CS/SE-3354 Software Engineering

Final Project Deliverable 2

Where2Next

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

Delegation of Tasks for the Entire Project:

Throughout the entire project, all team members actively communicated and kept up to date with the project via Microsoft Teams and our occasional online meetings. Alongside this, in an attempt to distribute the work evenly, individuals who were assigned a larger portion for the first deliverable were given a lighter workload for deliverable 2 and vice versa. Overall, our group agreed that everyone contributed fairly and did their part.

Below is the explicit list of what each team member did for deliverable 1 and 2.

Charlie Alpert

* Deliverable #1
  + 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

* Deliverable #2
  + Wrote the Conclusion for the project alongside Nuria
  + Proofread the project to ensure it was free of any errors
  + Finalized and refined the presentation to its finished stated

Joseph Farghal

* Deliverable #1
  + Set up the GitHub repository for the project
  + Depicted Layered Architecture pattern for the project
* Deliverable #2
  + Listed the delegation of tasks for Deliverable #2
  + Collaborated with Lennon to complete the timeline for the project
  + Compiled final report for Deliverable #2 with Ariel

Dara (Oluwadara) Morakinyo

* Deliverable #1
  + Made the first commit to the repository with the README file
  + Created the functional and non-functional requirements.
* Deliverable #2
  + Collaborated with Ariel to estimated the cost of the project
    - Determined costs of software needed for development
    - Determined parameters for function point algorithm.

John Kasperbauer

* Deliverable #1
  + Created the use case diagram for the project
  + Created the functional and non-functional requirements
* Deliverable #2
  + Created the Test Plan for the project
  + Wrote the code to test with JUnit

Manh Ngo

* Deliverable #1
  + Created the sequence diagram for each case use
* Deliverable #2
  + Compared team project to existing apps such as Life360 and Swarm

Nuria Habib

* Deliverable #1
  + Created the class diagram showing the cardinalities and relationship types between the classes as well as attributes and methods
* Deliverable #2
  + Wrote the conclusion for the project
  + Proofread the project to ensure it was free of any errors
  + Finalized and refined the presentation with Charlie

Ariel Ong

* Deliverable #1
  + Wrote the delegation of tasks and justification behind choosing the spiral model
* Deliverable #2
  + Compiled final deliverable document with Joseph
  + Collaborated with Dara to estimate the cost of the project
    - Determined costs of software needed for development
    - Determined parameters for function point algorithm.

Lennon Crow

* Deliverable #1
  + Kept group on schedule while also compiling the report for Deliverable #1
* Deliverable #2
  + Created the timeline for the project and provided day by day breakdown of tasks

**Project Deliverable One Content**

**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 Permanent Activities require admin approval

Ex: new restaurant added

**URL to Group GitHub**

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

**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 seconds

* 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 efficiency.
* 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.

