# Cognac Bandits - SENG2021 Deliverable 2

## Part 1: Software Architecture

### External Data Sources

The main external data source the web application will be using is the free *TheCocktailDB* API.

The API has features which allow the user to search through a database of cocktails and filter through it. There are methods to search the database by cocktail name and a method to search by cocktail ingredient. These methods help in satisfying the first user story referring to the user wanting to search a particular cocktail or ingredient. The API also can look up full cocktail details which satisfies the second user story referring to searching for details on how to make a specific cocktail.

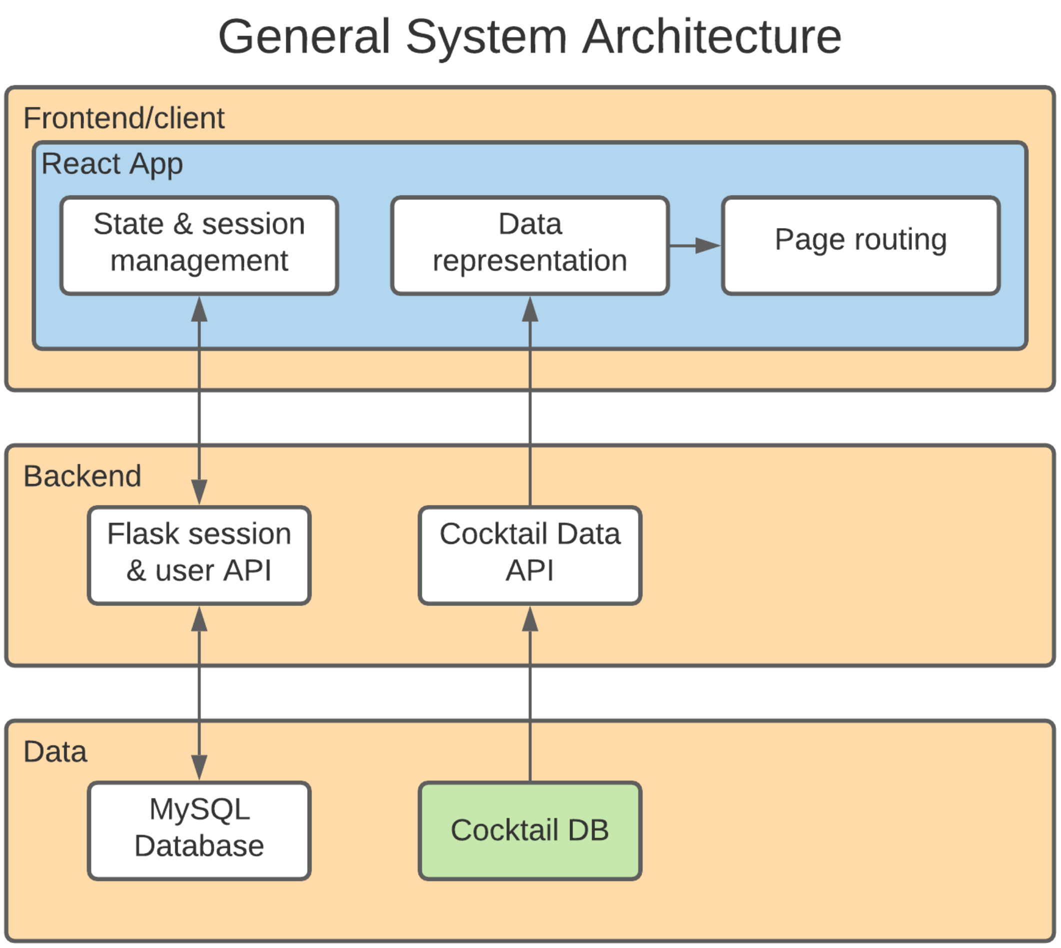
### Software Components (Web Stack)

The software components architecture gives an abstract representation of the system as a whole and the primary communications between each of the significant software components.

*Figure 1* interlays said significant software components into the desired architecture of the system, with arrows denoting abstract data flow. This diagram gives an idea of the structure but more detail on each component will be described in the following sections.

# Cognac Bandits - SENG2021 Deliverable 2

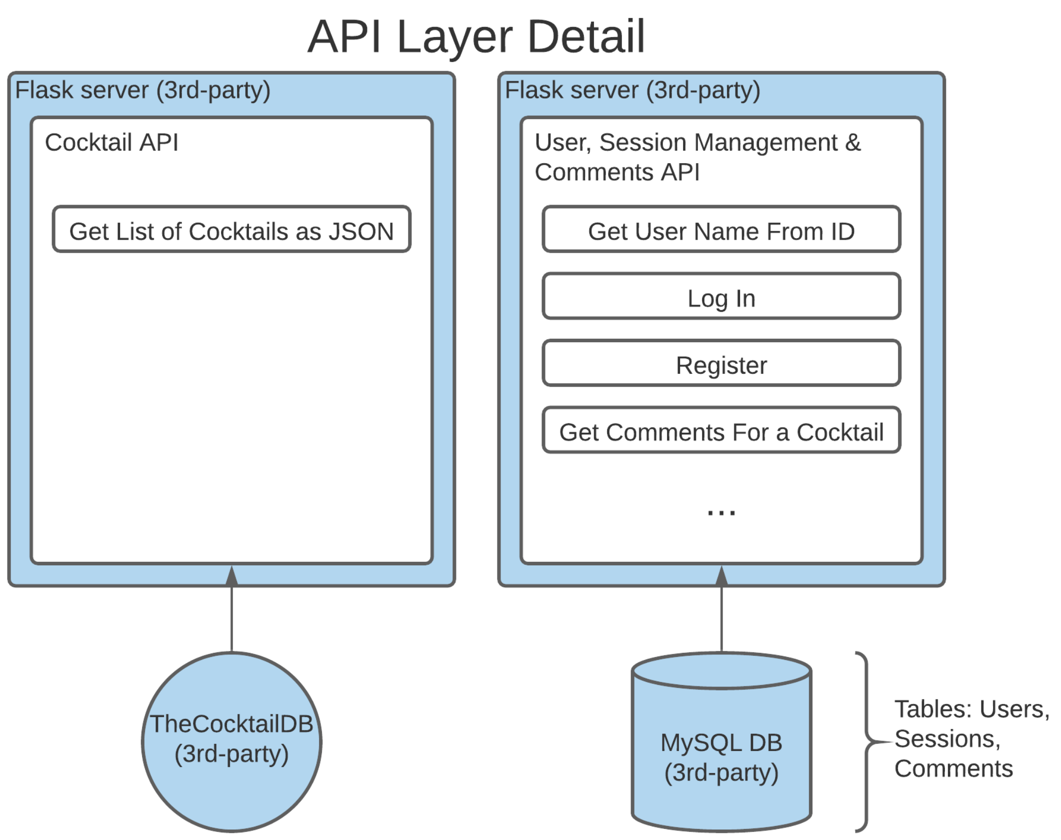
*Figure 1*



## Backend / API Layer Detail

This section will give further detail on the API layer of the software, namely the servers that run independently of the client web application. The API layer contains two Python Flask servers, each providing their own functionality.
*Figure 2* gives a general representation of the two servers, however, more detail will be provided in the following sections.

