Caravan Design Document

CS495 Spring 2018

Zach Hartzog, Meaghan Martin, Riley Van Wyk, Brian Zeifert

Change History

Date	Version	Description	Author
February 9, 2018	1.0.0	Introduction/Glossary	Zach Hartzog
February 9, 2018	1.0.1	Formatting Improvements	Zach Hartzog
February 9, 2018	1.0.2	Competitors Table Updates	Zach Hartzog
February 15, 2018	1.0.3	User Requirement Edits	Zach Hartzog
February 17, 2018	1.0.3	Project Description	Zach Hartzog
February 22, 2018	1.0.4	Competitor Analysis	Meaghan Martin
February 22, 2018	1.0.5	Project Description Edits	Riley Van Wyk
February 22, 2018	1.0.6	Project Description Edits	Zach Hartzog
February 22, 2018	1.0.7	Activity, Use Case, Class Diagrams	Zach Hartzog
February 22, 2018	1.0.8	Competitor Analysis	Riley Van Wyk
February 23, 2018	1.0.9	Class Diagram, Project Description Updates	Zach Hartzog
February 26, 2018	1.0.10	Activity Diagram Updates	Zach Hartzog
March 1, 2018	1.0.11	Diagram Descriptions	Meaghan Martin
March 28, 2018	2.0	Clean up Requirements Doc formatting based on feedback	Meaghan Martin
March 28, 2018	2.01	New Class Diagram & Description	Meaghan Martin
March 28, 2018	2.0.2	Added Sequence Diagrams & Descriptions	Meaghan Martin
April 2, 2018	2.0.3	Updated Introduction & Goals	Meaghan Martin
April 3, 2018	2.0.4	Description Edits	Meaghan Martin
April 3, 2018	2.0.5	Updated Activity Diagram Formatting	Meaghan Martin

Contents

3. Introduction

3.1 Purpose

Caravan is an Android app that exists to simplify traveling your way. The app will provide curated lists with location suggestions to the user based on their selected City, Neighborhood, Mood, etc. The app will provide easy access to information on the exact type of location the user is looking for at any given place or time.

3.2 Scope

The proposed system will create a way to find the locations best suited to the user by leveraging smartphone technology as a way to provide location-specific blueprints. The app will be immersive and create a new, visual way for users to create the perfect itinerary. The app will also be highly connected to existing social media platforms to enhance ease of use and to build the user base through content sharing.

The vision for the Caravan application is a multi-city travel experience where users can create and save blueprints for cities across the globe. For the purpose of this class, the app will be focused solely on Nashville, Tennessee as a starting point and proof of concept.

3.3 Goals

The app will be built with scaling in mind, as the end goal for the Caravan app is entrepreneurial in nature. The goal of Caravan is to create a seamless experience for the user to find curated types of venues that best match their preferences, and minimize the user's time spent searching for a location and allow them to simply enjoy the moment. Furthermore, Caravan aims to provide a travel experience that is both highly personalized and customizable, but still leaves room for flexibility and exploration.

4. Glossary

- **Android**: a software stack for mobile devices that includes an operating system, middleware and key applications.
- **Android SDK**: the software development kit that provides tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.
- **API**: application programming interface, a particular set of rules and specifications that software programs can follow to facilitate interaction.
- **Application framework**: framework that enables reuse and replacement of components
- **Blueprint**: list of places
    - **Curated Blueprint**: Pre-made guide
    - **User-Created Blueprint**: Users can pick and choose places from featured guide to add to their own super-personalized guide
- **City**: A city (i.e. Nashville, Atlanta, etc.) chosen by the user to search in for places/neighborhoods/guides.
- **Explore:** Search activity that looks through all Caravan items including locations, blueprints and cities
- **Framework**: an abstraction in which software providing generic functionality can be selectively changed by user code, providing application specific software.
- **Location**: Location or attraction like a restaurant, park, theater, etc.
- **Mood**: a Curated Guide that pertains to a certain setting or type of location chosen by the user.
- **Neighborhood**: A specific area of a city, i.e. "West Nashville", "NorthEast Atlanta", etc.
- **Traveller -** Any Caravan application User.

---

5. Project Description

5.1 The Application

Planning a trip can be overwhelming - trying to plan everything you want to do in a city can be exhausting. More often than not a traveler will leave a place having missed out on the most ideal experience(s). Enter Project Caravan. Caravan aims to connect travellers and people exploring new cities with a complete blueprint of places according to what they are looking for through a simple and visual application on their mobile device. Our goal is to maximize experience and minimize effort. We will accomplish this by providing a mobile system a user may install on their phone or mobile device.

5.2 Explore Feature

A simple, easy-to-use view that will allow the user to search for curated blueprints, cities, neighborhoods, or individual locations provided by Caravan.

5.3 City Feature

A list of blueprints and places organized by category specific to a user-specified city.

5.4 City Blueprint Feature

A list of blueprints organized by mood, neighborhoods, and locations specific to a user-specified city.

5.5 Blueprint Feature

A Blueprint will serve as the all-encompassing guide for the user's experiences. The Blueprint will contain eight (8) locations, each of which will have a small descriptive blurb. To see more information about a specific location a user will access the Place feature.

5.6 Neighborhood Feature

Larger cities are usually a conglomerate of smaller neighborhoods, each with their own culture, cuisine and attractions. The Neighborhood feature is aimed at travellers who want to take a deeper dive into the area or those who are trying to experience all the area has to offer with traveling too far. Curated blueprints for a neighborhood will showcase the area's can't-miss attractions.Whereas city-specific blueprints may take travelers to opposite sides of the city, neighborhood-specific blueprints are guaranteed to only show locations within that specific geographic area.

5.7 Location Feature

A Location will serve as the information hub for a specific experience.

5.8 Home Feature

A simple view with featured Blueprints and Places for the user to explore based on Geographic location.

5.9 Login Feature

The initial view a user will encounter (after the splash screen). The Login view will give the user an option to either create an account, or login with Caravan, Google, or Facebook credentials.

5.10 Profile Feature

The profile view will contain personal information stored in the user's Caravan account, as well as provide access to all saved or created Blueprints and Places.

6.11 Constraints

The amount of data transferable to the device at one time will be constrained to the amount of available memory space on the device.

6.12 Assumptions and Dependencies

The minimum SDK version the application will support is Android 5.1 Lollipop and the target SDK version is Android 8.0 Oreo. The application will require data connection (3G) in order to update maps and database listings.

6. User Requirements Definition

6.1. Functional Requirements

High Priority:

1. The user must be able to create a new account, or sign in with Facebook, Google, or Personal credentials.
2. The user must be able to traverse a list of curated guides
3. The user must be able to create their own guides, saving specific places from other guides
4. The user must be able to view information about a specific place: Hours of Operation (HOP), Menu / Types of services, Calendar of Events, Website, etc.
5. The user must be able to store and access saved guides (Personal and Curated)
6. The user must be able to access guides without downloading them

Middle Priority

7. The system must be able to recognize GPS location and suggest a guide accordingly in the form of a push notification
8. The user must be able to share guides on social media platforms or with direct messaging and email

Low Priority

9. The user must be able to view all places in a specific guide on a map

6.2. Non-Functional Requirements

High Priority

1. The system must be able to operate and provide access to guides without a network
2. The system must be able to find places/guides associated with a search term in minimal time
3. Must not strain battery life

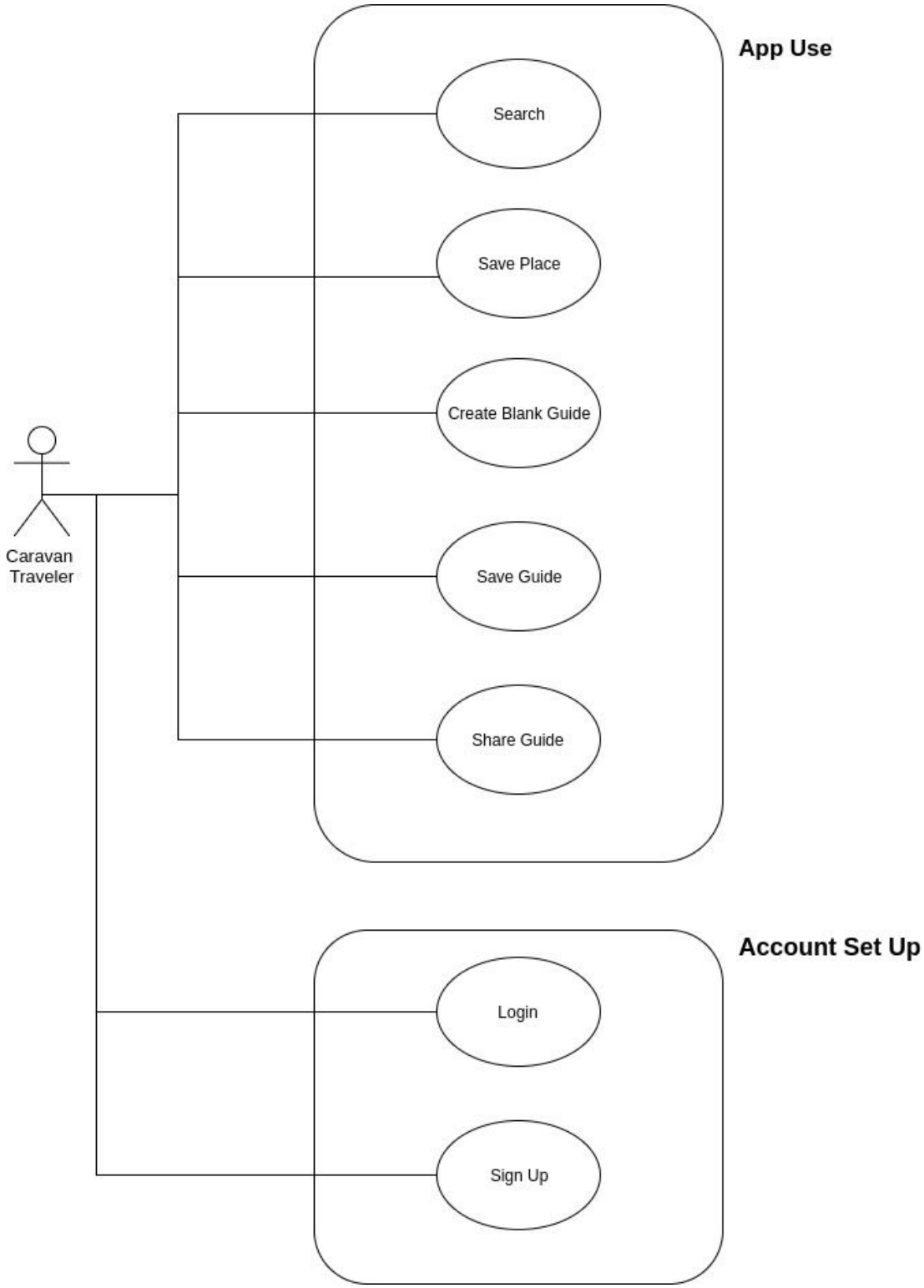
---

7. Competitor Identification

Y = Yes, N = No, P = Partial

	Caravan	**Travefy** Monetize Travel Plans, like a personal travel agent	**Google Trips**	**Roadtrippers** Geared more towards road trips across multiple locations	**Tripit** Geared more towards the transportation process of *traveling*	**Guides by Lonely Planet** *One* guide for each city
Offers Curated Guides	**Y**	**N**	**Y**	**N**	**N**	**P**
Offers User Created Guides	**Y**	**P**	**N**	**N**	**N**	**N**
Provides a Complete Itinerary (Breakfast, Lunch, Dinner, Recreation, Entertainment, Nightlife, etc.)	**Y**	**Y**	**Y**	**N**	**N**	**N**
Content Available Offline	**Y**	**Y (PDF)**	**Y**	**N**	**Y**	**Y**
Cost	**Free**	**$16-39 per month**	**Free**	**Free**	**Partially Free (TripIt Pro)**	**Free**