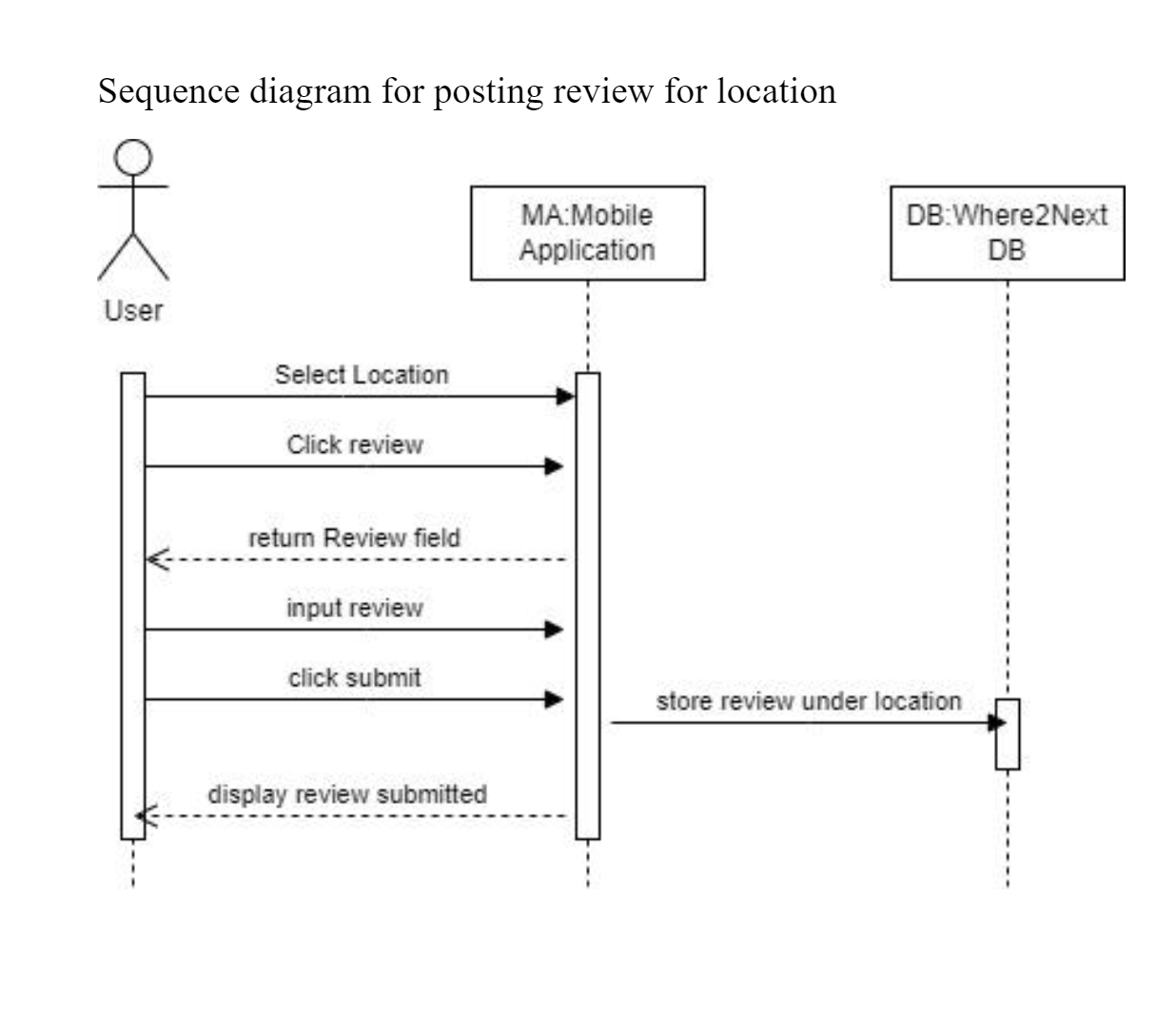
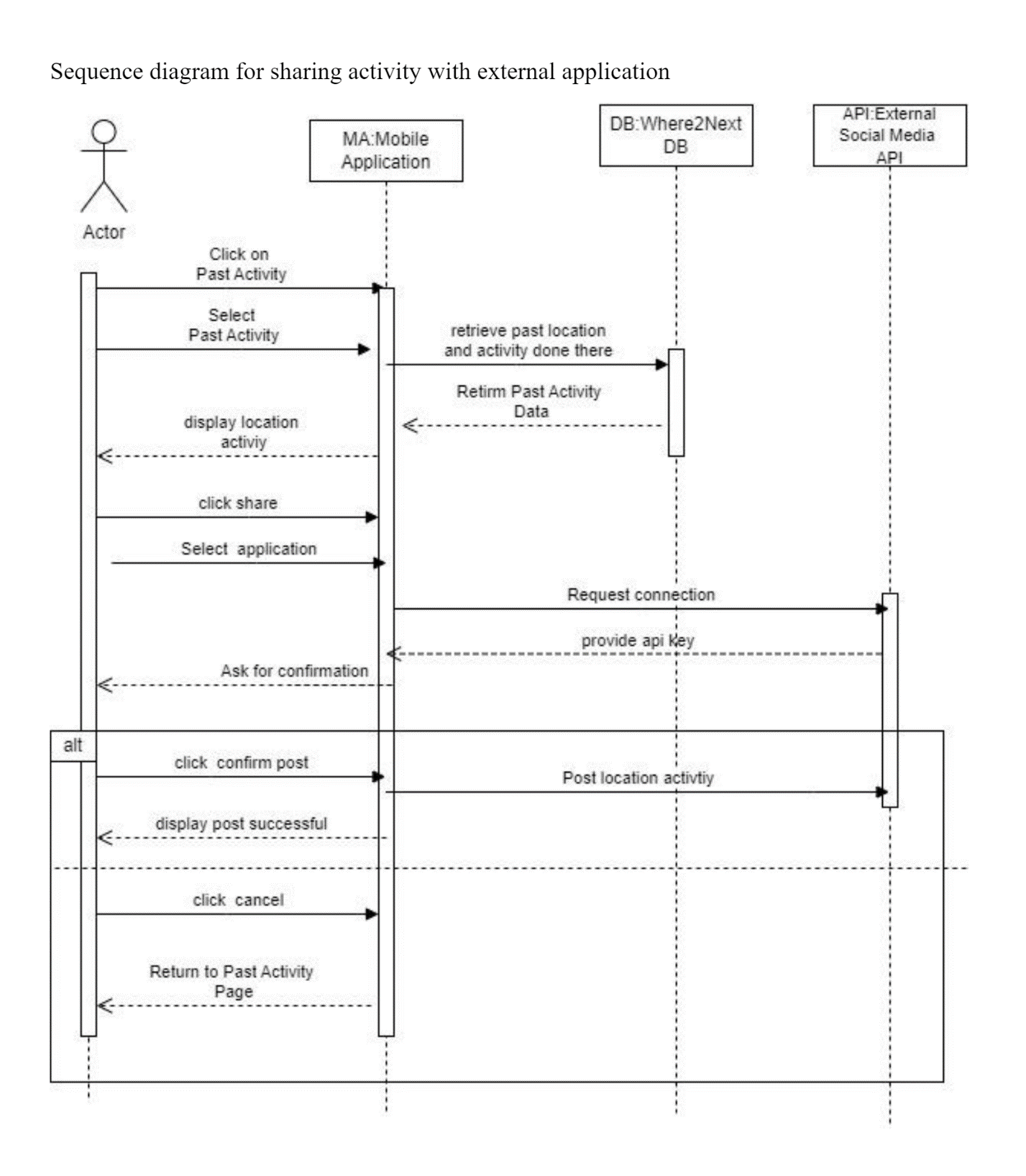
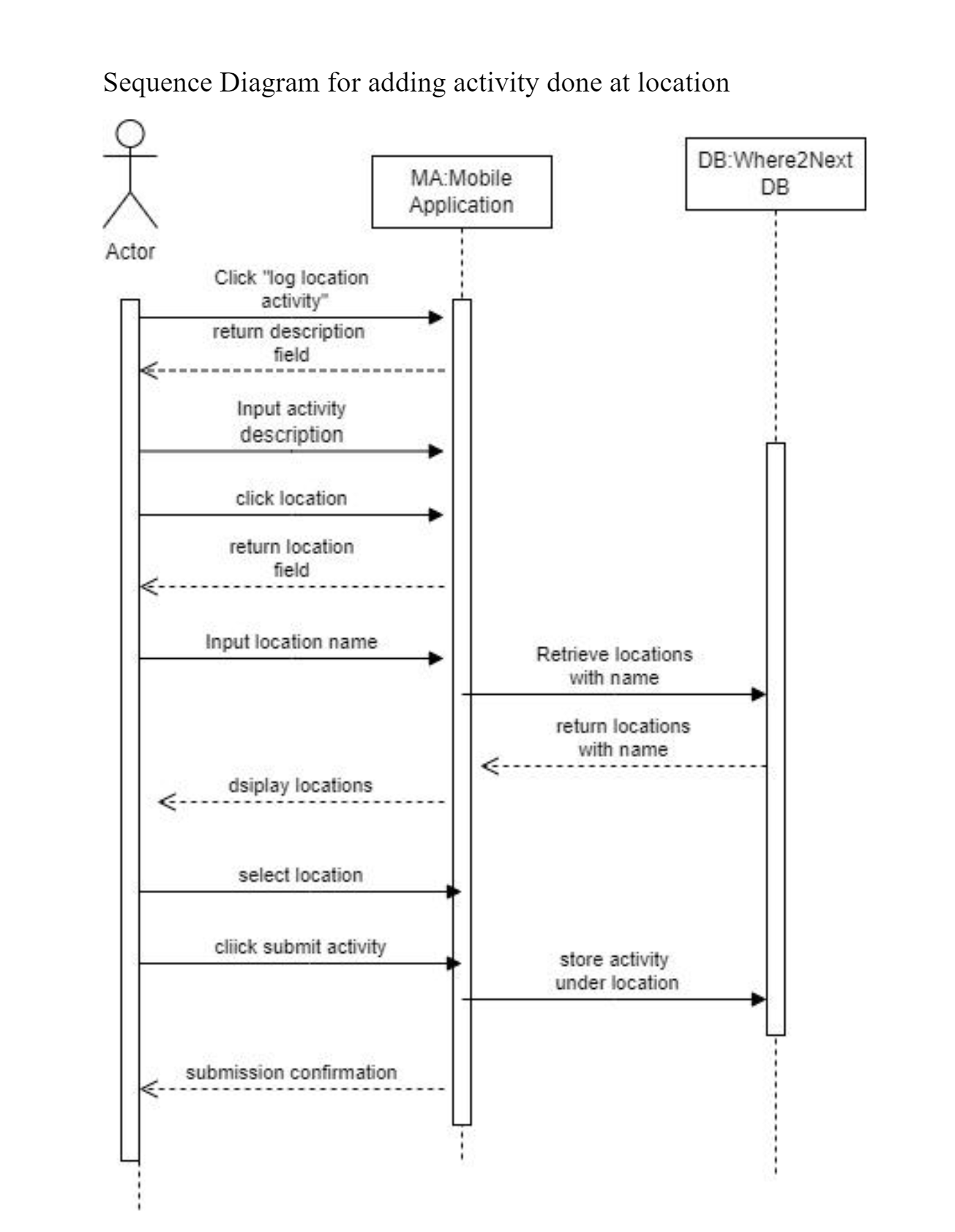