*Figure 2*


API Layer Detail
Flask server (3rd-party)
Cocktail API
Get List of Cocktails as JSON
Flask server (3rd-party)
User, Session Management & Comments API
Get User Name From ID
Log In
Register
Get Comments For a Cocktail
...
TheCocktailDB (3rd-party)
MySQL DB (3rd-party)
Tables: Users, Sessions, Comments

### Cocktail API

The Cocktail API will be the primary source of Cocktail data to be delivered to the user. This includes a wide range of cocktails with ingredients, a short description and the steps to follow to create the cocktail. This data is to be collected from *TheCocktailDB*, a public REST API providing cocktail information. This API gives access to 100 cocktails with the required data (ingredients, recipe, etc). The reason this API has been chosen by us is in large part because it is free. Furthermore, it follows an intuitive REST format which will reduce the time spent in development learning how to interface with it.

Our cocktail API can be seen as a mere wrapper for *TheCocktailDB*, providing only a minimal subset of the data and organising the information into a consumable JSON format. In particular, our Cocktail API will select cocktail name, ingredient, recipe data of

100 cocktails from *TheCocktailDB*, and format the data as JSON before delivering it to the client application.

### User, Session Management & Comments API

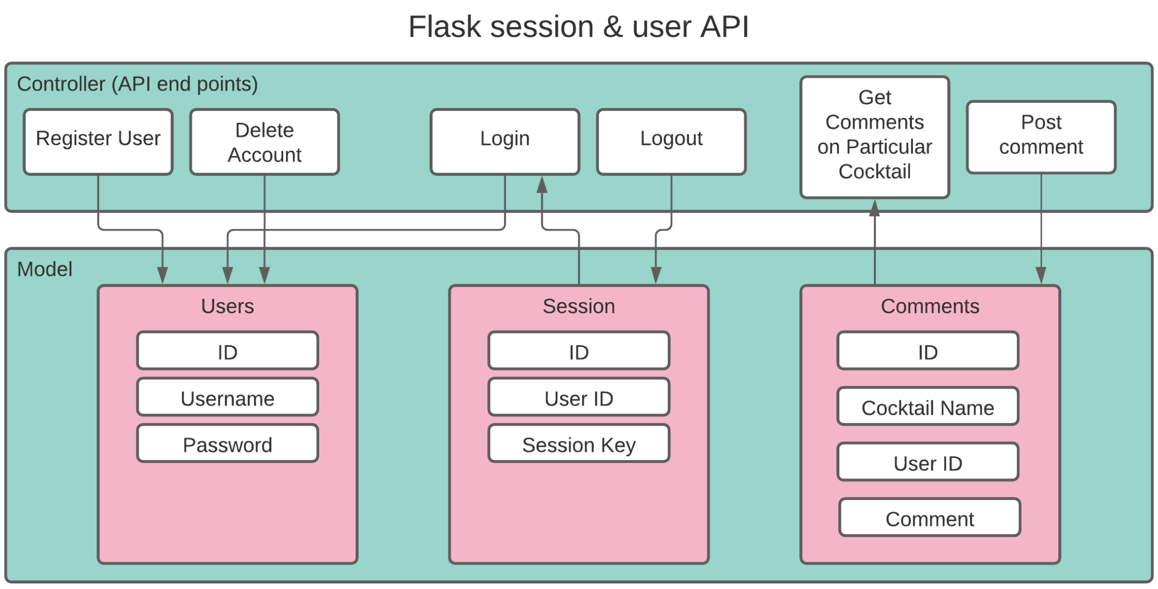
The second API server provides dynamic utility largely pertaining to users. This API will follow a REST structure. A REST structure will help separate user, session and comment data and functionality into cohesive routes.

One area of concern is the potential of high coupling to arise in placement of the commenting system. Primarily, comments are both related to cocktails (they are comments about cocktails), and users (users post comments under their username). However, comments functionality isn't placed in the Cocktails API in which cocktail names are managed. We have grouped our commenting functionality into the latter API since cocktails names are static, so a coupling between comments and cocktail objects in the Cocktail API needn't be developed. All in all, coupling should not arise if this software structure is maintained.

*Figure 3* gives a general representation of the API in terms of the Controller and Model. The Controller will handle REST routes such as "users/login", "comments/post/", etc. Furthermore, the controller will delegate the procedures to follow in order to execute the appropriate task (e.g. log in: validate user credentials and create a user session).
The Model will represent database information as Python objects to allow simple interfacing between the controller and the database. However, SQL logic will reside in the Model in order to retrieve, create, delete and modify data entries in the database.

# Cognac Bandits - SENG2021 Deliverable 2

*Figure 3*



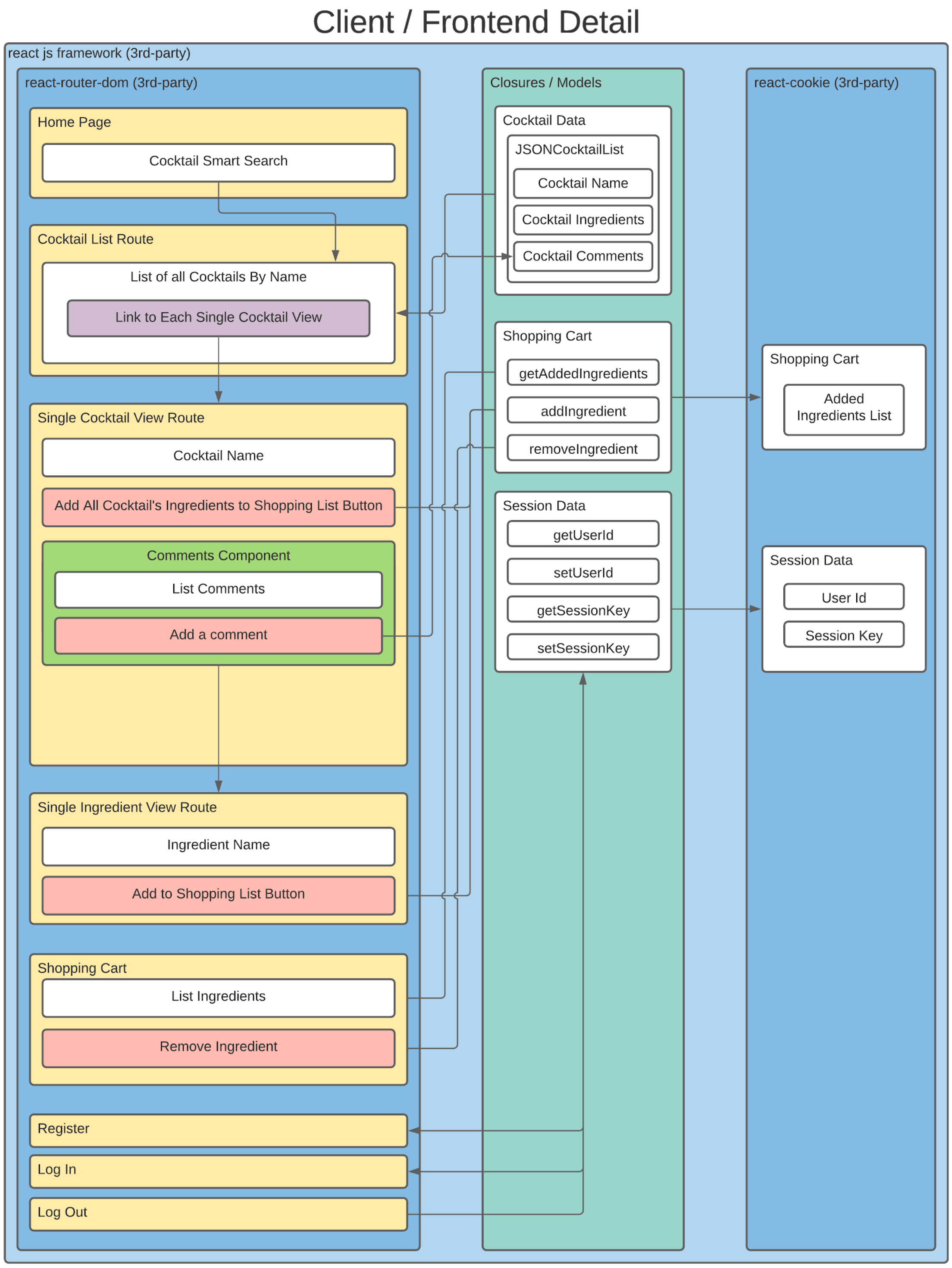
## Frontend / Client Detail

The client application functions as a React application. ReactJS is a front end JavaScript library used for building user interfaces. React can be used as a base in the development for single-page applications or mobile applications. React applications usually require the use of additional libraries for routing.