8. Use Case Diagram

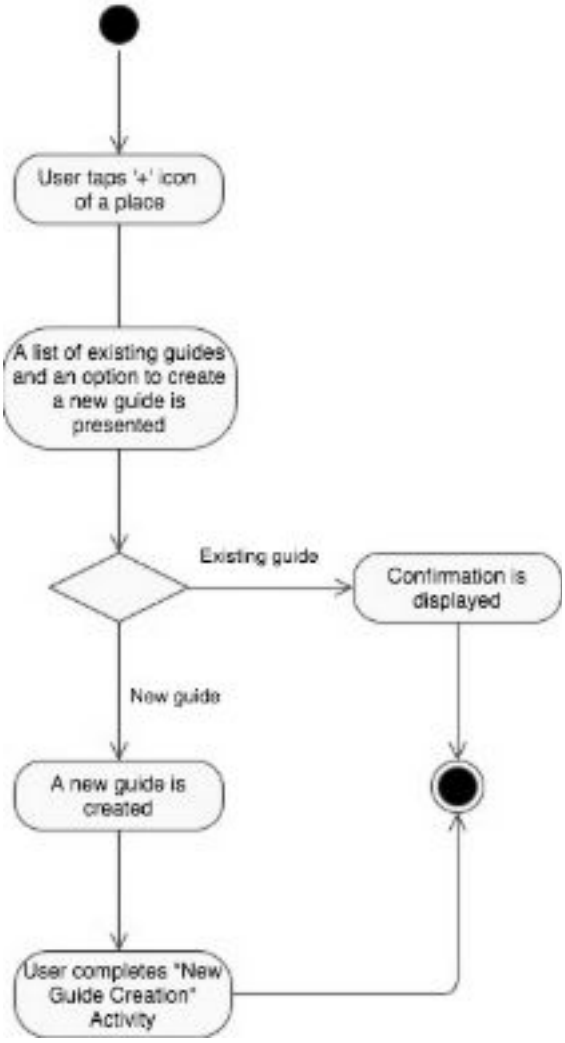


The above diagram shows the main uses of the Caravan app. We've identified a single actor based on our required features, which is the Caravan Traveller. We've broken down the uses of the app into two main categories: Account Set Up and Maintenance and Travel Exploration. The traveller can sign up and create a Caravan account and log in to a previously-created account. The app also has a few basic, but novel uses. The traveller can create their own travel blueprints by saving places to it, or they can save an existing guide they find interesting. Users can also utilize a search function to help them discover places or guides. Additionally, users can share a guide via email, text or social media post, all of which prompt the recipient to download the Caravan app.

---

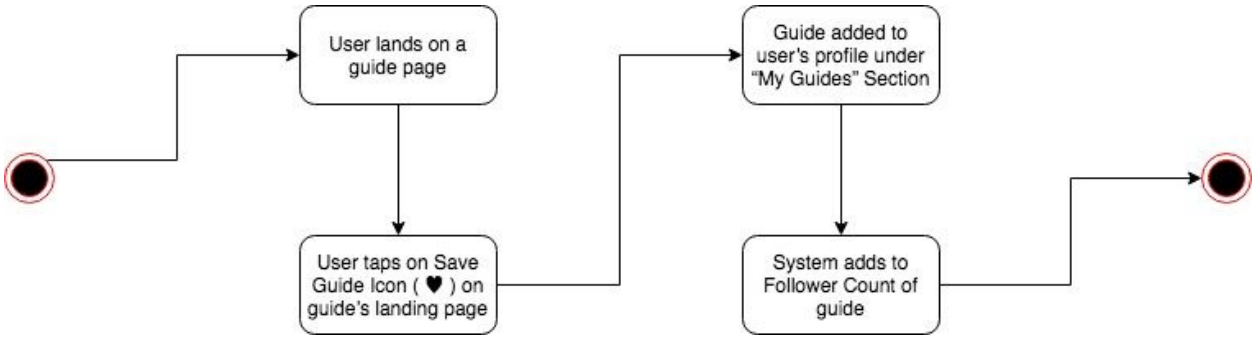
9. Activity Diagrams

9.1 Save a Place



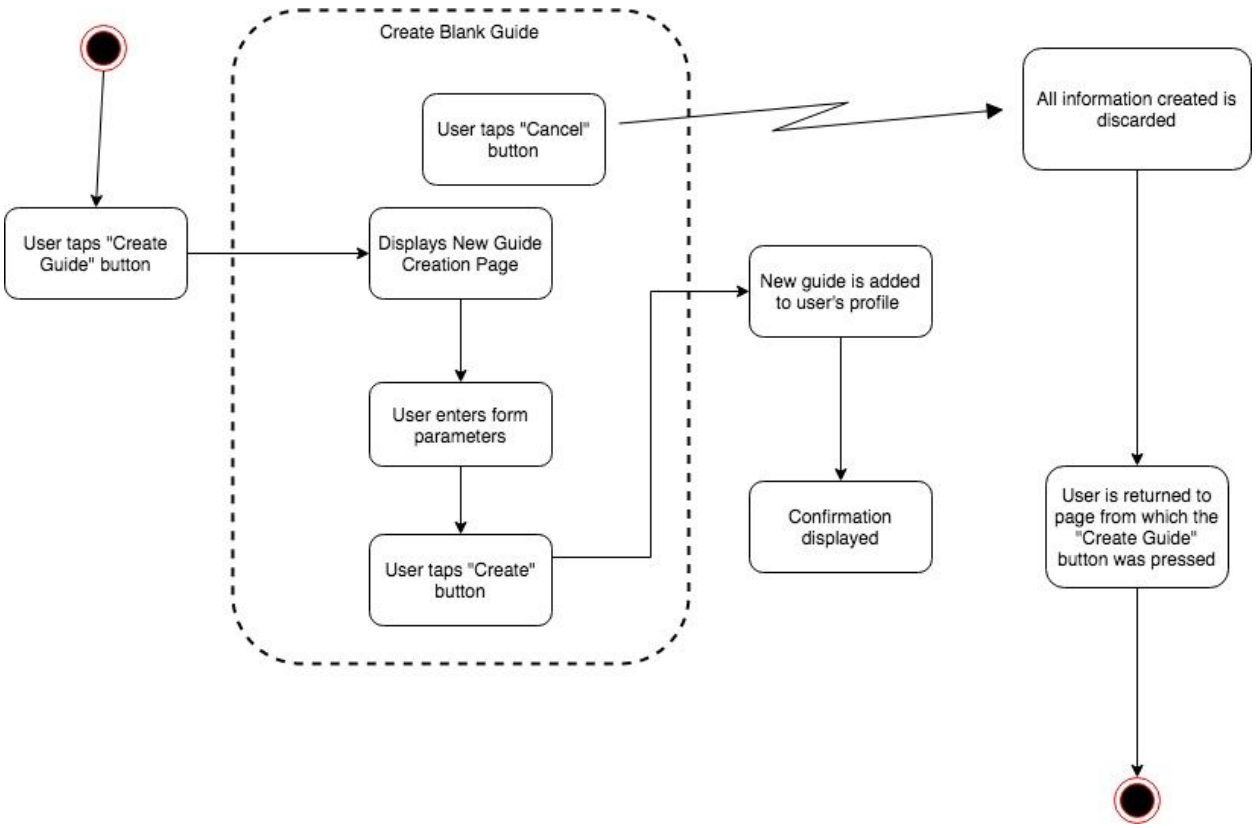
To the left is the process for *Saving a Place*. A user discovers a place either through the search screen or in an existing blueprint. When the user taps the **+** icon, they will be prompted to either add the place to one of their existing blueprints or to create a blueprint and add that place. If the user chooses one of their existing guides and the addition is successful, a confirmation will be displayed and the activity is over. If user chooses to create a new blueprint, they are taken through that activity. Once the new blueprint is created, the place is added and the activity is over.

9.2 Save a Guide



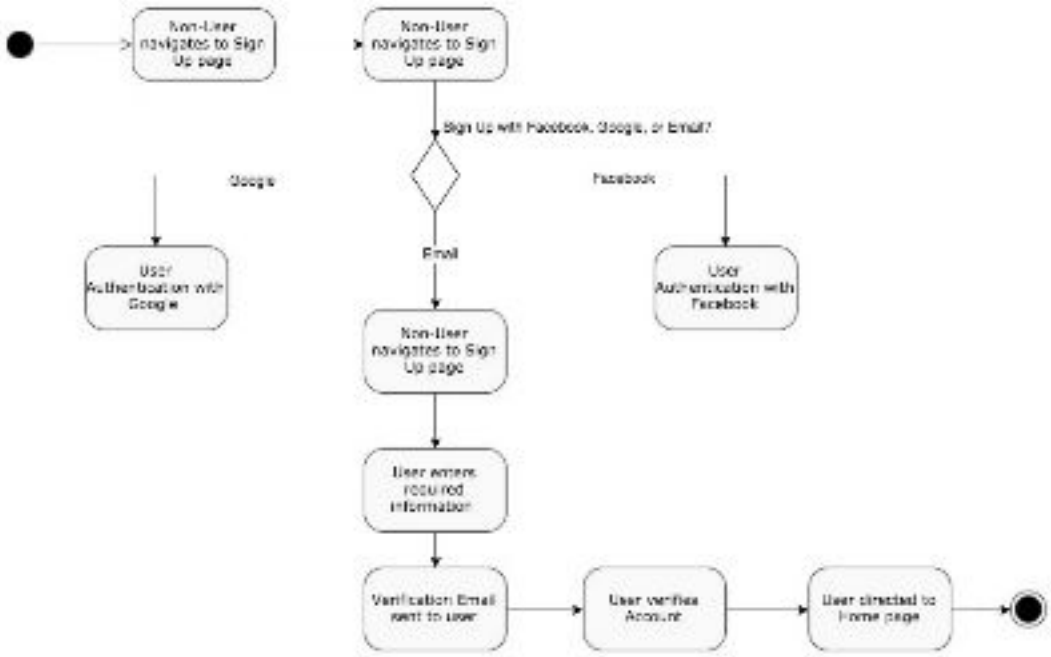
Above shows the workflow for *Saving a Guide*. This functionality is similar to Spotify's playlist following functionality. A user finds a guide, either through the curated blueprint section, the search screen or their own exploration. By tapping on the *heart* icon on the blueprint's landing page, they can save it to their profile. The user can then access the blueprint through the "My Guides" section of their personal profile and the system increments the blueprint's follower count before the activity ends.