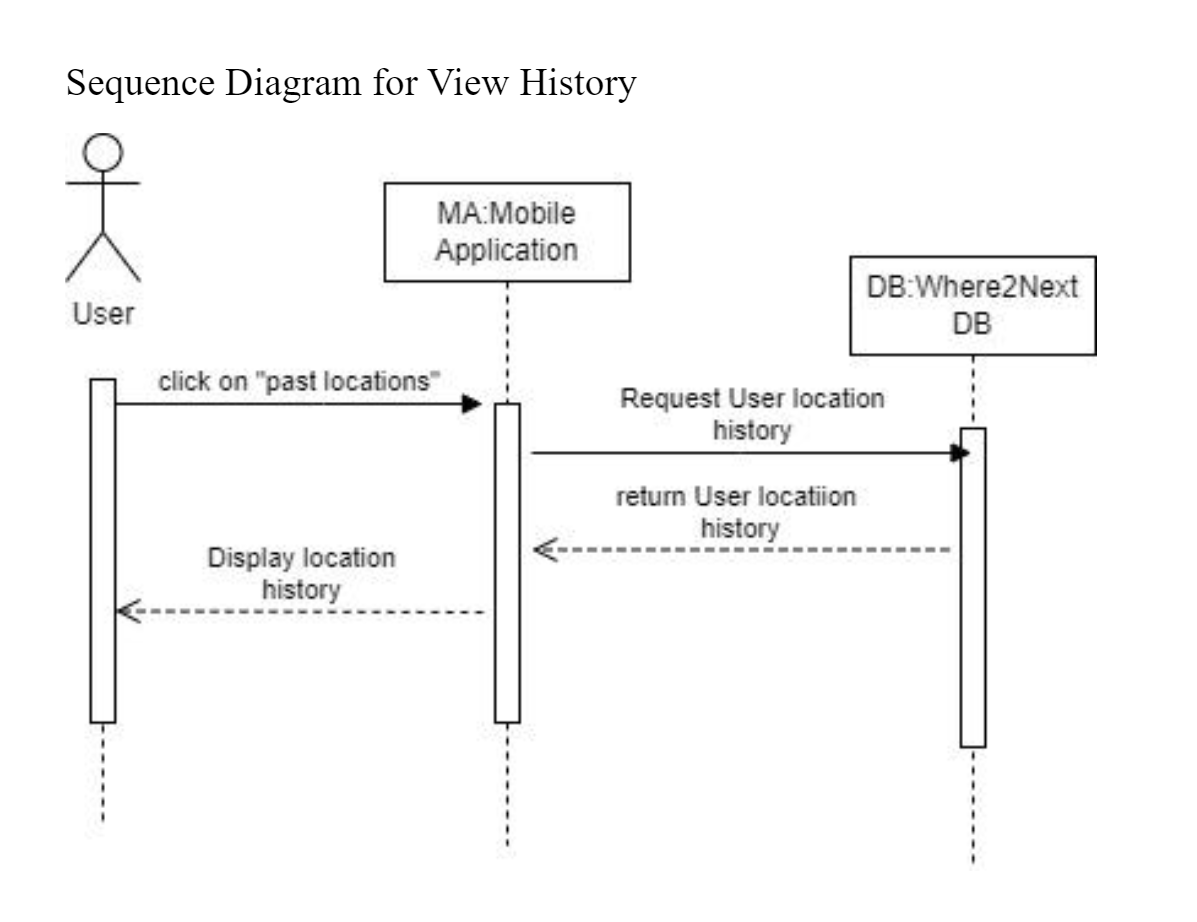
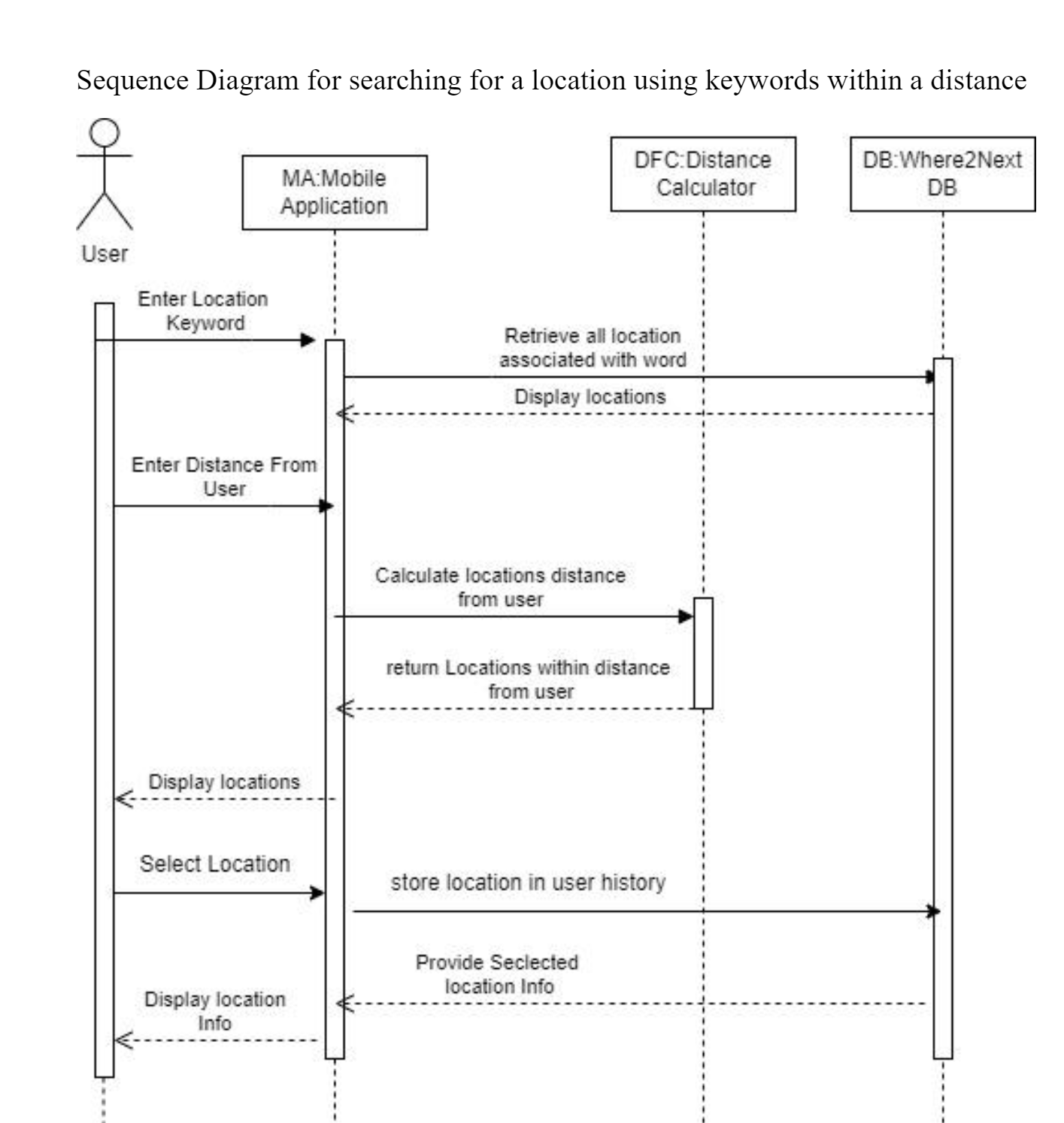
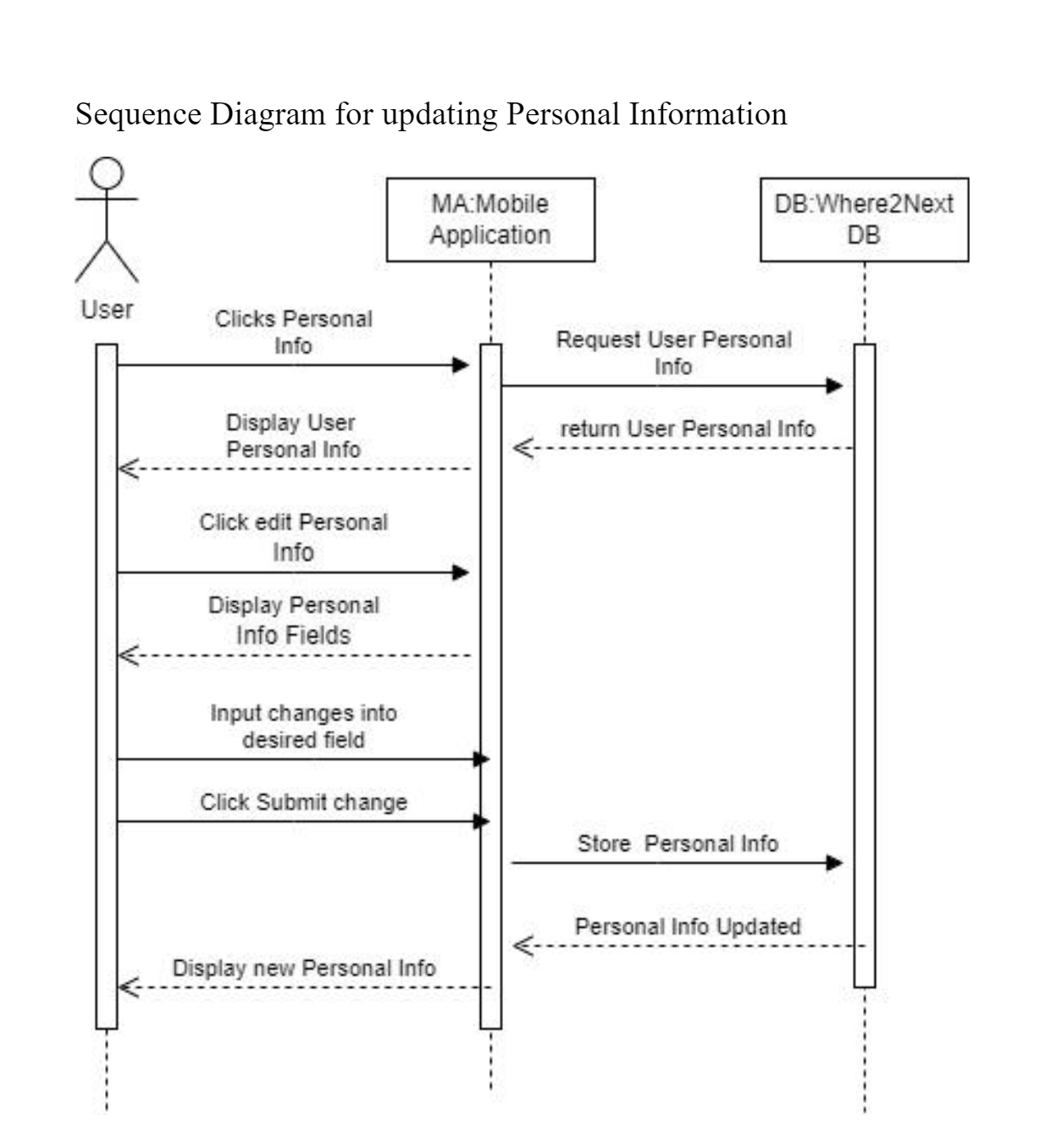
