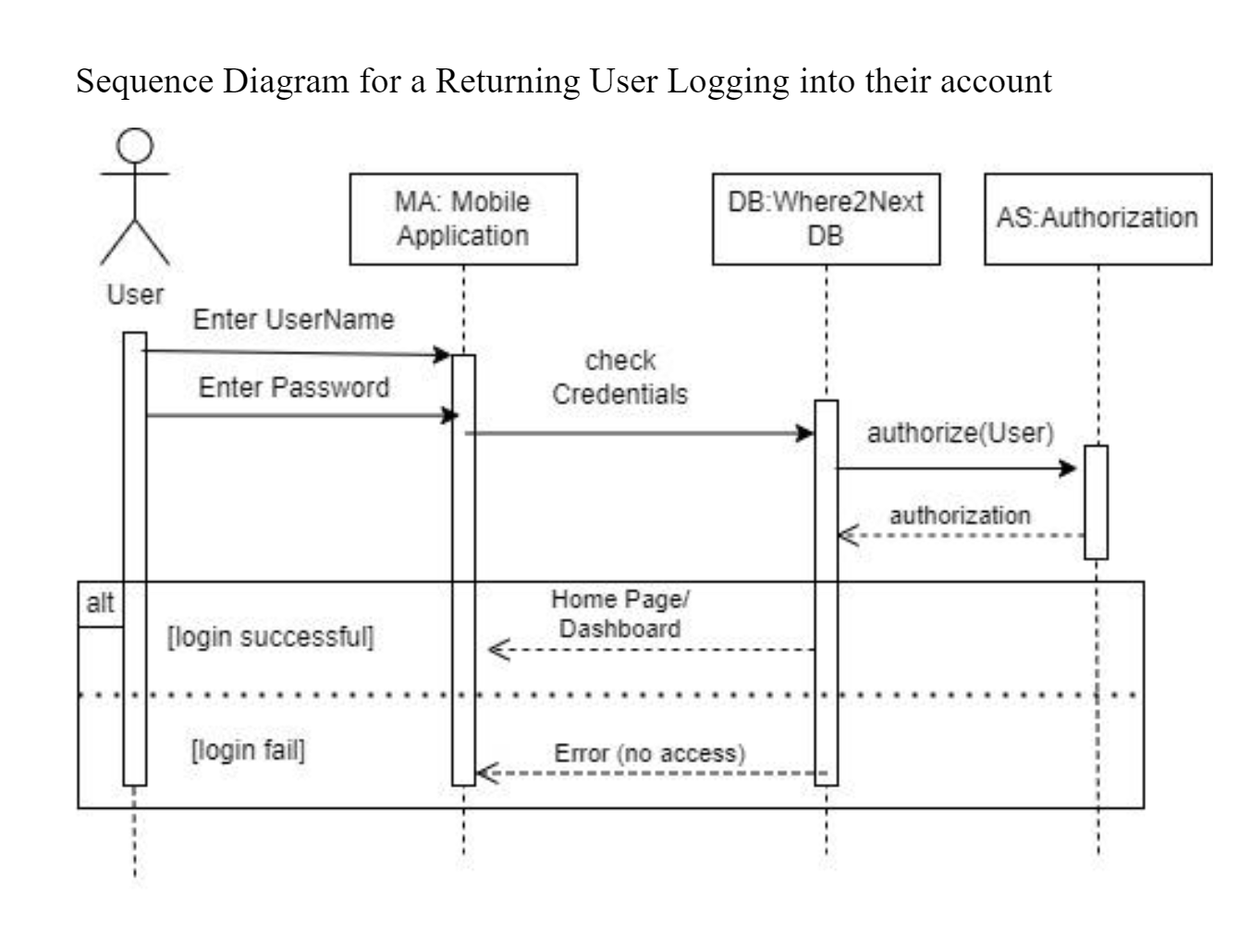
1. This app will not sell user data to third parties. Data will only be used to observe user trends and for the recommendation system.