9.3 Create a Guide



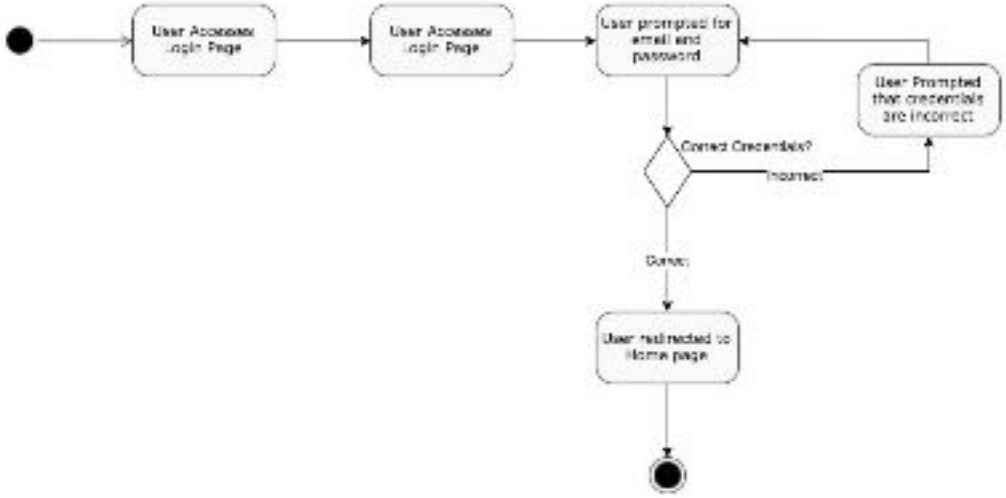
One of the key functionalities of the Caravan app is to create a custom travel blueprint based on the users interests and personality. A user can create a new guide and then add places that interest them. The user enters the *Create Blank Guide* activity through one of two activities: choosing to create a new blueprint when saving a place, or the "Create Blueprint" button. Once entering the activity, a form is displayed to the user. They are prompted to enter information like a Blueprint Name, a description and similar parameters. Once the user is content with the information they've entered, they tap on the "Create" button which finalizes the creation. The blueprint is added to the user's profile and a confirmation is displayed before the activity ends. At any time, the user can cancel the activity. Once choosing to cancel, all of the information that the user entered is deleted and the user is brought back to the previous page.

9.4 Sign Up



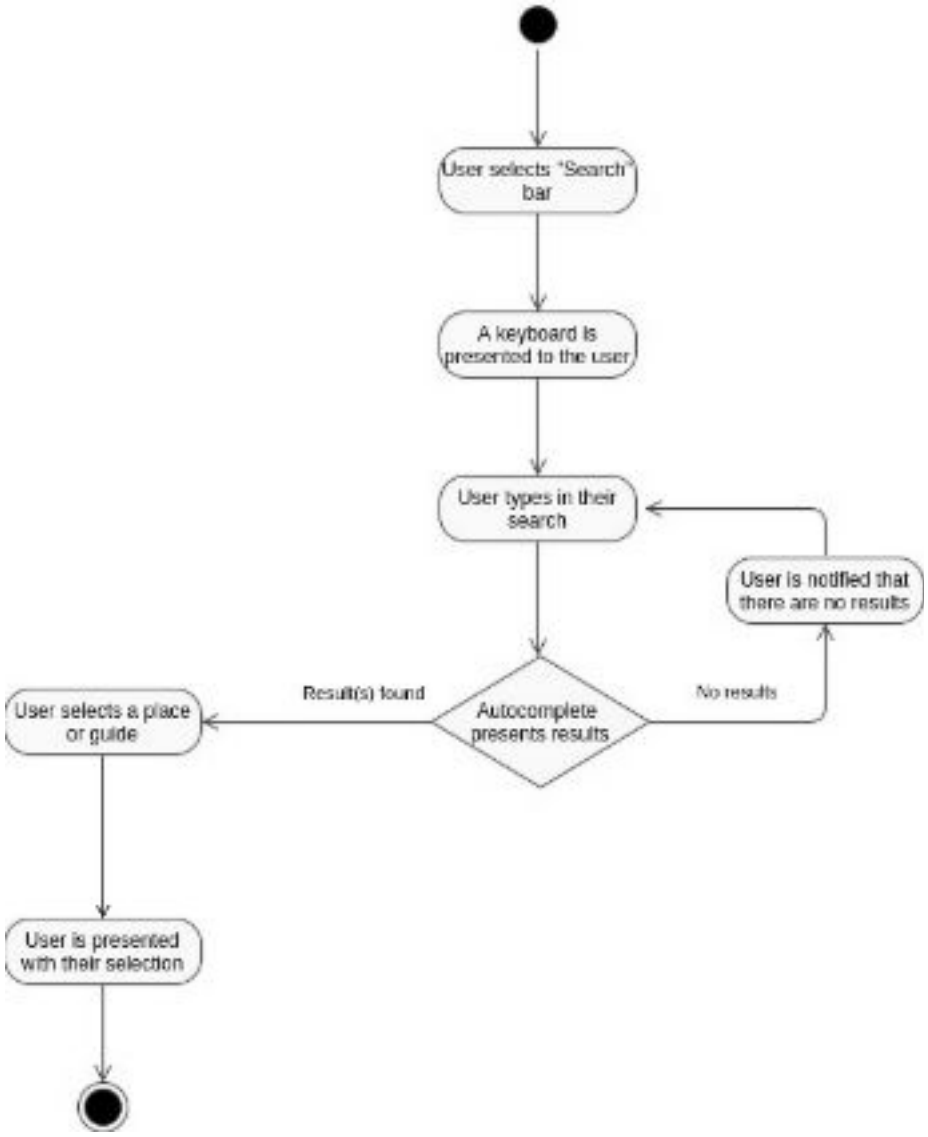
The diagram above shows the flow for *Signing up* for a Caravan Account. The new user has a few different options for how they created their account. If the user chooses to sign up using Google or Facebook, those respective APIs will be called for account verification. If the user chooses to sign up with just their email address, they will be prompted to enter more information like their name and date of birth. They will be sent a verification email to confirm their account. Once the account is confirmed, the new user is taken to the Caravan homepage.

9.5 Log In



The *Log In* workflow is shown above. A user accesses the Login page and enters their credentials. If the email and password match, the user is taken to the Caravan Homepage. If the credentials are not valid, the user is prompted to try again until they log in successfully.

9.6 Explore

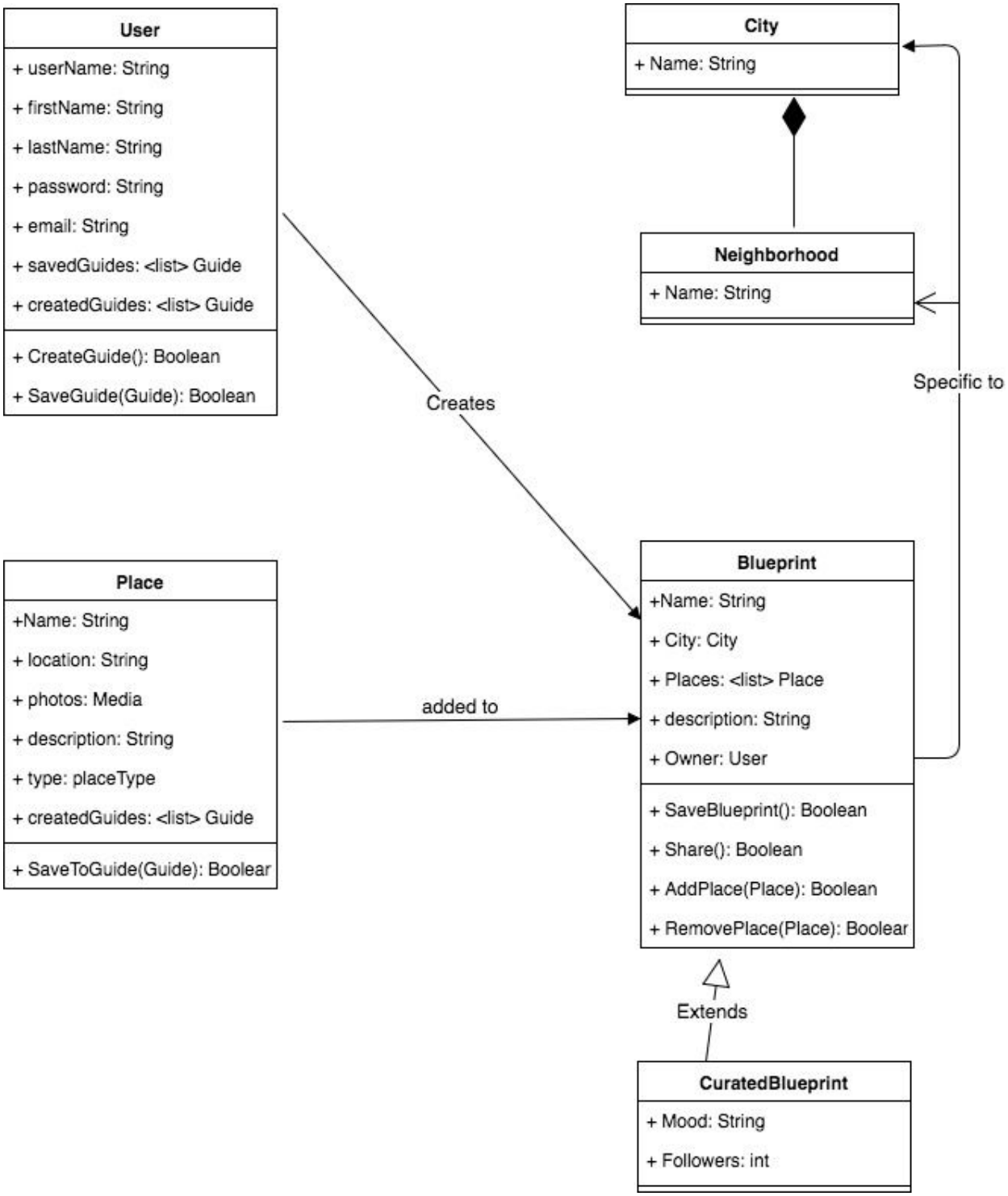


The *Explore* functionality is critical to the Caravan app. The Search screen will be accessible via the bottom bar on the app. When the user lands on the Search page, the keyboard will be brought up. As the user taps, the app will suggest autcomplete searches. If there are matching results, those are displayed to the user for them to explore. If there are no results, the user is notified and can try new search terms.