*Figure 4* gives a visual on the components of the client. In representing the client application, we can separate the client into 3 distinct - albeit abstract - sections. These include the pages, closures and cookies

# ***Cognac Bandits - SENG2021 Deliverable 2***

*Figure 4*



The pages are associated with a particular URL which directs the user to the correct page via the router (*react-router-dom)*. These pages allow navigation amongst each other and logical separation of UI/UX. The pages largely are mere representations of data collected from the closures/models.

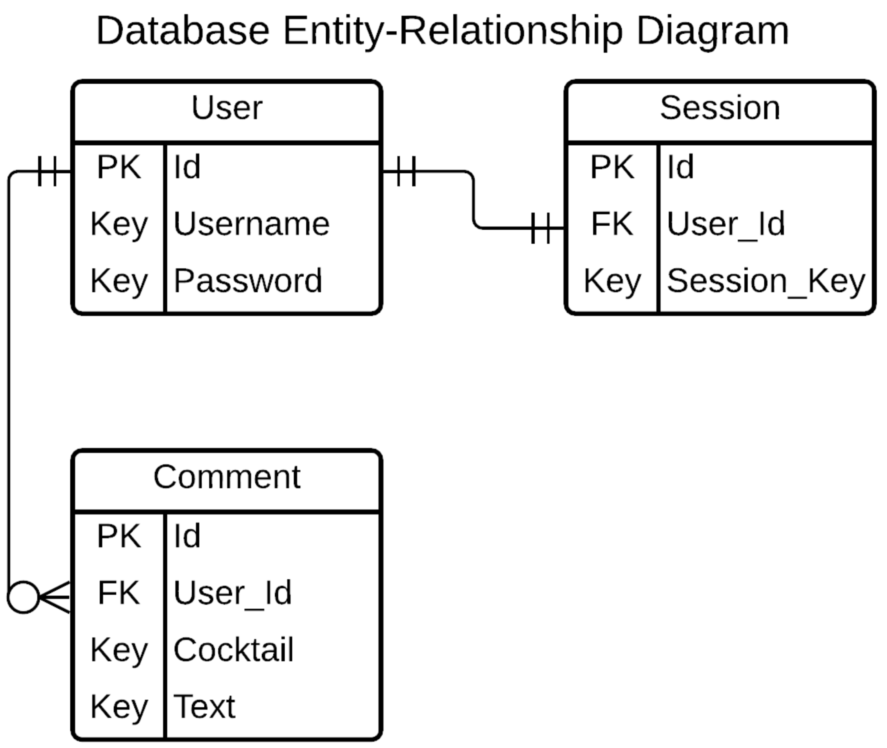
Great consideration has been placed into the methods in which data is stored and managed in the client. It is very important that we have a centralised state container for the websites in order to store vital and all-important information such as cocktail data, the shopping cart and session data. This is important since all of this information needs to be accessible by all pages and components on the website.

Significant is the comparison between using *Redux* alongside *redux-react-session,* and *react-cookie* implemented via closures.
*Redux* is a widely accepted solution to a centralised state container for *React* applications. This would work in combination with *redux-react-session* which provides further utility for storing user sessions. However, *Redux* is a large package with a complex API with more features that we need. Using *Redux* could result in large increases in development time and time learning *Redux's* API. This is why we have chosen a more straight-forward approach, that is using a small package *react-cookie,* a simple API for web cookies. We use this within closures, methods that represent state and that are accessible to the entire application. These closures will have procedures for getting a setting data to be stored website-wide in web cookies. In addition, the closures would encapsulate API integration with our backend servers, namely the Cocktails API and the User, Session management & Comments API.

## Database Detail

The database consists of a Users, Sessions and Comments table. The Users table stores the user information (username and password). The Comments table stores the comments posted on cocktail pages by specific users. Finally, sessions store the information needed to allow users to stay logged in on the client.

*Figure 5*



## Relating Choices to Components

Our application will be using ReactJS for the front end with Bootstrap for the styling, Python for the back end, SQL for a database structure and Flask as a web framework.

We have decided to use React since many members of our team have previously used it. That being said, not everyone in our team has experience with the Javascript language so a simple javascript framework will aid in allowing quick development. React allows us to use web components which significantly reduces duplicate code and helps with maintainability via encapsulation of logic and the ability to use a component in more than one place. React also allows us to implement routing with the third-party package *react-router-dom*. This allows us to logically separate our pages despite the website being a single-page application.

Bootstrap is a free CSS framework directed towards front-end web development utilising CSS and javascript designs for topography, forms, buttons and navigation. Bootstrap features a responsive web design approach which allows web pages designed with it to render well on a variety of devices such as mobile devices as well as different window or screen sizes. All of these features will allow us to make an appealing front end that caters for a wide range of users.

Python is a well rounded language making it perfect for creating the back end of the web application. We have decided to use the Python Flask framework since the majority of our team have used it before. Flask is largely a minimal framework being described as a “micro web framework”. This will allow us to produce quick prototypes during development. In particular Flask only provides the bare necessities for our web application including a server runtime and route handling, with the potential to add database and external API interfacing.

We decided to split the backend functionality into two separate API servers because our backend consists of two cohesive sets of functions, one being the management of static cocktail data and the other being web app utility including the management of user, session and comment data.

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system. It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables. Firstly, it introduced the concept of accessing many records with one single command. Secondly, it eliminates the need to specify how to reach a record, e.g. with or without an index. This decision to choose SQL, allows us to easily handle data and implement a database for storing users and comments for our web application.

SQL is perfect for modeling the logic for the login and commenting system in the web application. The Database will store information regarding users, sessions and comments. We have decided to use a MySQL server since members of our team have experience with SQL, as well as SQL befitting our minimal and finely structured data requirements.

Flask is a micro web framework written in Python. Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

## Choice of Platform

The web application we are building will be designed to work with UNIX like platforms such as macOS, Windows and Linux, with the main requirements of having an internet connection and a web browser.

For Windows users, for complete compatibility the web application is designed for Windows 7, Windows 8, Windows 8.1, Windows 10 or later and an Intel Pentium 4 processor or later that's SSE3 capable. With macOS support, OS X El Capitan 10.11 and later releases will be required for the web application. Finally for Linux, the web application is compatible with Ubuntu 14.04+ and Debian 8+. These are the platform requirements for current support Chromium-based browsers, hence the choice of platform for our web application.

The preferable web browser would be Google Chrome, as the application is being tested on Chrome browsers. However, other Chromium based browsers would work such as Opera, MS Edge, Vivaldi and Brave. Other non-Chromium based browsers such as Safari are tested for basic fundamental functionality of the web application, but not guaranteed to have supported formatting with alignments and fonts.