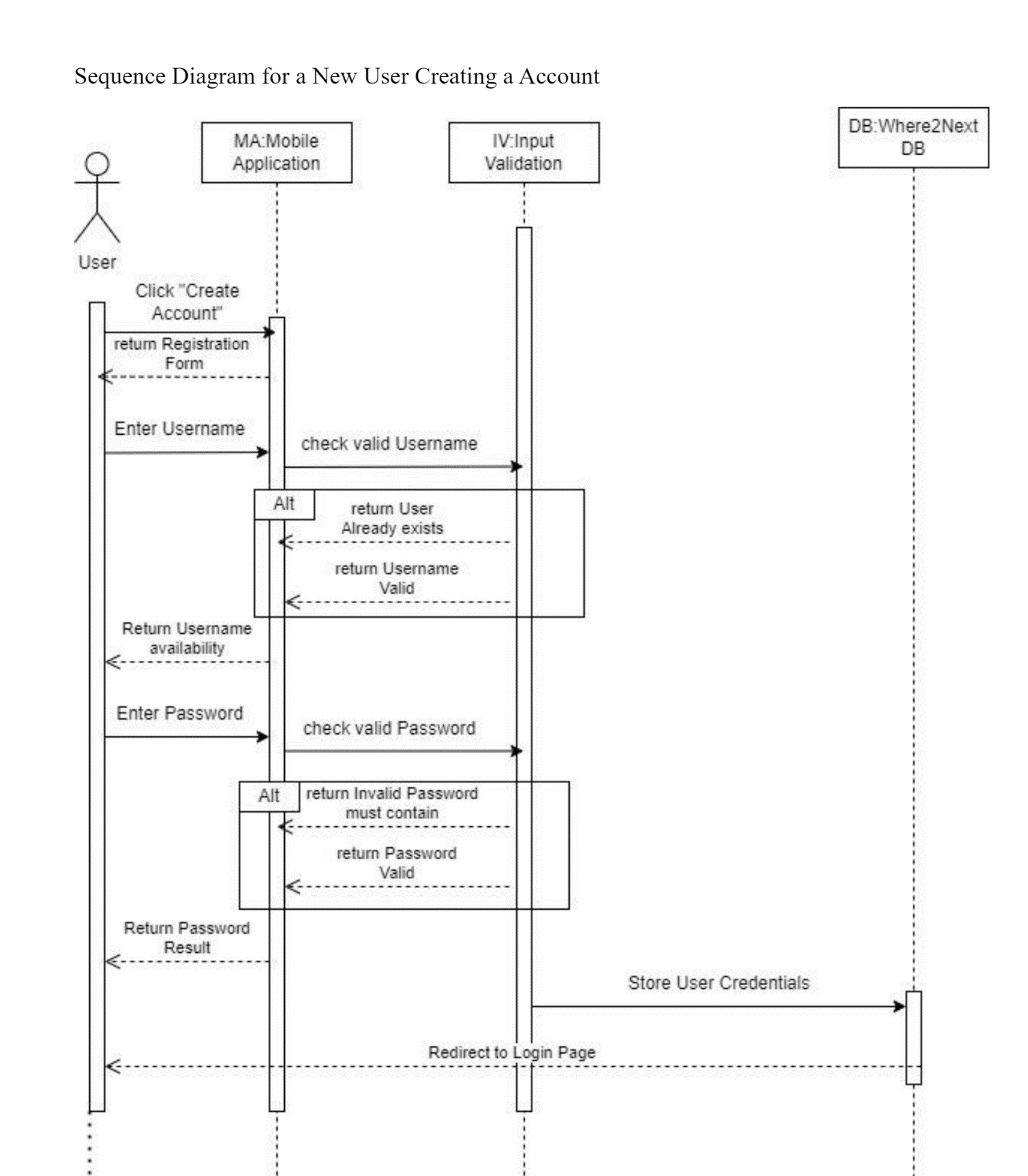
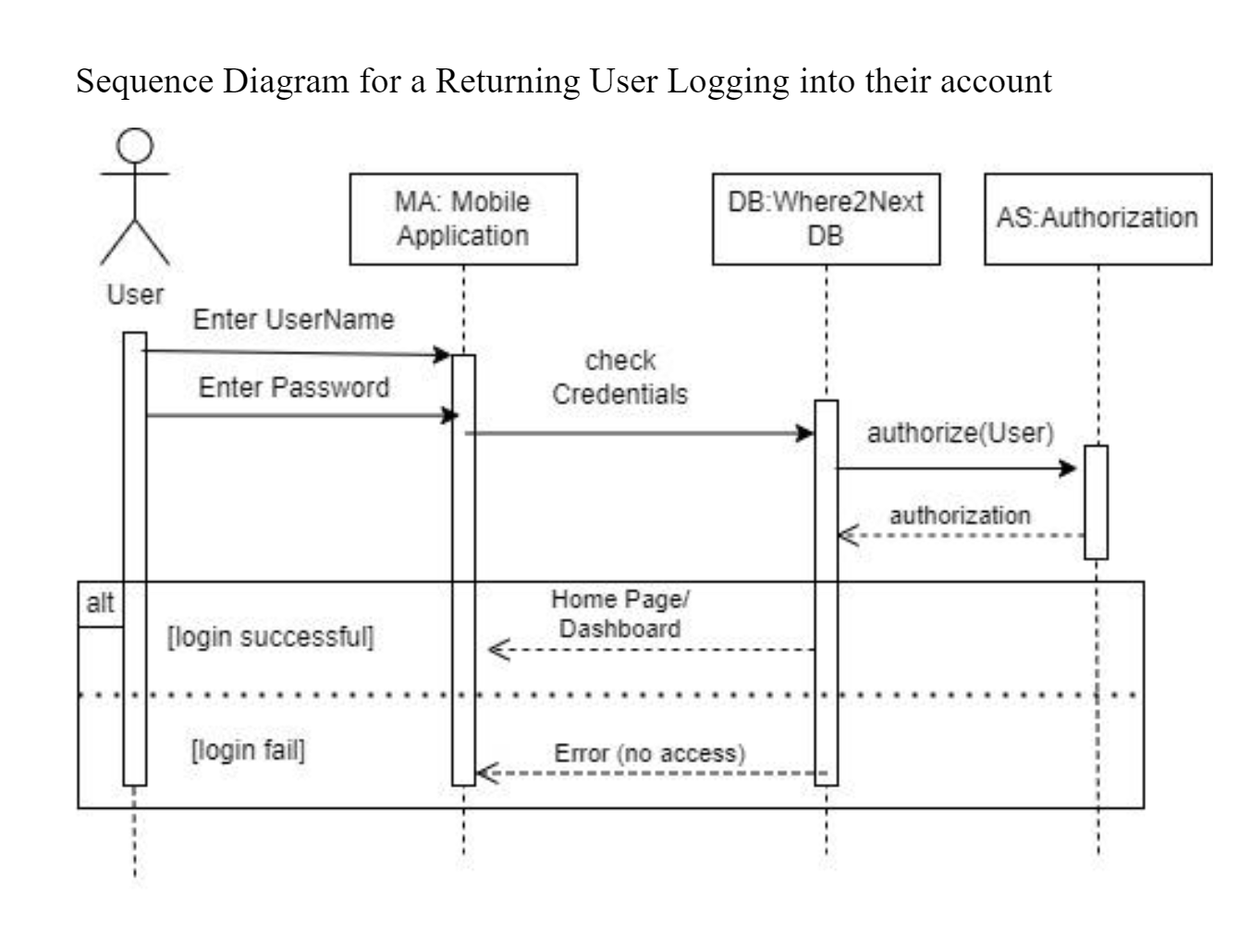
1. This app will not sell user data to third parties. Data will only be used to observe user trends and for the recommendation system.