---

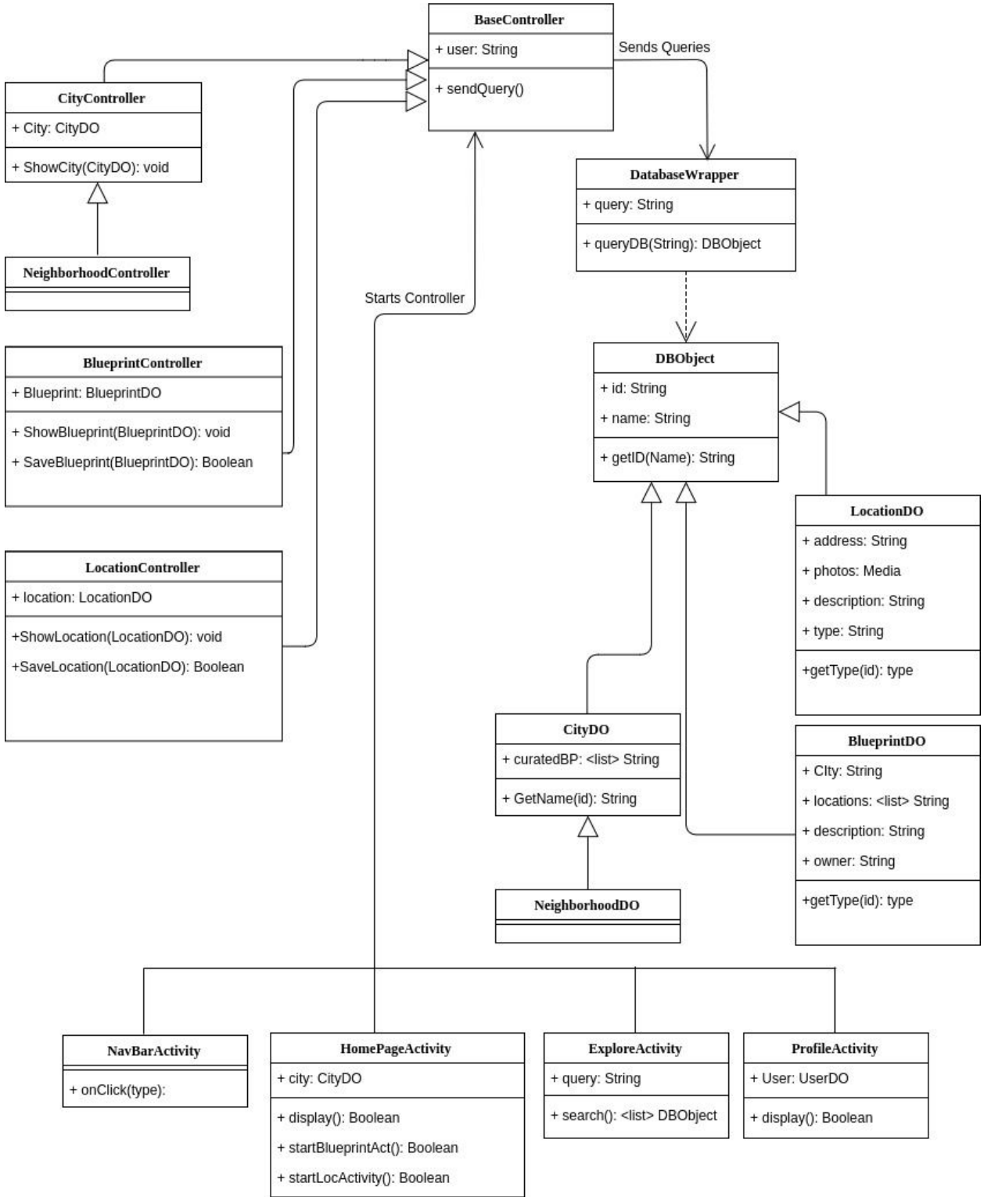
10. Class Diagrams

10.1 Requirements Phase



Above shows a high-level overview of class interaction in the Caravan App. Users can create and share blueprints. Places are added to Blueprints. The Curated Blueprint is an extension of the Blueprint class. Curated blueprints are mood specific and will like be created by some sort of verified user or by a member of the Caravan team. Each Blueprint is specific to a city and sometimes a neighborhood as well.

10.2 Design Phase



Above is the updated Class Diagram that has been refined from the original class diagram. The classes associated with app can be broken down into three categories: *Enities, Controllers,* and *Activities. Entities* are represented as *Database Objects* and include the Blueprints, Locations, Cities and Locations that are stored in the database. The objects will have member variables to reflect the fields in the database. The *Database Wrapper* will be responsible for accessing the database. Using the *Wrapper* will help the app be as independent of the database as possible. If the database is migrated to a different service or changes in structure, only the *Wrapper* class would need to be changed. Without the *Wrapper* any class that accessed the database would need to be updated to keep up with database changes.
The *Controller* classes will handle most of the logic and services the app offers. All showing, sharing and saving of *Database Objects* will be handled by the corresponding *Controller.* The *Controllers* all inherit from a *BaseController* Class that has an attribute to keep track of a logged-in user and a method for sending queries to the *DatabaseWrapper.*
The *Activities* are all of the classes that the app user interacts with and control the flow of data and activity through the application. The *Authentication* Activity makes use of Amazon Web Services Cognito and Facebook and Google to handle registration, sign in, and forgotten passwords. The *NavBarActivity* allows the user to change their view, between *Homepage, Explore and Profile*. The *NavBarActivity* launches a activity bar that is docked to the bottom of the screen through all views in the app. The *ExploreActivity* controls all of the logic for the search functionality where users can discover Cities, Neighborhoods, Locations and Blueprints. The *HomepageActivity* is the main browsing activity and will control populating the homepage of the app with curated blueprints and new locations the user may find interesting. The *ProfileActivity* will be the hub for maintaining a user's guides, both saved and created. The *Controller* classes can be started by any of the *Activity* classes and can create many different *DatabaseObject* instances depending on the user actions. Seamless interaction between all of the Caravan Classes is imperative to be able to complete the goal of having a responsive, clean, easy-to-use application.

---

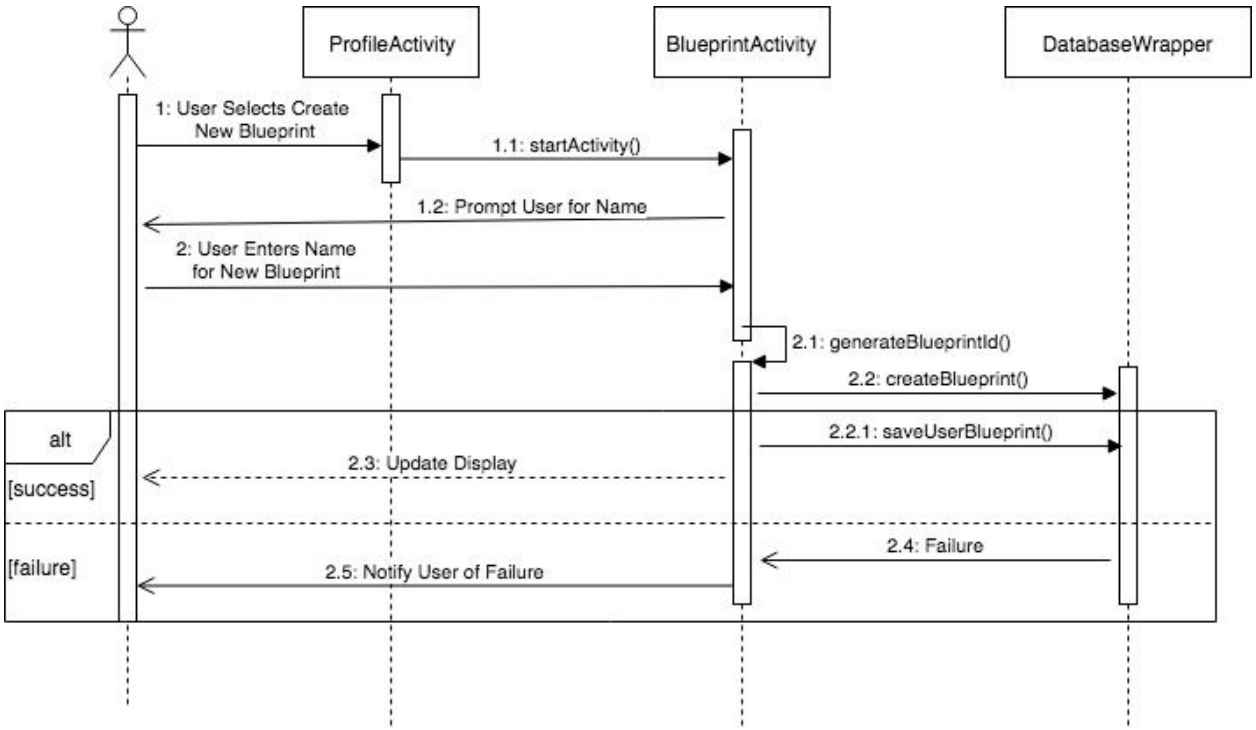
11 Sequences

11.1 Create a Blueprint

Blueprint creation is an essential feature of the Caravan Application. While the app will feature many curated plans based on moods and personality types, we also wanted to give travellers full flexibility by allowing them to create their own custom blueprints.

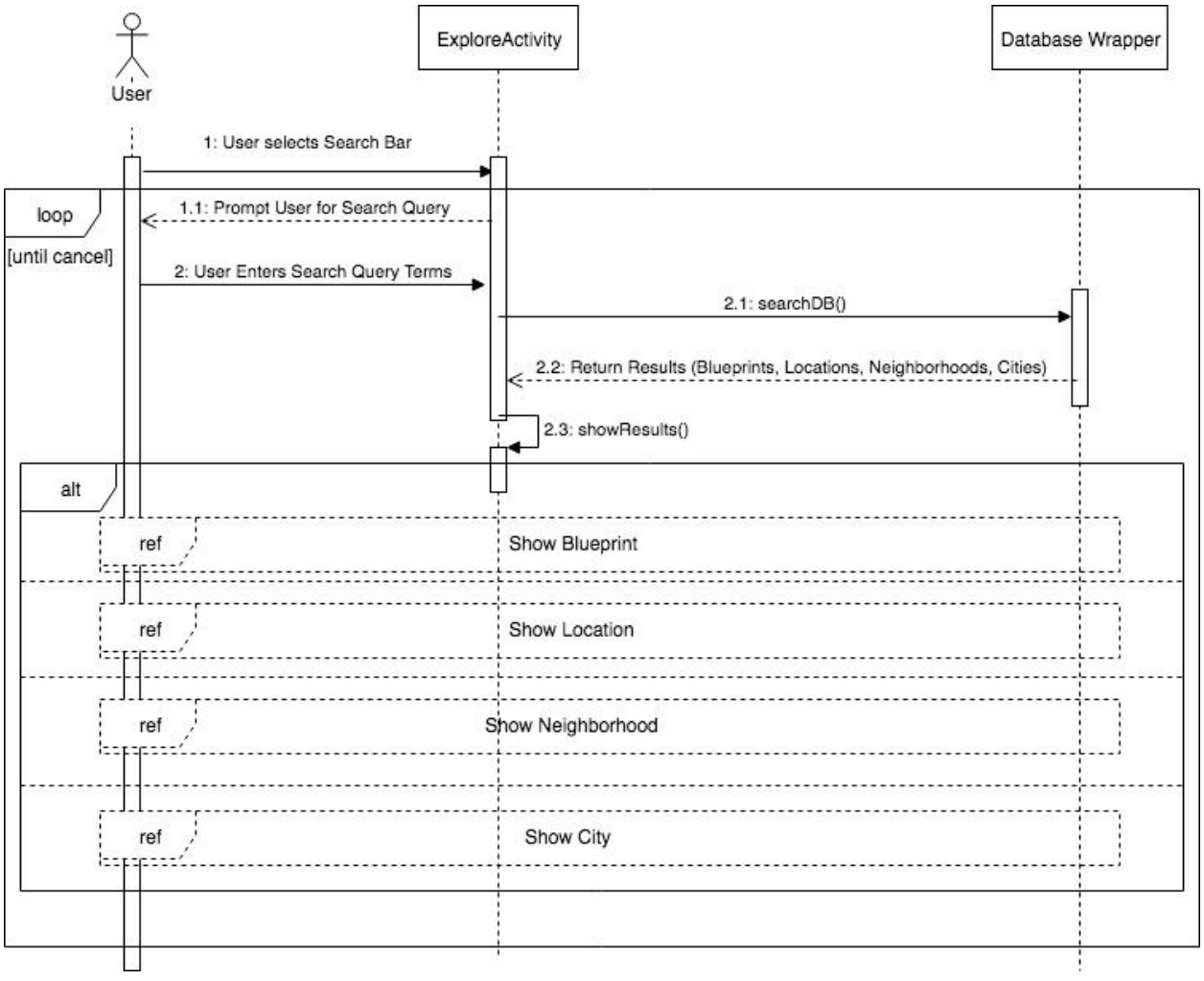
Because creating a blueprint will be a common activity within the app, we have aimed to streamline the process and make it as seamless for the user as possible.