Our web application does feature support for mobile devices, meaning the pages are dynamic to the window size and shape of the browser. As both Android and iOS have support for Chromium-based browsers, the web application will be compatible with these operating systems. Specifically, at least Android Lollipop 5.0 and iOS 12 to be able to run a browser compatible with the web application.

As a result of implementing support for mobile devices, the application will be able to adapt to different screen sizes and resolutions.

# ***Cognac Bandits - SENG2021 Deliverable 2***

## Summary Table

| **Architecture element** | **Choice** | **Benefit/Achievement** |
|---|---|---|
| External data sources | TheCocktailDB(API) | Provides sufficient details on the information including ingredients and instructions on most cocktails. |
| CSS framework | Bootstrap | Abundant resources are accessible, easy for beginners, UI style meets our requirements. |
| JavaScript library | ReactJs | Simple to use, team has pre-existing experience with React |
| Web framework | Flask | Team has experience in developing application with Flask, simple and flexible for beginners, |
| Relational database management system | MySQL | Structured database design is befitting to our minimal data architecture. |
| Platform | Windows, macOS, Linux (Chromium-based browser supported) Mobile operating system (iOS, Android) | Browsers e.g. Google Chrome is available on all systems, widely used, cross-platform. |
| Backend programming language | Python | The programming language for Flask, open-source, cross-platform, many useful libraries for development. |

# ***Cognac Bandits - SENG2021 Deliverable 2***

## Part 2: Initial Software Designs

### Updated User Stories

#### Story #1:
**Feature:** Search cocktail by ingredient

**As a:** Cocktail maker

**So that:** I can find a particular cocktail containing ingredient

**I want to:** search cocktails based on their ingredients

**Scenario:** Searching for a cocktail by its ingredients

**GIVEN** I am on the homepage of the web app
**WHEN** I enter an ingredient of a cocktail on the search bar input box
**AND** I click the search button
**THEN** I can see the list cocktails matching with ingredient/s
**WHEN** I can click on a specific cocktail
**THEN** I am redirected to cocktail page with its information

#### Story #2:
**Feature:** Search cocktail by name

**As a:** Novice cocktail maker

**So that:** I can know how to make a specific cocktail

**I want to:** search cocktail name to view method to make it

**Scenario:** Search cocktail to find method on how to make it

**GIVEN** I am on the homepage of the web app
**WHEN** I enter the name of a cocktail on the search bar input box
**AND** I click the search button
**THEN** I can see the cocktail listed
**WHEN** I can click on the cocktail
**THEN** I am presented with the method on how to make it

# ***Cognac Bandits - SENG2021 Deliverable 2***

## Story #3:

**Feature:** Viewing Discussion Forum for Cocktails

**As a:** novice cocktail maker

**So that:** I can read suggestions on specific cocktails

**I want to:** browse a forum of comments posted by users for the cocktail I am looking at

**Scenario:** Viewing a specific cocktail comments in the forum

**GIVEN** I am on the homepage of the web app
**WHEN** I enter the name of a cocktail on the search bar input box
**AND** I click the search button
**THEN** I can see the cocktail listed
**WHEN** I can click on the cocktail
**AND** I scroll to the bottom of the cocktail page
**THEN** I can read comments and suggestions made by other users on the cocktail

## Story #4:

**Feature:** Creating an account

**As a:** regular cocktail maker

**So that:** I can post comments on cocktails

**I want to:** engage in conversations / discussions on cocktail pages to expand my knowledge on cocktails

**Scenario:** Creating a user account

**GIVEN** I am on the homepage of the web app
**WHEN** I click on the ‘Sign In’ button on the top right of the navigation bar
**THEN** I am redirected to the sign in page
**WHEN** I click the ‘Register’ button and enter my email and a password into the input boxes
**THEN** I am logged into the website with my account

# ***Cognac Bandits - SENG2021 Deliverable 2***

## Story #5:
**Feature:** Logging into an account

**As a:** frequent user of the web application

**So that:** I can access my account to post comments

**I want to:** log into my account.

**Scenario:** Logging into a user account

**GIVEN** I am on the homepage of the web app
**WHEN** I click on the ‘Sign In’ button on the top right of the navigation bar
**THEN** I am redirected to the sign in page
**WHEN** I enter my email and my password into the text input boxes
**THEN** I am logged into the website with my account

## Story #6:
**Feature:** Logging out of an account

**As a:** frequent user of the web application

**So that:** I ensure security that no one comments under my name

**I want to:** log out of my account.

**Scenario:** Logging out of a user account

**GIVEN** I am on the homepage of the web app
**WHEN** I click on the ‘Sign Out’ button on the top right of the navigation bar
**THEN** I am signed out of my account

# ***Cognac Bandits - SENG2021 Deliverable 2***

## Story #7:

**Feature:** Commenting on Forum for Cocktails

**As an:** expert / experienced cocktail maker

**So that:** I can post suggestions and comments on specific cocktails

**I want to:** browse a cocktails and post comments for advice / recommendations

**Scenario:** Commenting on a specific cocktail

**GIVEN** I am on the homepage of the web app
**WHEN** I enter the name of a cocktail on the search bar input box
**AND** I click the search button
**THEN** I can see the cocktail listed
**WHEN** I can click on the cocktail
**AND** I scroll to the bottom of the cocktail page
**THEN** I can view the comment text box
**WHEN** I enter 'Name' and 'Message'
**AND** I press the post button
**THEN** I post a comment on the cocktail page

## Story #8:

**Feature:** Fresh Shopping List

**As a:** novice cocktail maker

**So that:** I can make a cocktail from scratch

**I want to:** form a shopping list of ingredients I need to make a cocktail

**Scenario:** Creating a fresh shopping list of a particular cocktail

**GIVEN** I am on the homepage of the web app
**WHEN** I search on the input box or choose a particular cocktail from navigation on the top of the page and make selection
**AND** I click the add to shopping list button
**THEN** The ingredients are added to the shopping list
**WHEN** I click the 'Shopping List' tab on the navigation bar on the top of the page
**THEN** I can view the shopping list with the added ingredients