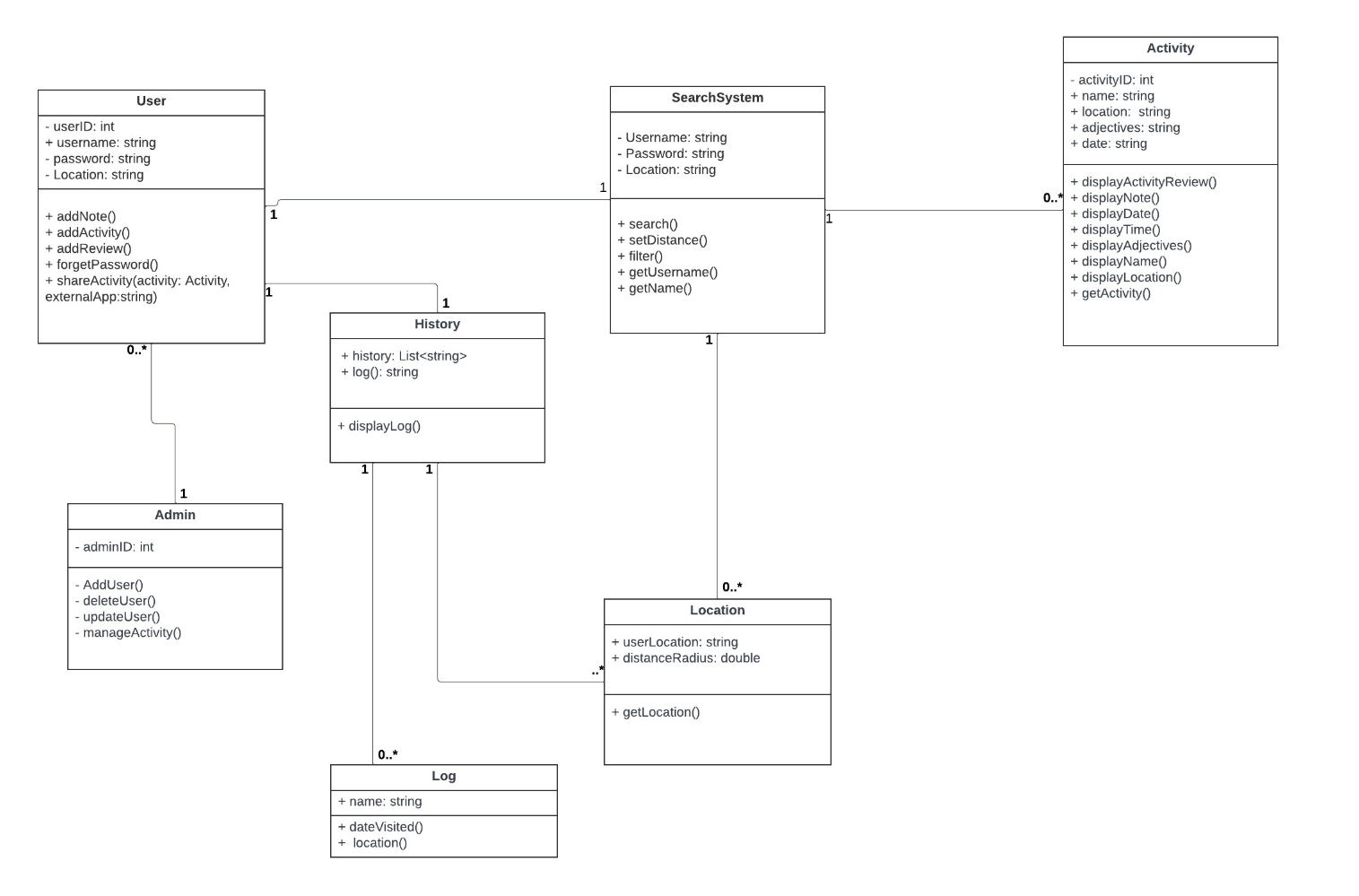
* This requirement is derived from the safety/security requirements leaf derived from the legislative requirement node from figure 4.3.