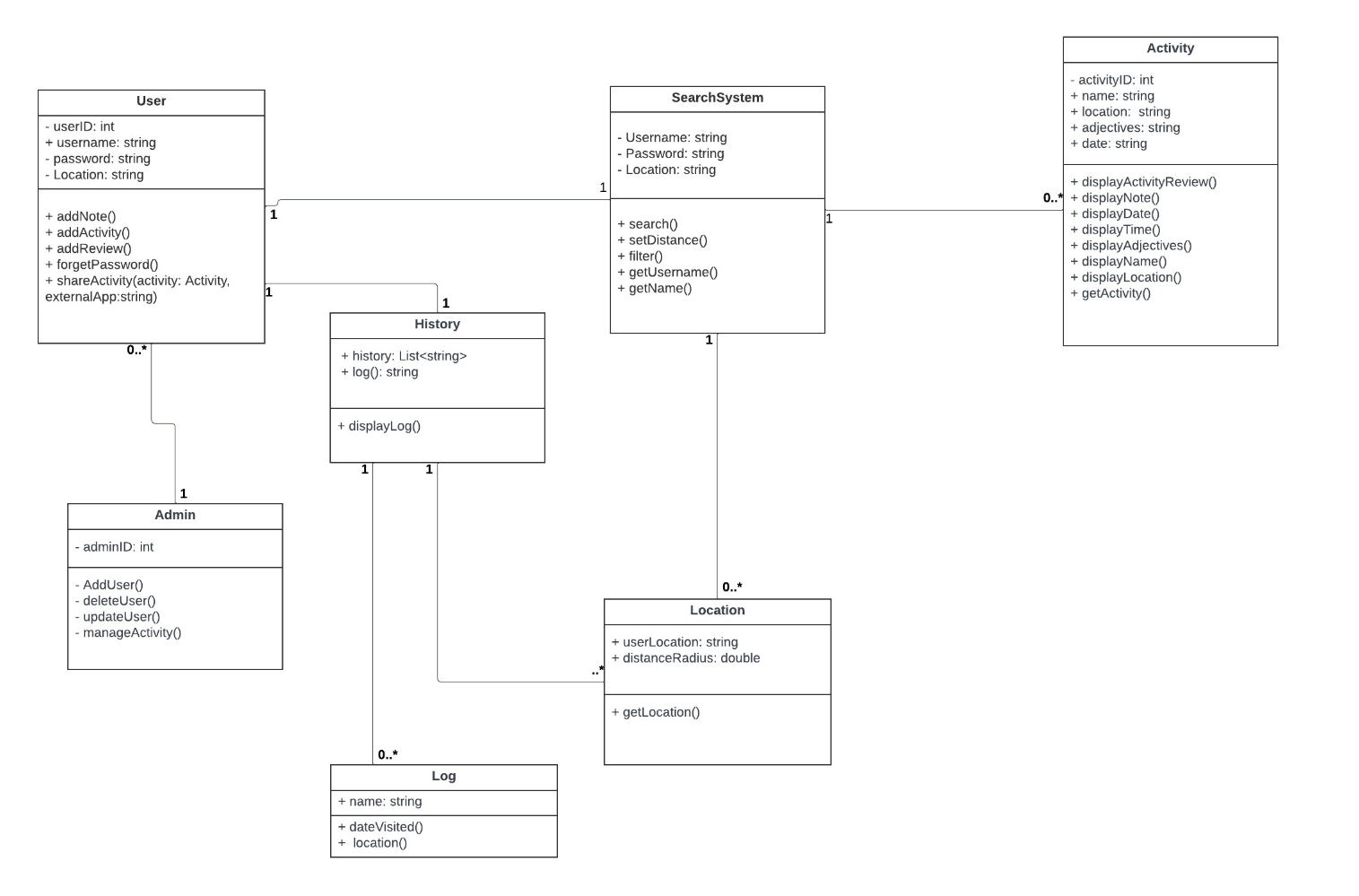
* This requirement is derived from the safety/security requirements leaf derived from the legislative requirement node from figure 4.3.

**Use Case Diagram**

A diagram of a software system

Description automatically generated

**Sequence Diagrams**

**Class Diagram**

**Architectural Design**

Layered Architecture Pattern

An overview of the architectural model for our software project. The architectural model serves as a blueprint for the design and structure of our software, outlining the key components, their interactions, and the overall system's organization.

Figure 6.8 and 6.9 of the Chapter 6 Slides were used as a reference for the model.

ExampleSearch.pdf is a diagram displaying the steps of a search operation in the software to help better demonstrate how the layer interacts with one another.

Layer Details:

Where2Next User Interface:

* Provides the graphical interface to user (Renders data and interface to user).
* Initiates all requests for data and operations via the users actions.

Where2Next Logic/Functional Layer:

* Receives inputs from the User Interface and maps user interaction to corresponding logic operations/functions.
* Carries out operations and requests necessary data from Where2Next Database

Where2Next Database:

* Houses all data specific to Where2Next such as tags and user data (Account Information, History, Reviews, Activities, Events).
* Actual location and event details are not stored here as this information can easily be retrieved from Google Maps and we want this to avoid excessive and redundant memory usage.
* The goal here is restrict stored data only to that which is necessary and unique to Where2Next.

Google Maps Interface:

* We do not have access to the internals to Google Maps, so Where2Next must use it as if it were a ordinary end user.
* Google Maps will provide end details such as address, closing time, and directions.
* The data from Google Map can be processed however necessary and presented via the Where2Next Interface rather than the Google Maps Interface.

Layered Architecture Pattern

A close-up of a document

Description automatically generated

ExampleSearch.pdf

A screenshot of a computer application

Description automatically generated