The user will select the *Create a Blueprint* option from their *profile screen*. This will begin the Blueprint Activity which will pull up a widget to prompt the traveller for a name for their new blueprint. After receiving the user input, the *Blueprint Activity* will generate a unique ID for the new blueprint and call on the *DatabaseWrapper* to complete the creation process. The *DatabaseWrapper* will add the new blueprint to the database and save the user-provided information. If this process is successful, the user's display will be updated accordingly. If the *DatabaseWrapper* fails for any reason, the user will be notified and can manually restart the process to try again.



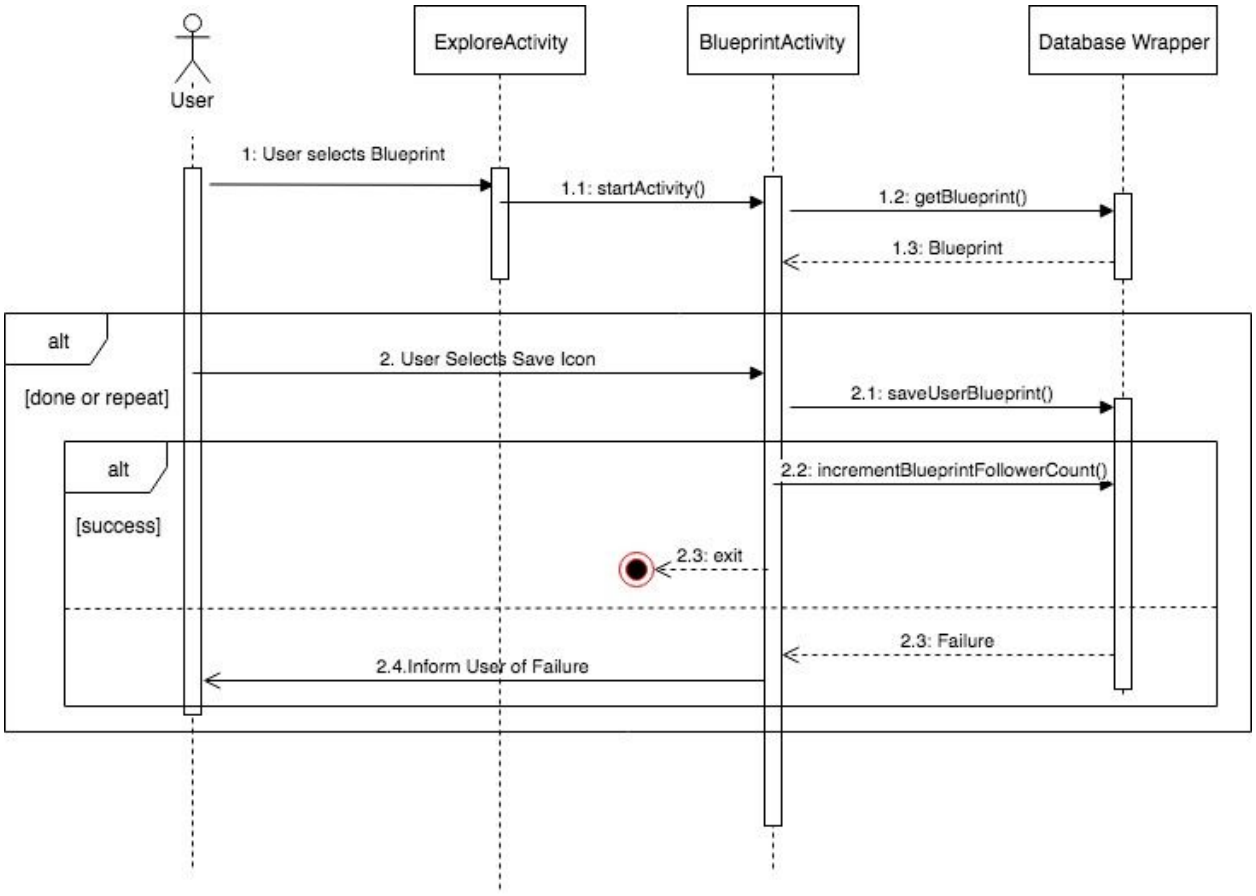
11.2 Explore

Finding new places to explore is another key activity within the app. To search for a location, blueprint, neighborhood or city, the user will tap on the *Explore* icon in the Navigation Bar. As the user enters a query into the search bar, the *ExploreActivity* sends the search terms to the *DatabaseWrapper*. The *DatabaseWrapper* returns the results to the *ExploreActivity* which displays them to the user. Cities, Neighborhoods, Locations and Blueprints are all searchable. The explore sequence is the starting point for many of the other sequences including *Show Location, Blueprint, City* and *Neighborhood.*



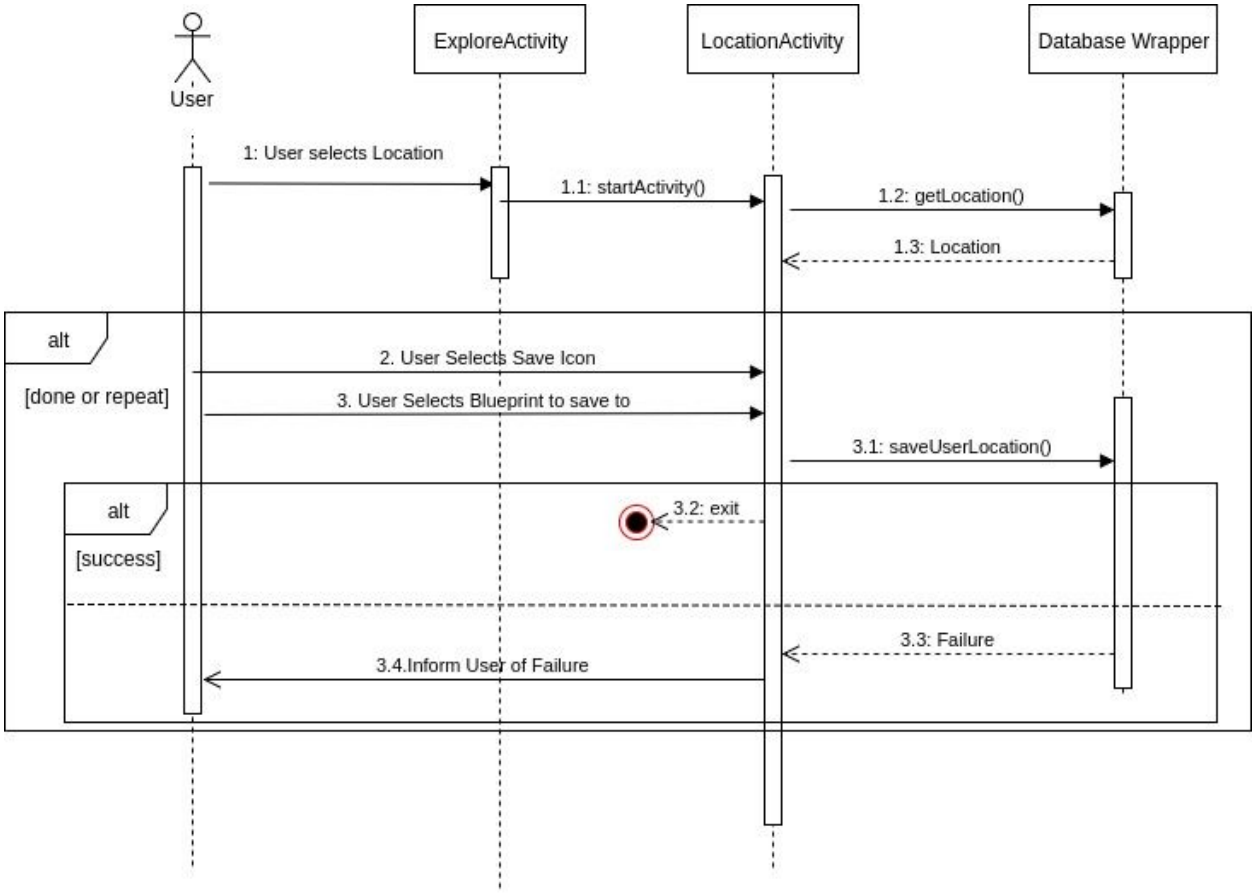
11.3 Save A Blueprint

Saving a blueprint is designed to be as simple as possible for the user. The sequence starts after the user has completed the *Show A Blueprint* sequence. A heart icon will be shown to the user on the Blueprint Display. Tapping the icon will start the *Save a Blueprint* Sequence. The *BlueprintActivity* calls on the saveABlueprint method that will add that Blueprint to the user's list of saved plans and call on the *DatabaseWrapper* to increment the follower count of the Blueprint. If successful, the sequence will terminate without notifying the user of the success. The user will be notified if there is a failure to save the blueprint.