# ***Cognac Bandits - SENG2021 Deliverable 2***

## Story #9:
**Feature:** Completing Shopping List

**As a:** cocktail maker

**So that:** I can make a cocktail with ingredients I have

**I want to:** form a shopping list of ingredients that are left to complete ingredient list of a cocktail

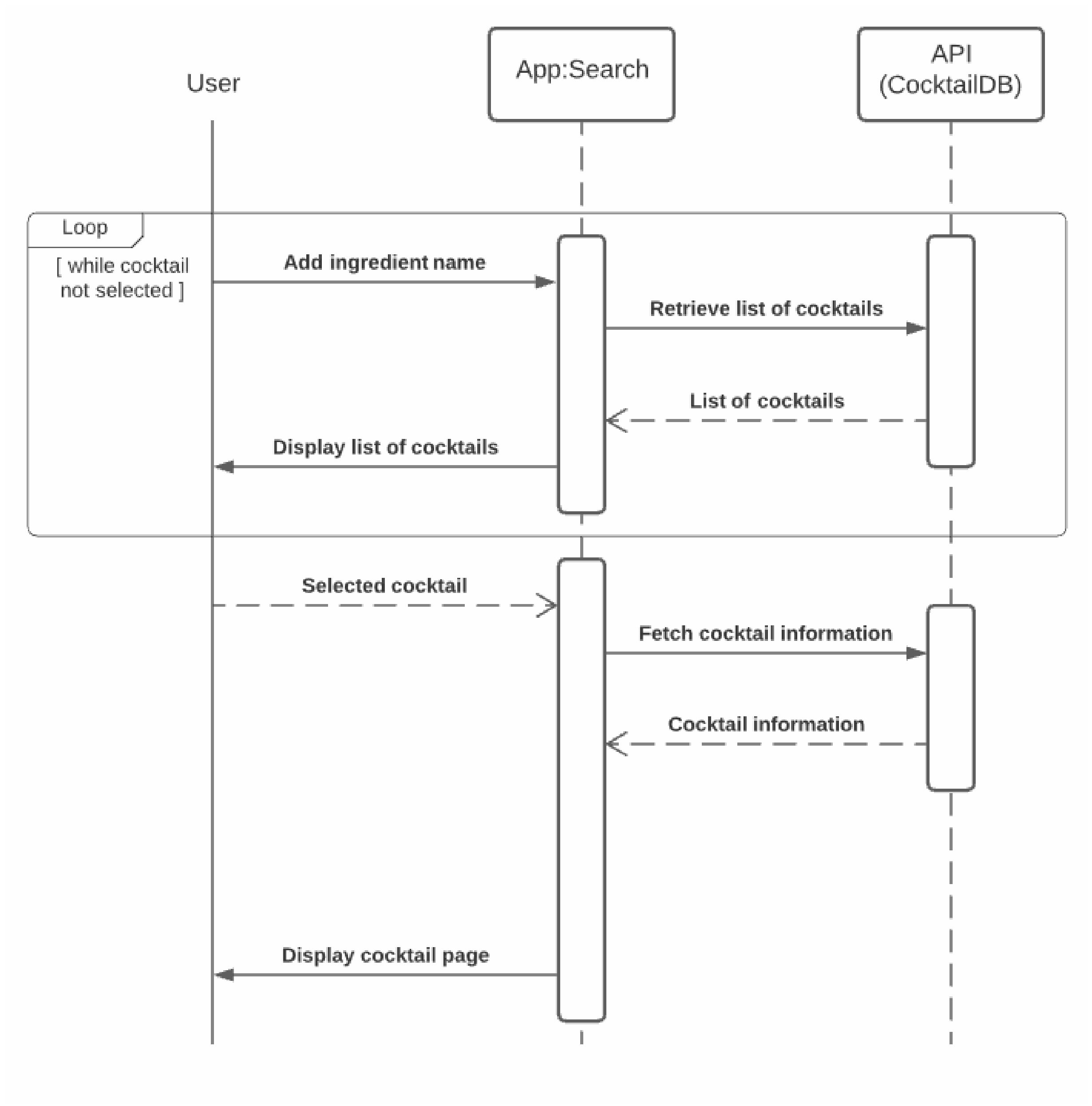
**Scenario:** Creating a completing shopping list of a particular cocktail

**GIVEN** I am on the homepage of the web app
**WHEN** I search on the input box or choose a particular cocktail from navigation on the top of the page and make selection
**AND** I click the ingredients that I am missing
**THEN** The ingredients are added to the shopping list
**WHEN** I click the 'Shopping List' tab on the navigation bar on the top of the page
**THEN** I can view the shopping list with the added ingredients