**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

**<< End of Project Deliverable One Content >>**

**Project Scheduling, Cost, Effort and Pricing Estimation**

*Project Scheduling*

Start Date: January 1st, 2023

Release Date: January 22nd, 2023

Week 1: Initial Planning and First Design

- Day 1: Start the project and define its scope.

- Days 2-3: Set up infrastructure and begin design work.

- Days 4-5: Conduct risk analysis on the design.

- Days 6-7: Develop the first prototype.

Week 2: Development and User Feedback

- Days 8-9: Develop the first prototype.

- Days 10-12: Conduct user testing and gather feedback. Assess risk analysis from suggestions.

- Days 13-14: Develop the final prototype, incorporating user feedback.

Week 3: Prototype Refinement and Final Testing

- Days 15-16: Complete the final prototype and commence in-depth testing.

- Day 17: Beta test again

- Days 18-19: Analyze beta test feedback, make final adjustments and assess risk from final suggestions

- Days 20-21: Complete project documentation and launch.

This timeline is based on our function point modeling technique with a team of 10 people. It is estimated to take 3 weeks to complete the project given to our team. The spiral method is also implemented so aspects of said model are in our timeline such as multiple prototypes and risk assessment. Weekends are counted as part of the schedule/work period. Normal working hours are 9 am - 5 pm as this is common company policy.

*Cost, Effort and Pricing Estimation*

The function point method was chosen. It was chosen because the fourteen questions, which determines processing complexity, gives significant insight on which characteristics are important to the application.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Function Category** | **Cost** | **Complexity** | | | **Count x Complexity** |
| **Simple** | **Average** | **Complex** |  |
| **1** | **Number of user input** | **5** | **3** | **4** | **6** | **20** |
| **2** | **Number of user output** | **4** | **4** | **5** | **7** | **28** |
| **3** | **Number of user queries** | **3** | **3** | **4** | **6** | **9** |
| **4** | **Number of data files and relational tables** | **5** | **7** | **10** | **15** | **50** |
| **5** | **Number of external interfaces** | **2** | **5** | **7** | **10** | **20** |
|  | | | | | **GFP** |  |

Q1: 4 Q2: 3 Q3: 2 Q4: 5 Q5: 5 Q6: 5 Q7: 2 Q8: 3 Q9: 3 Q10: 3 Q11: 3 Q12: 1 Q13: 3 Q14: 5

GFP = (5 \* 4) + (4 \* 7) + (3 \* 3) + (5 \* 10) + (2 \* 10) = 127

PCA = 0.65 + 0.01 (4+ 3 + 2 + 5 + 5 + 5 + 2 + 3 + 3 + 3 + 3 + 1 + 3 + 5) = 1.12

FP = 127 x PCA = 127 \* 1.12 = 142.24 FP

Productivity: 5 function points per person week because employees are not familiar with developing a project of this scope

E = FP/ productivity

E = 142.24/5 = 28.448 = 29 person-weeks

Team size = 10 since our project is a start-up company.

Project duration = E/team size = 29/10 = 3 weeks

Estimated cost of hardware products: $15000

* Company laptops: $1500 x 10 = $15,000

Estimated cost of software products over a year: $63,599.95

* Jira is a project management software, which we believe is necessary to keep track of our project schedule and employees. Jira is essential when using agile, so we believe it is worth the price of $1600 [1].
* In order to cut costs on hardware, we plan to use Google cloud services for storing information. The cost is around $5000 a month [2].
* Because our app utilizes Google Maps, we must purchase the Google Maps API license, specifically the license that allows us to use dynamic maps. Its cost is $1500 a year [3].
* Although we are a start up, we take cybersecurity very seriously, which is why we subscribed to FalconPro’s services, which cost $499.95 a year for small businesses [4].

Estimated cost of personnel over a year:

* Employee engagement has a budget of $500 a month.
* 10 employees
* Each employee will be paid $65,000 a year.
* According to the Harvard Business Review, on average, employee retraining costs a company about $24,800 per worker [5]. This cost covers the time spent training as well.

**Test Plan**

To test our software the minimum unit required for our software to function is the user profile containing user information. Here, there are six test cases in this order: a successful creation of a new user (as an object), a test case for both valid and invalid ages when registering, a test case for both valid and invalid names when registering, and a failed test case for passwords when registering. A single user profile requires three things: a valid name, a valid password, and a valid age. An age is considered valid if the user's age is within 15 to 109 years of life (14 < X < 110, where X is the user age). A [user]name is valid if both input names are no greater than 20 characters in length (the underscore is added by the system and not by the users). Finally, a password is considered valid if the length of the password is either between or exactly 6 and 12 characters long, complexity is not required! (6 <= X <= 12, where X is the number of characters in the password). Should a user successfully meet all criteria they will become registered as a user and added to the database [implementation pending], but should a user fail at any stage of the criteria then they will be prompted to correct their error. This does not extend to user age verification as a failure for that component results in a critical failure to discourage underaged and joke accounts from becoming created.