11.4 Save A Location

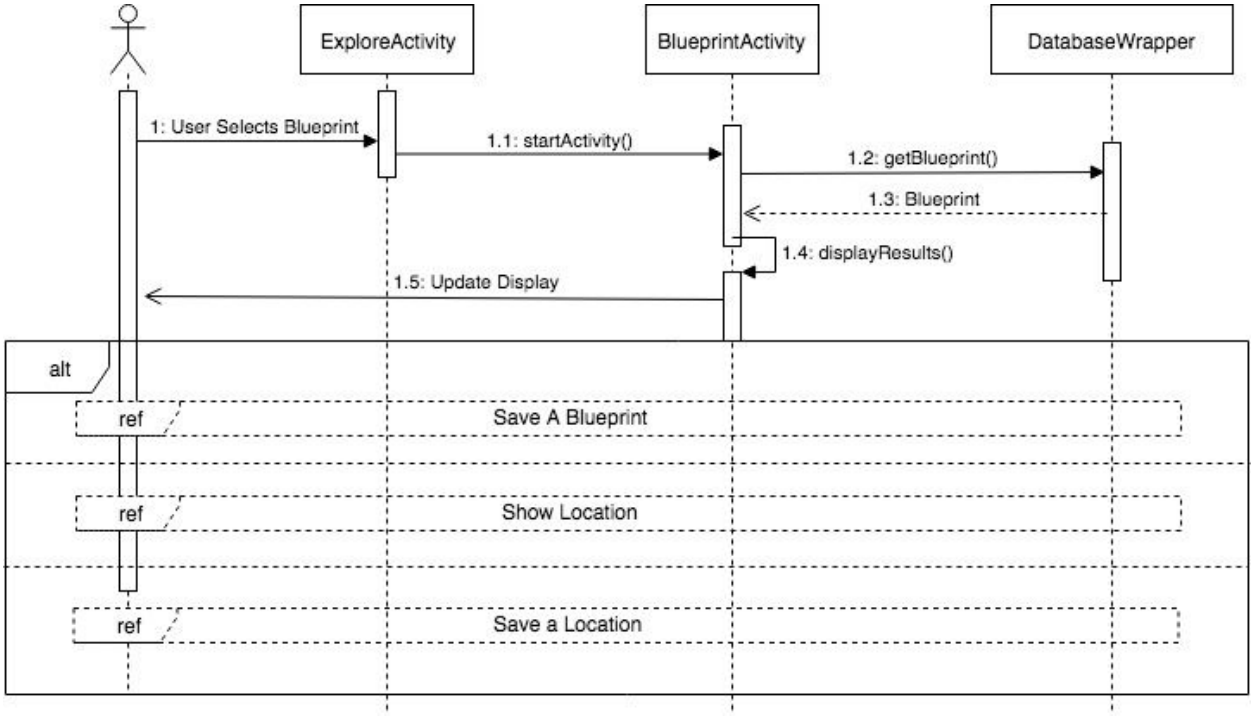
Similarly to *Save a Blueprint*, the user access the *Save a Location* sequence by first completing *Show a Location*. On the location screen, the user will tap on the + icon to save the location. The *LocationActivity* will prompt the user to select a blueprint they'd like to save the location to. The *LocationActivity* will call on the *Database Wrapper* to add the location to the desired blueprint in the database. If successful, the sequence will terminate without notifying the user of the success. The user will be notified if there is a failure to save the blueprint.



11.5 Show Blueprint

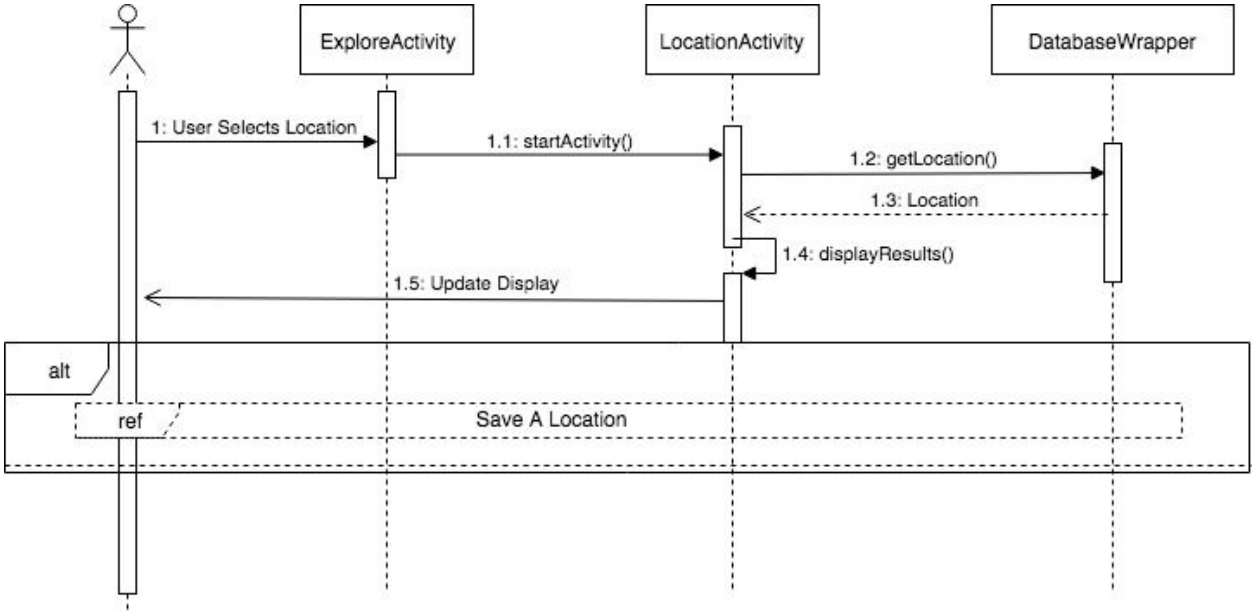
The *Show a Blueprint* is one of the foundations of the application. The sequence starts when the user taps on a thumbnail for a blueprint; this can happen on the homepage of curated and suggested guides, after utilizing the explore feature and search for a blueprint, or the user's profile page which lists the user's saved blueprints. Once a blueprint is selected, the *BlueprintActivity* begins. The *BlueprintID* is

sent from the *BlueprintActivity* to the *DatabaseWrapper* and the details of the selected blueprint (name, description, location list etc.) are retrieved and returned to the *BlueprintActivity*. The *displayResults* method is called and the user's display is updated. From there, the user can move through the *Save a Blueprint, Save Location* and *Show Location* Sequences.



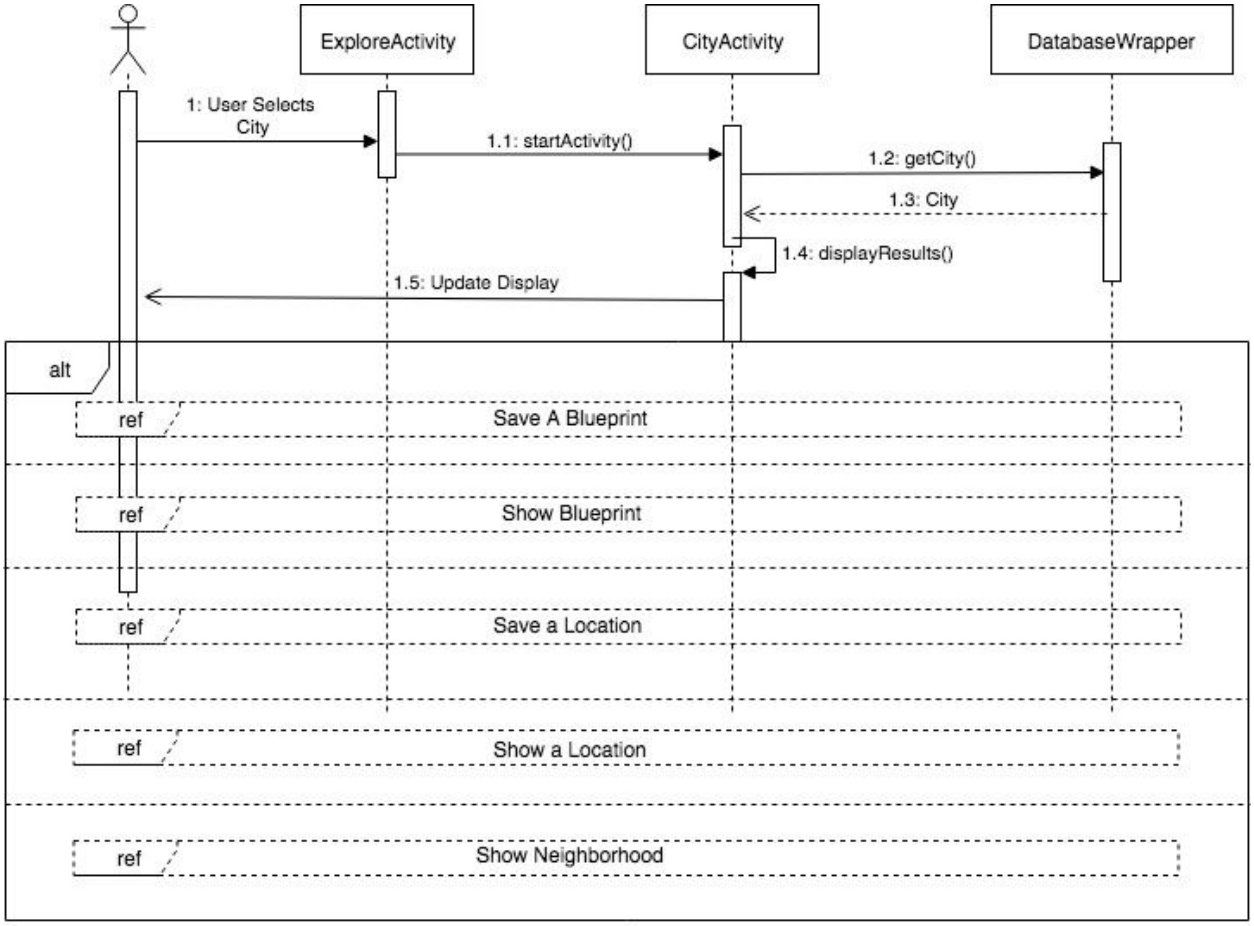
11.6 Show Location

The *Show Location* Sequence is very similar to that of *Show Blueprint*. Sequence starts either through the homepage, explore activity or the *Show Blueprint* Sequence. The user selects a location to display and the *LocationActivity* is started. *LocationActivity* sends the ID of the selected location to the *DatabaseWrapper* which retrieves the details of the location. The results are displayed to the user by the *LocationActivity.* From here, the user can read through the details of the selected location or continue to the *Save a Location* sequence.