## Sequence Diagrams

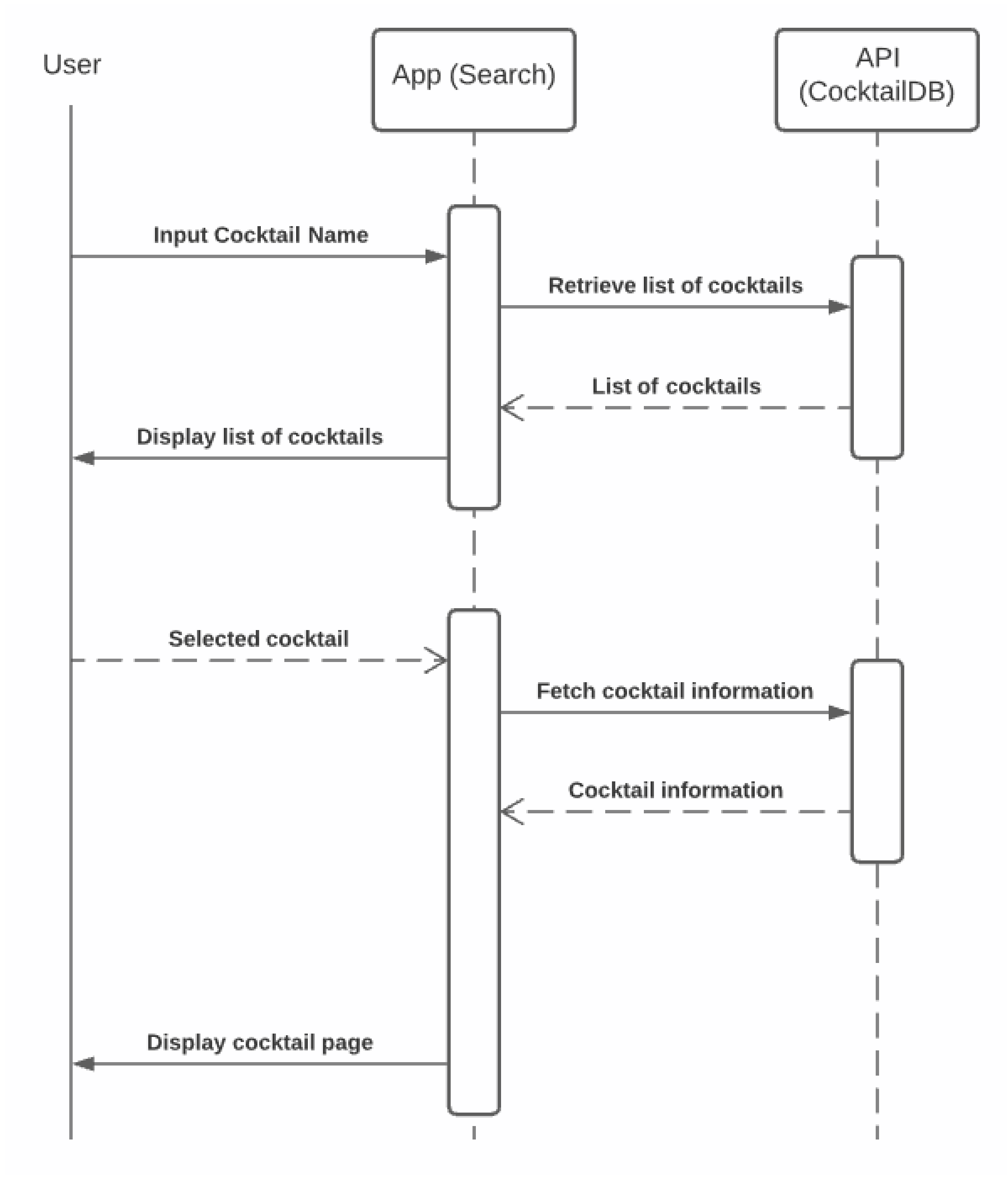
User Story 1: Cocktail Search

*Figure 6 - Sequence Diagram 1*



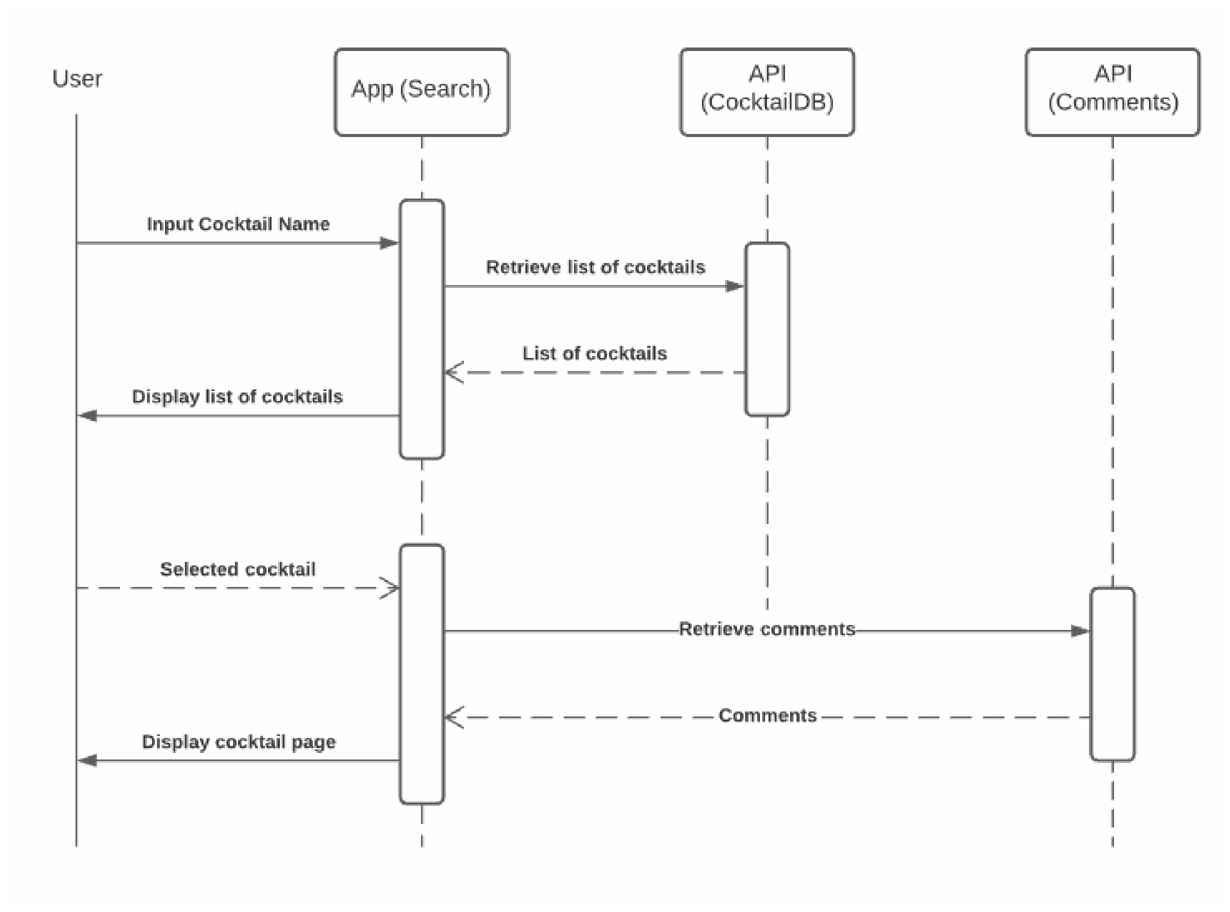
User Story 2: Ingredient Search

*Figure 7 - Sequence Diagram 2*



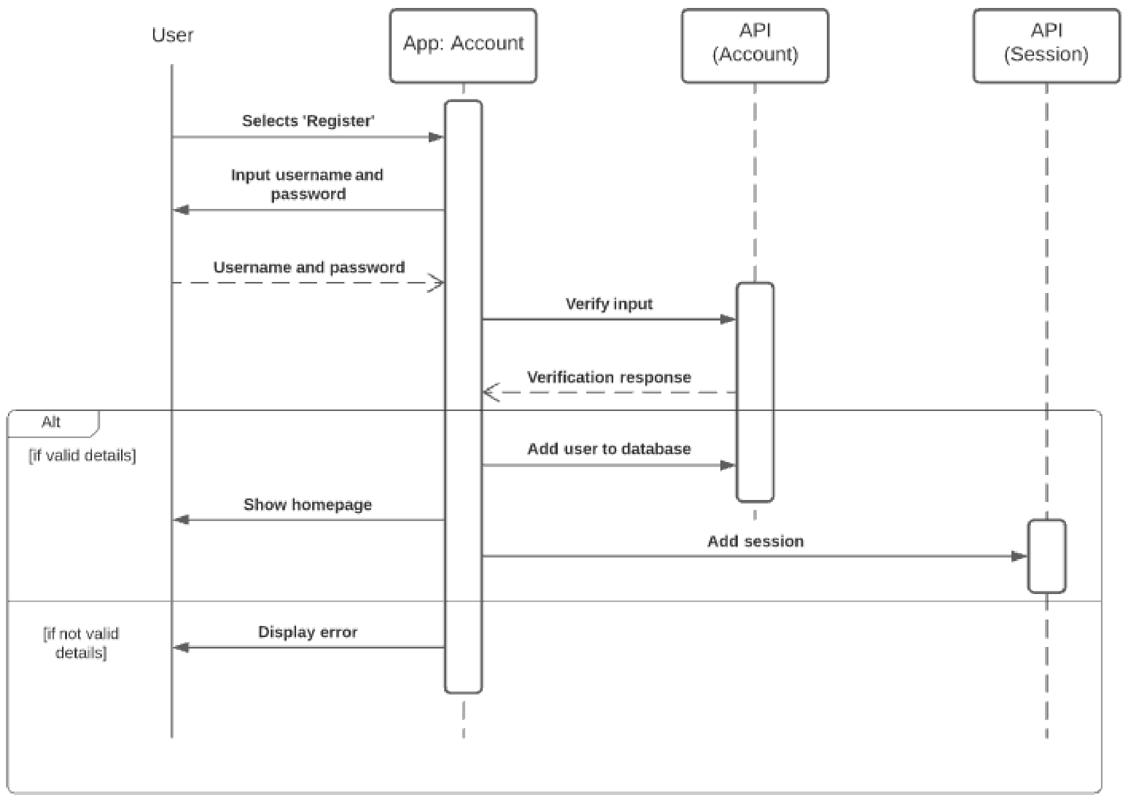
User Story 3: Viewing comments

*Figure 8 - Sequence Diagram 3*



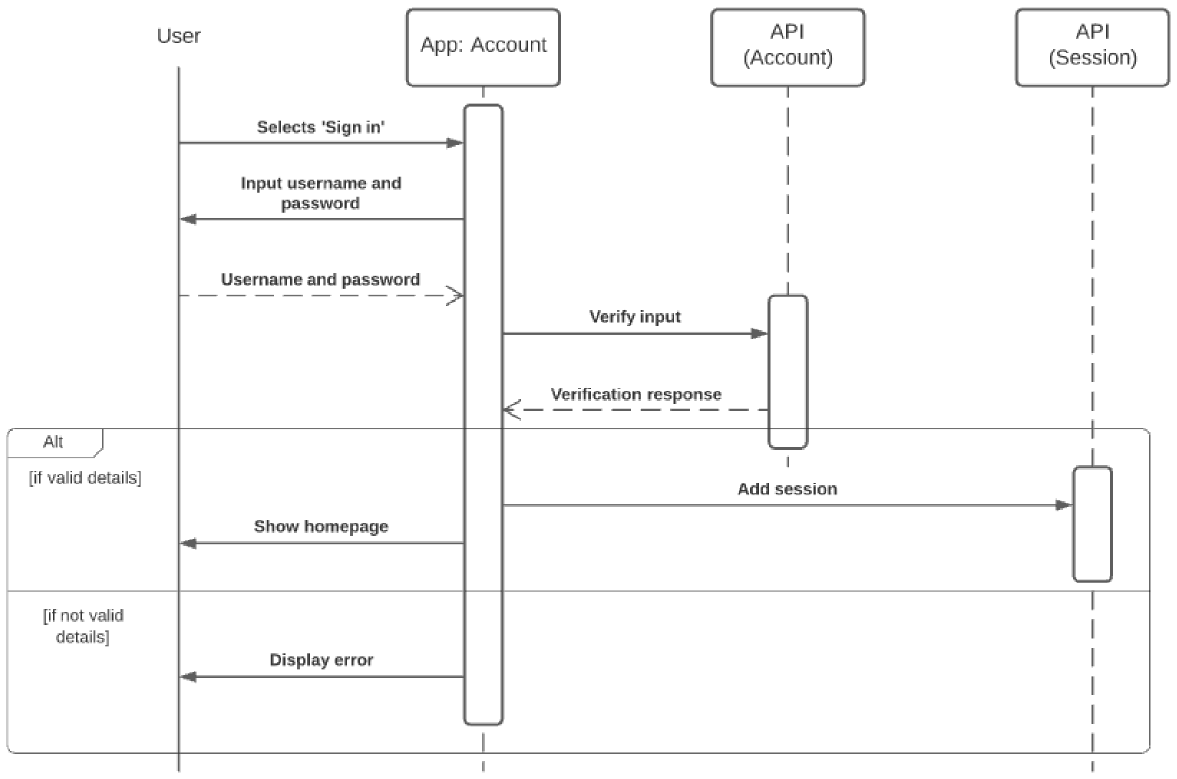
User Story 4: Registering Account