Successful Test Cases: A screenshot of a computer program

Description automatically generated

Test Case Overview for each individual component required to make a single user: A screenshot of a computer screen

Description automatically generated

All Test Cases Passed:

A screenshot of a computer

Description automatically generated

Failed test:

A screen shot of a computer

Description automatically generated

**Comparison of work with Similar Design**

Our team is developing an app called Where2Next, which aims to help people share their

experiences of various locations, and discover new places to visit. This paper will compare, and

contrast Where2Next with various popular location sharing apps available. There are many apps

that include location sharing as a feature, but the three apps this paper will be focusing on are

Life360, Glympse, and Swarm. These apps were selected because they focus primarily on

location sharing, rather than having it as a secondary feature.

The first app that will be compared to Where2Next is Life360 [6]. Life360 is a location

sharing app with a focus on family safety. In contrast to Where2Next, Life360 offers features that

alert you when members of your group visit a predefined location, or encounter a health

threatening accident. The location statistics shared by the app is in real time, and continues to be

sent from a user’s device as long as the app is installed on their device. To some people this may

raise a privacy concern. For this reason, Where2Next only shares a location when an individual

consents to sharing with other people. Furthermore, Where2Next allows individuals to share

context about what they did at the location. The two apps may differ in terms of their treatment

of privacy, but they are similar in their range of use. Both applications are mobile applications

that are functional internationally. This allows users to utilize the app regardless of where they

travel.

The second app that will be compared with Where2Next is Glympse [7]. Glympse is a

location sharing app that attempts to remedy the privacy concerns of life360 by giving others a

“glimpse” or short window where real time location data is shared. This feature allows Glympse

to be a community focused app that helps coordinate meetups, and events. Where2Next differs

from Glympse in how community meetup’s function. In the Where2Next app, meetups are

limited to sharing the current individual location, and not the current progress toward a location

like in the Glympse App. This prevents any possible privacy concerns that may result from an

individual forgetting to turn off location sharing in the Glympse App. Where2Next is similar to

Glympse in its use of tags to identify related events. Yet Glympse is limited in the scope of its

tag. Glympse tags are limited to small towns, or areas in order to effectively coordinate events.

The locations found by Where2Next tags may be expanded to include the entire city, or state.

The third app that will be compared with Where2Next is Swarm [8]. Swarm is a location

sharing app made by Foursquare in order to share locations with friends while traveling. Out of

the three apps, Swarm is the most similar app to Where2Next. Like Where2Next, Swarm focuses

on the social aspect of location sharing. Both apps detect when a user visits a location, and

records it into a log. Furthermore, both apps allow the user to add activities done at a location

into the comment section of a post, and share the post with friends. Yet Swarm differs to

Where2Next in its implementation of social networking. In the swarm app, social media is built

into the app as a feature. In the Where2next app, the app relies on sharing to external social

media sites. By relying on external social media sites, the app reduces the number of resources

that are necessary in order to support those services.

In conclusion although there are many location-based sharing apps available,

Where2Next is distinct from its competitors through its implementation of privacy-based

location sharing, wide search range, and utilization of external social media platforms. These

features help create a smooth functioning app that ensures the users privacy.

**Conclusion**

Our project, Where2Next has demonstrated feasibility and utility of a Google Maps extension designed to enhance user experience in exploring new outings and restaurants. Throughout our journey, we've employed a systematic approach, utilizing the Spiral Model to ensure iterative development and continuous improvement.

The core of Where2Next lies in its innovative features like adjustable radius search, tagging system for user-preferred attributes, and a log of visited locations, setting us apart from mainstream apps such as Yelp. In comparison to similar apps like Life360 and Glympse, Where2Next shines with its unique emphasis on privacy in location sharing, extended search capabilities, and seamless social media integration. This positions us favorably against competitors and addresses specific user needs that are currently overlooked in the market.

However, our project isn't without its challenges. The dependence on external platforms for certain features and the ongoing task of ensuring robust data privacy and security are areas that need further development. Looking ahead, focusing on making Where2Next more self-sufficient and reinforcing data protection would be beneficial. Future improvements could include a pivot towards a more self-reliant architecture, possibly through the MVC framework, and advanced data protection measures.

In summary, Where2Next demonstrates our commitment to innovation and user-focused design, potentially reshaping the use of location-based services with its unique combination of functionality, personalization, and privacy.

**References**

[1] Atlassian, “Jira Pricing - Monthly and Annual Subscription Cost per User,” *Atlassian*. https://www.atlassian.com/software/jira/pricing# (accessed Nov. 12, 2023).

[2] “Cloud Storage pricing  |  Cloud Storage  |  Google Cloud,” *Google Cloud*, Sep. 13, 2019. https://cloud.google.com/storage/pricing (accessed Nov. 12, 2023).

[3] Google, “Pricing Plans and API Costs,” *Google Maps Platform*, 2023. https://mapsplatform.google.com/pricing/ (accessed Nov. 12, 2023).

[4] “Cybersecurity Solutions For Small Businesses (SMB) - CrowdStrike,” *crowdstrike.com*. https://www.crowdstrike.com/solutions/small-business/ (accessed Nov. 12, 2023).

[5] A. Chopra-McGowan, “Make Sure Your Company’s Reskilling Efforts Pay Off,” *Harvard Business Review*, Jul. 13, 2021. https://hbr.org/2021/07/make-sure-your-companys-reskilling-efforts-pay-off (accessed Nov. 12, 2023).

[6] “What does Life360 do?: Family safety app features,” Life360 Family Locator App,

https://www.life360.com/learn/what-does-life360-do/ (accessed Nov. 8, 2023).

[7] “Glympse app,” Glympse Real-Time Geo-Location Technology,

https://corp.glympse.com/glympse-app/ (accessed Nov. 8, 2023).

[8] E. Moreau, “What is the Swarm app and how are you supposed to use it?,” Lifewire,

https://www.lifewire.com/foursquares-swarm-app-3485804 (accessed Nov. 8, 2023).