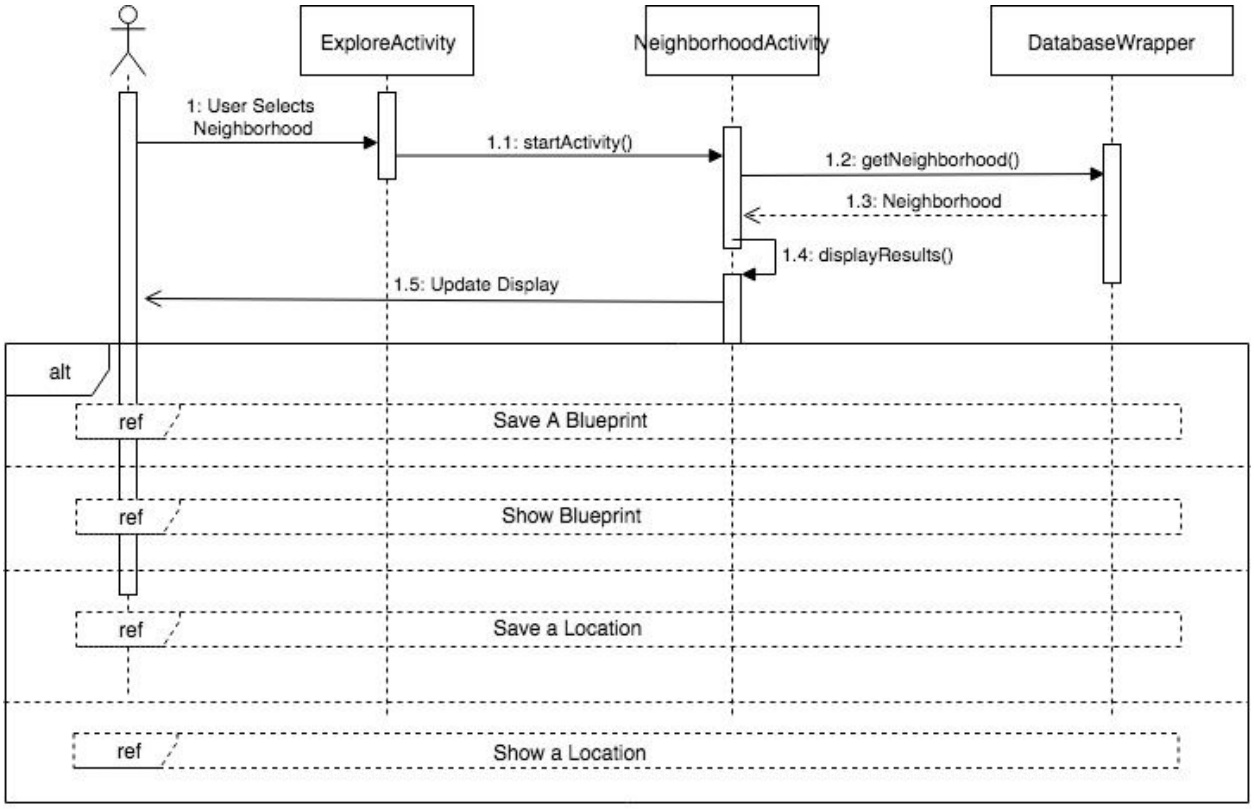
11.7 Show City

The *Show City* sequence is very similar to the other *Show* sequences. The user begins the *ShowNeighborhood* sequence from the homepage or explore activity. The user selects a City to be displayed and the *CityActivity* sends the ID to the *DatabaseWrapper* which retrieves the details of that city. The *CityActivity* then displays the results. The sequence will display a page similar to the application's main homepage. It will show featured neighborhoods within that city, a list of curated Blueprints specific to the city and a list of fun and interesting locations for the user to explore.



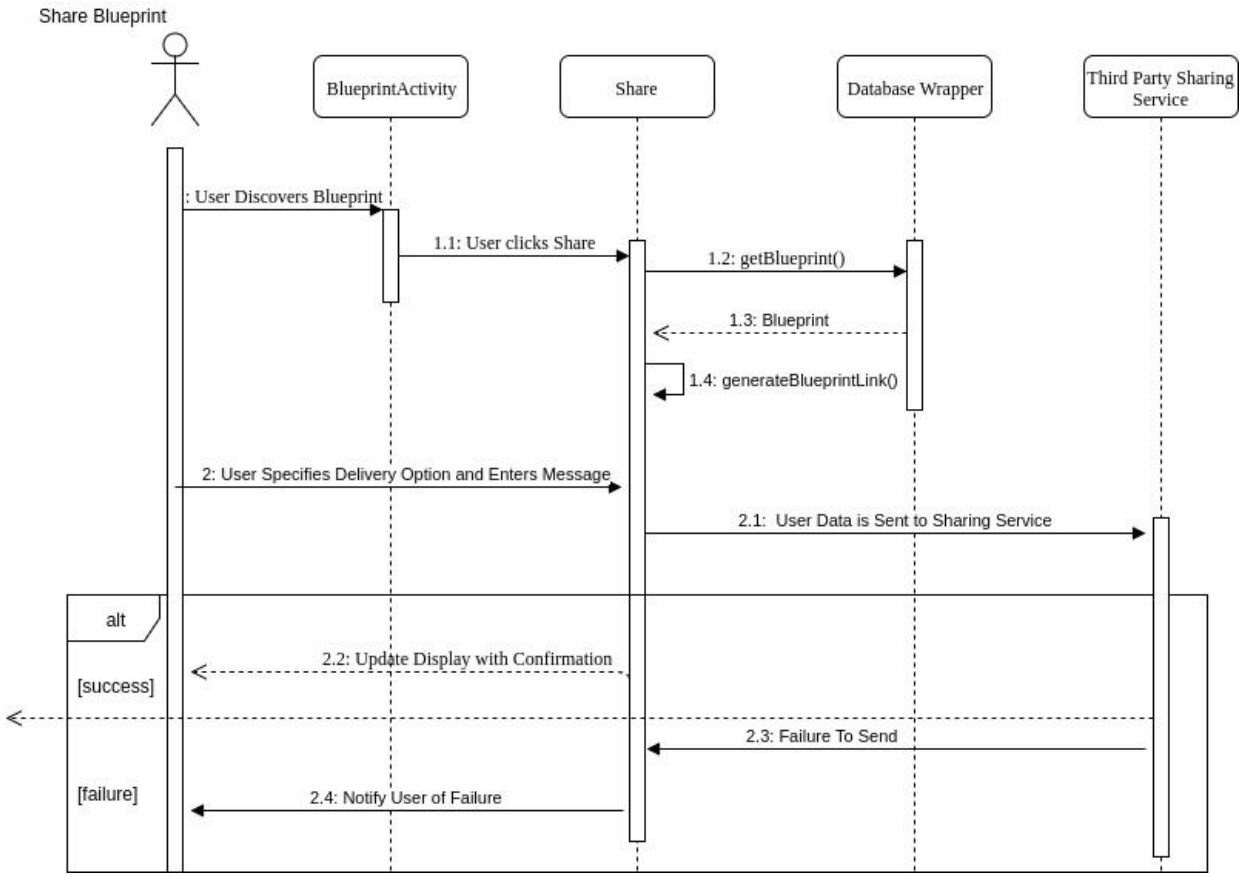
11.8 Show Neighborhood

The *Show Neighborhood* sequence is very similar to the other *Show* sequences. The user begins the *ShowNeighborhood* sequence from the homepage or explore activity. The user selects a Neighborhood to be displayed and the *NeighborhoodActivity* sends the ID to the *DatabaseWrapper* which retrieves the details of that Neighborhood. The *NeighborhoodActivity* then displays the results. The sequence will display a page similar to the application's main homepage. It will show a description of that Neighborhood, a list of curated Blueprints specific to the Neighborhood and a list of fun and interesting locations for the user to explore.


ExploreActivity
NeighborhoodActivity
DatabaseWrapper
1: User Selects Neighborhood
1.1: startActivity()
1.2: getNeighborhood()
1.3: Neighborhood
1.4: displayResults()
1.5: Update Display
alt
ref
Save A Blueprint
ref
Show Blueprint
ref
Save a Location
ref
Show a Location

11.9 Share Blueprint

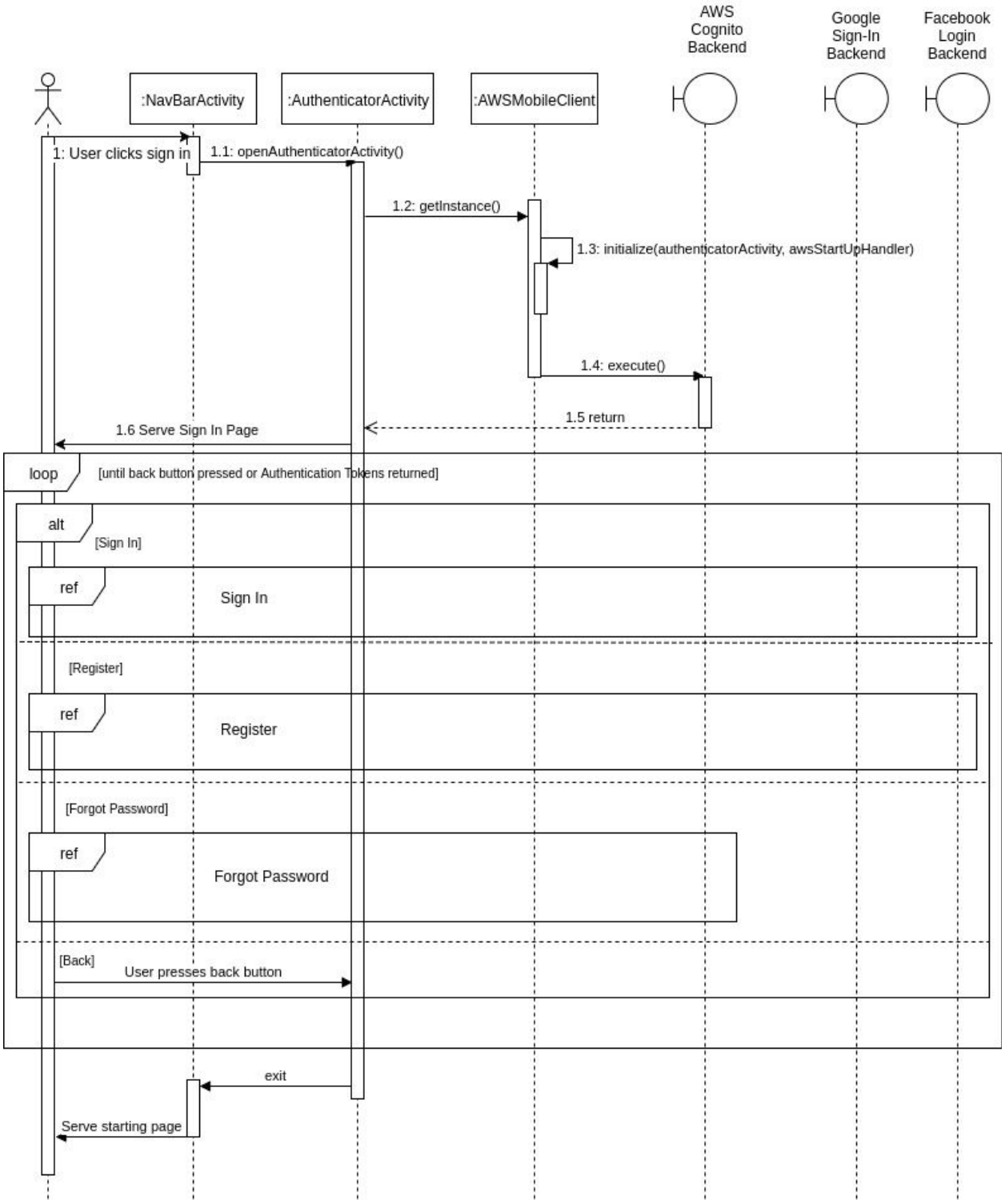
The user begins this sequence after completing the *Show Blueprint* Sequence. The user select the option to share the Blueprint. *BlueprintActivity* then generates or fetches a unique URL that links directly to the Blueprint. The user will be prompted to choose how they'd like to share the Blueprint (text message, Facebook, email etc). Based on the user selection, the user will be linked out to that platform to write a message to go along with the blueprint link and confirm their post. The sharing sequence is key to growing the Caravan user base, so it's designed to be simple while also allowing users to customize the post they share.