*Figure 9 - Sequence Diagram 4*



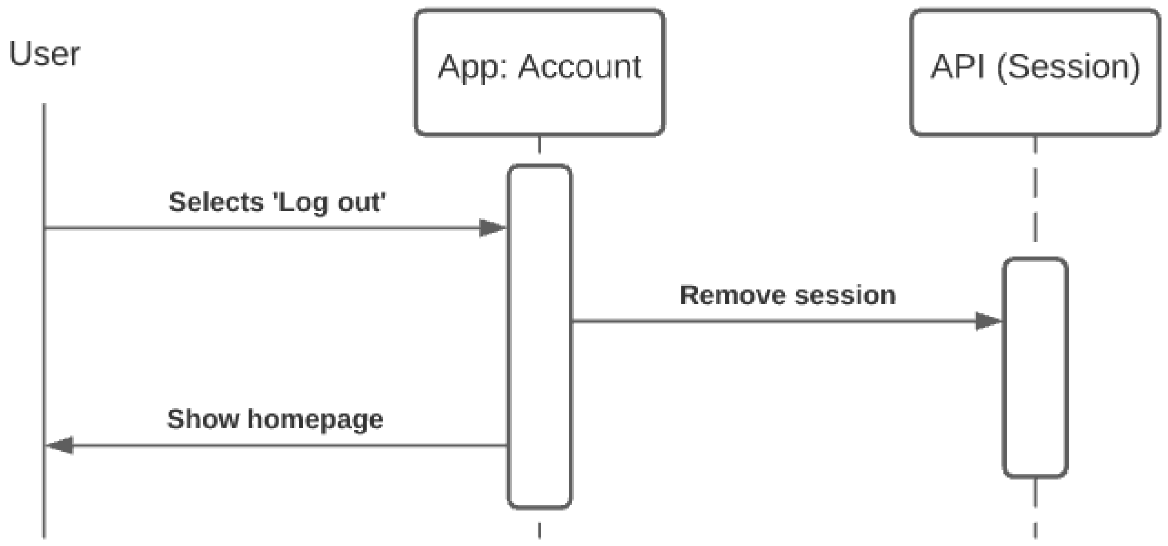
User Story 5: Signing on

*Figure 10 - Sequence Diagram 5*



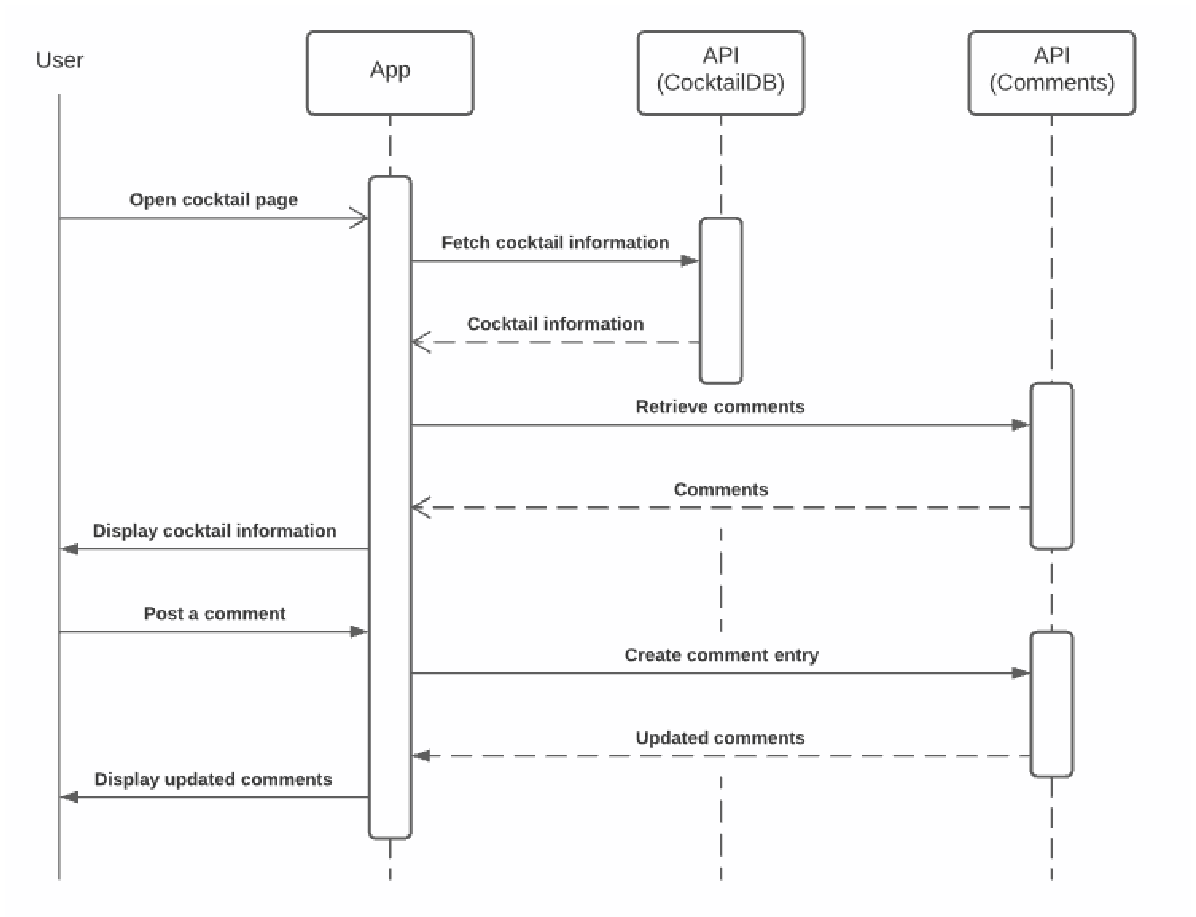
User Story 6: Logging out

*Figure 11 - Sequence Diagram 6*



User Story 7: Posting a comment

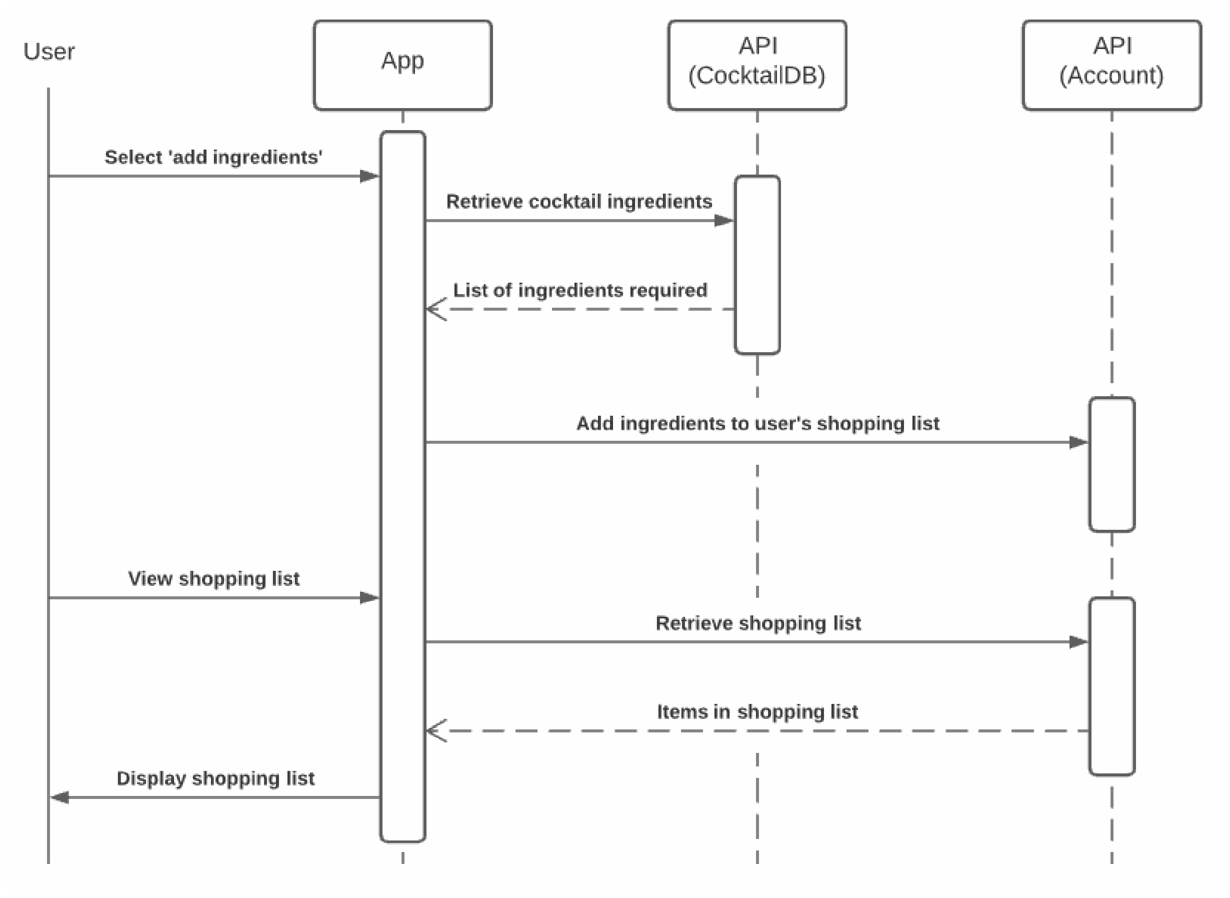
Figure 10: Sequence Diagram 7



# Cognac Bandits - SENG2021 Deliverable 2

User Story 8: Adding ingredients to shopping list