11.10 Authentication

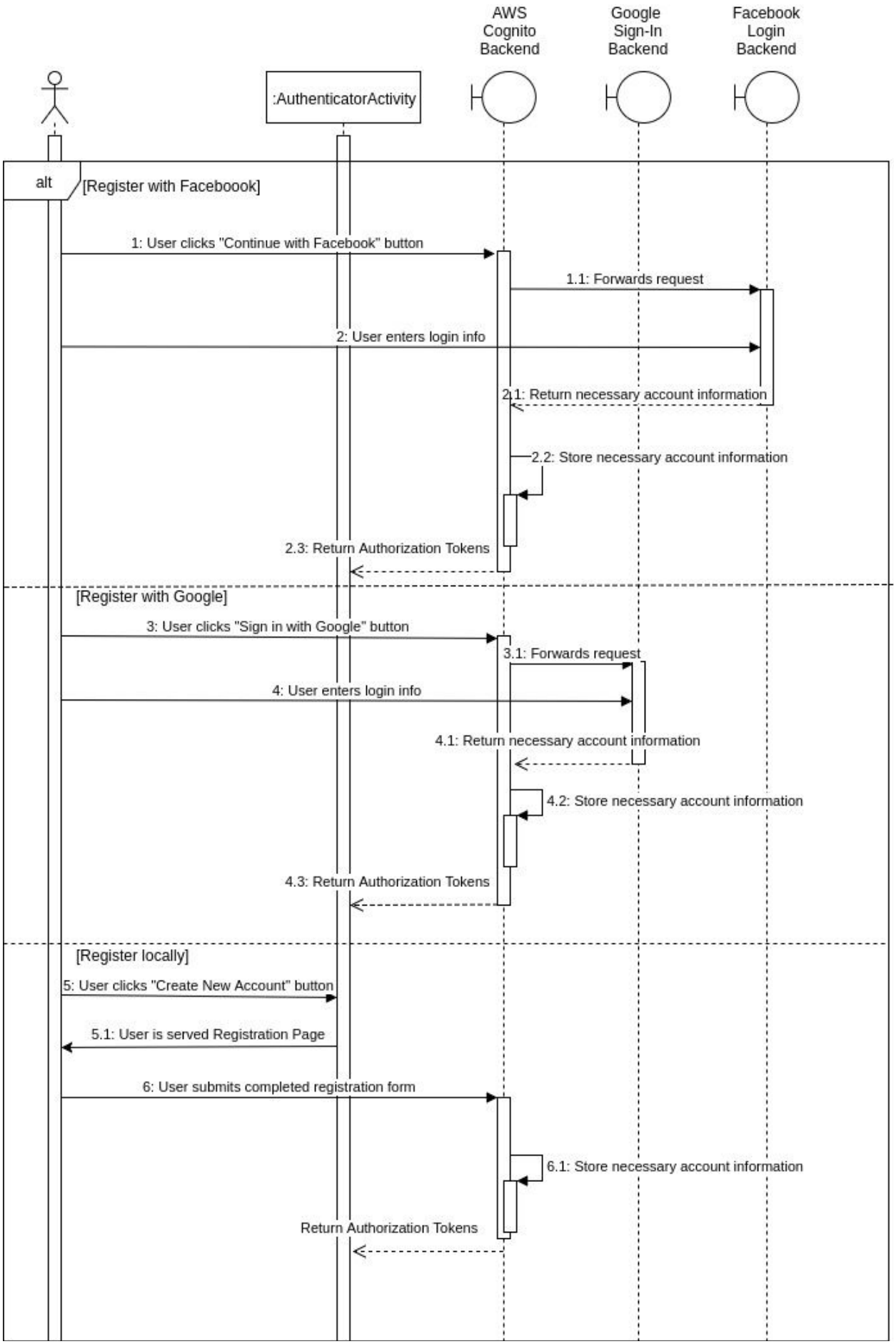
The Authentication Sequence is one of the very first steps in using the Caravan Application. When the user selects the *Sign In* icon, the *AuthenticatorActivity* is started. The *AuthenticatorActivity* starts an *AWSMobileClient*

instance. Once the instance is in initialized and the call to *AWSCognito* executes, the user is brought to the Sign In page. From there, the user can Register, Sign In or Recover their password.



11.11 Register

The Register Sequence caters to those users who do not have Caravan accounts and wish to sign up. There are three different registration options that all follow similar logic. The first option is to register with an existing Facebook Account. To do so, the user selects the “Continue with Facebook” option. The AuthenticationActivity sends the request to the Facebook API. The user enters their Facebook login credentials and those are validated by Facebook services. The Cognito service stores account information returned from Facebook and the user is brought to the Sign In page.
Registering for the app with a Google account follows the same logic as doing so with a Facebook account, but calls on Google’s services instead of Facebook’s. After selecting “Continue with Google”, the user’s Google account credentials are entered and verified and any important account information is stored for future Sign In activities. Once completed, the user is brought to the Sign In page to log in to their account.
If a user does not want to connect their Facebook or Google accounts to the app, they also have the option to register locally. To do so, the user would choose the “Create a New Account” Option. The user is directed to the Registration page where they complete a form to gather email, password, name, and other important account credentials. The user’s credentials are sent to Cognito for verification. Once registered, the user is directed to the Sign In activity.

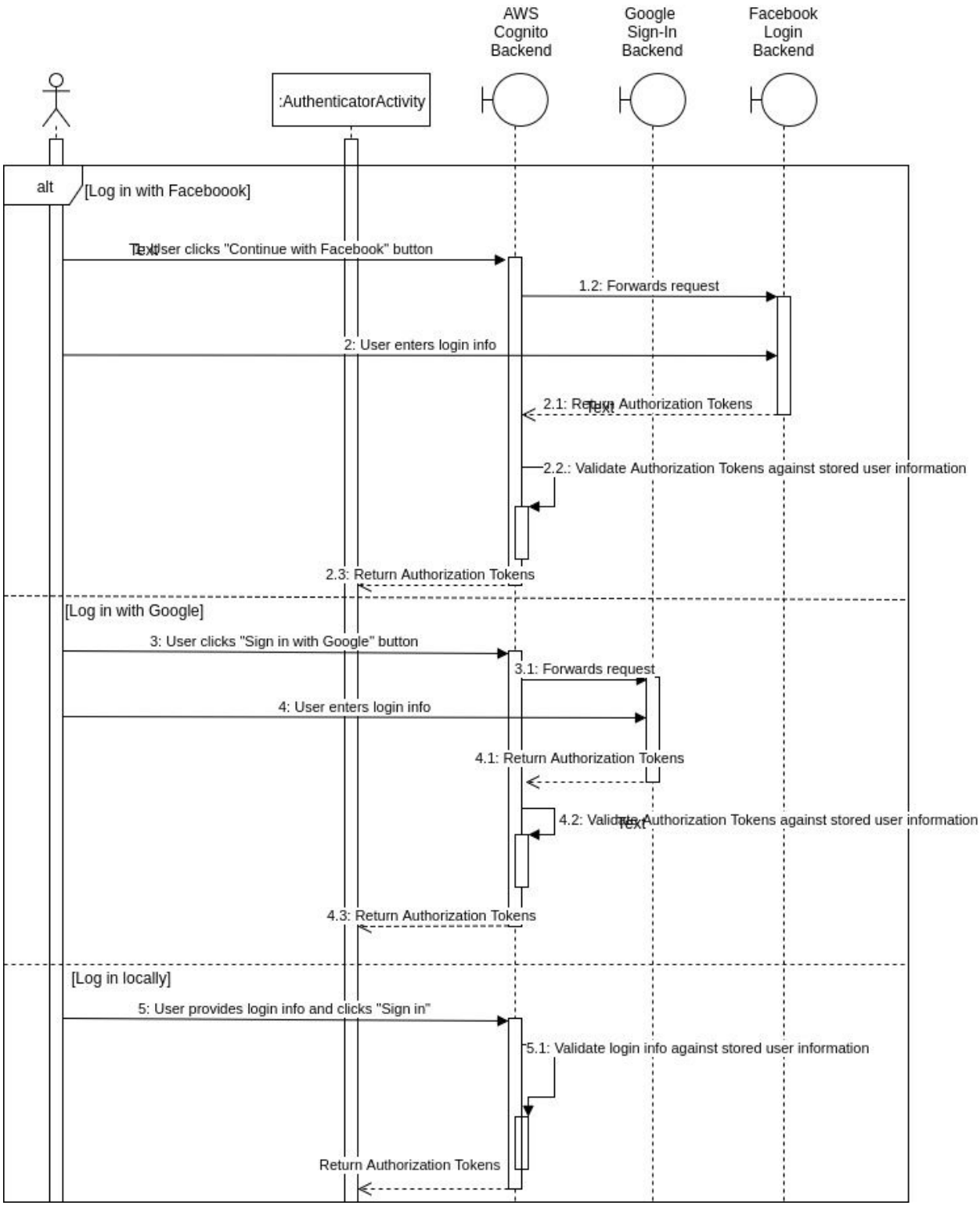

AWS Cognito Backend
Google Sign-In Backend
Facebook Login Backend
:AuthenticatorActivity
alt
[Register with Faceboook]
1: User clicks "Continue with Facebook" button
1.1: Forwards request
2: User enters login info
2.1: Return necessary account information
2.2: Store necessary account information
2.3: Return Authorization Tokens
[Register with Google]
3: User clicks "Sign in with Google" button
3.1: Forwards request
4: User enters login info
4.1: Return necessary account information
4.2: Store necessary account information
4.3: Return Authorization Tokens
[Register locally]
5: User clicks "Create New Account" button
5.1: User is served Registration Page
6: User submits completed registration form
6.1: Store necessary account information
Return Authorization Tokens

11.12 Sign In

Similar to the *Register* Sequence, there are three ways the user can sign into their Caravan Account. The first is through Facebook. If a user registered their account with Facebook, they can also sign in through the social media platform. To do so, the user chooses the Facebook option on the Sign In screen. The request is forwarded from Cognito to Facebook. The user enters their Facebook credentials, which are authenticated by Facebook and Authentication tokens are checked against the user information currently stored. If the sign in fails, the user will be notified and can try again.

The user can also sign into Caravan using their Google account. To log in with Google, the user would select "Continue with Google" from the sign in options. Similar to the Facebook Sign In, Cognito will send the request to Google and the user will enter their credentials. The credentials are verified by Google which will return authorization tokens to be checked against user information. If the sign in is successful, the user will be brought to the Caravan Homepage. If there is a failure, the user will be able to try again.

The final way to sign in is through a local Caravan app that is not connected to Facebook or Google. The user enters the credentials which are verified by AWS Cognito. If successful, the user continues to the homepage. If the process fails, the user is notified and can try again.


AWS Cognito Backend
Google Sign-In Backend
Facebook Login Backend
:AuthenticatorActivity
alt
[Log in with Faceboook]
1: User clicks "Continue with Facebook" button
1.2: Forwards request
2: User enters login info
2.1: Return Authorization Tokens
2.2.: Validate Authorization Tokens against stored user information
2.3: Return Authorization Tokens
[Log in with Google]
3: User clicks "Sign in with Google" button
3.1: Forwards request
4: User enters login info
4.1: Return Authorization Tokens
4.2: Validate Authorization Tokens against stored user information
4.3: Return Authorization Tokens
[Log in locally]
5: User provides login info and clicks "Sign in"
5.1: Validate login info against stored user information
Return Authorization Tokens

11.12 Forgot Password

Similar to the Authentication Sequence, the Forgot Password Sequence utilizes the AWS Cognito Service. The user navigates to the Account Sign In Page and selects the “Forgot your Password?” option. The user provides their unique username. Cognito verifies the username and sends an email with a unique confirmation code to the address associated with the username. The user is brought to the Forgot Password Page and prompted for the email code and a new password. Cognito updates the user’s information if it is valid. The user is directed to the Sign In page where they can begin the Sign In Sequence.