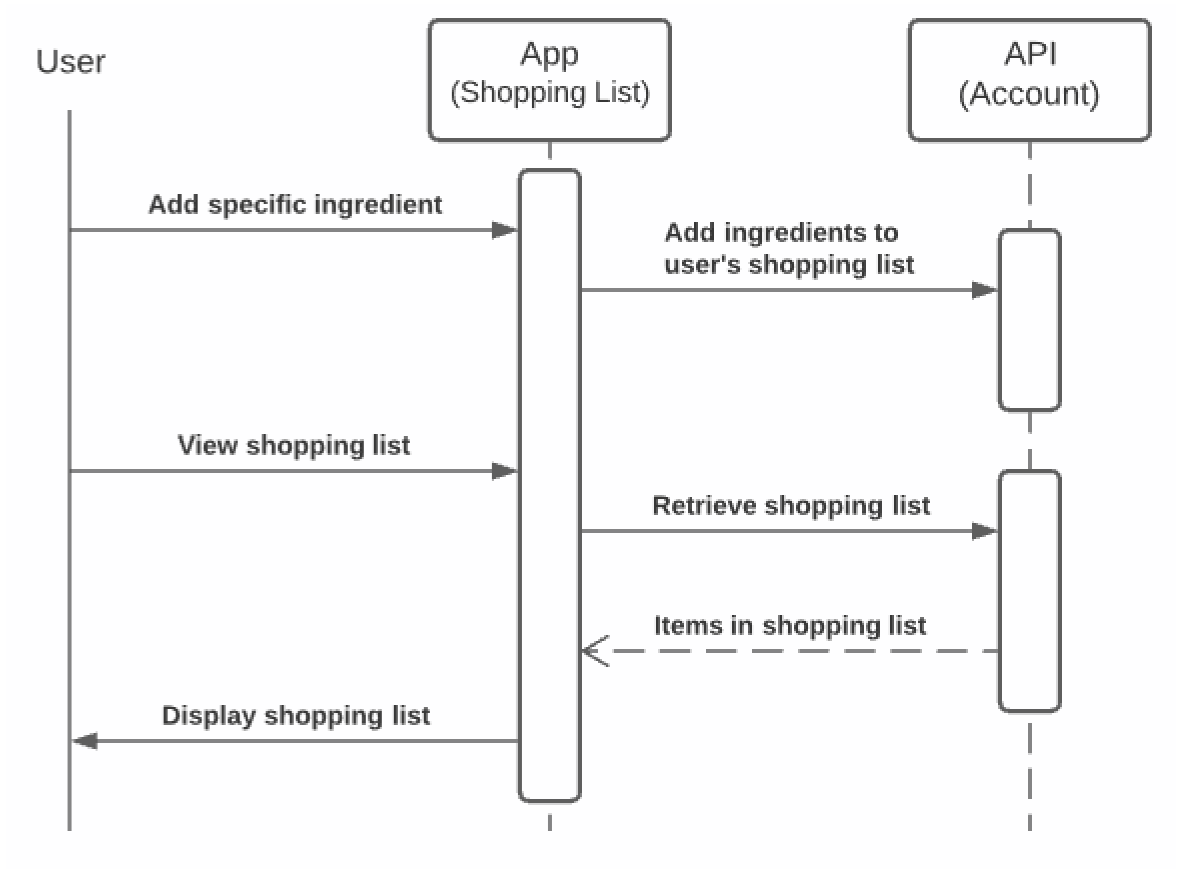
Figure #: Sequence Diagram 8



# ***Cognac Bandits - SENG2021 Deliverable 2***

User Story 9: Filtering out owned ingredients

Figure #: Sequence Diagram 9


User
App
(Shopping List)
API
(Account)
Add specific ingredient
Add ingredients to user's shopping list
View shopping list
Retrieve shopping list
Items in shopping list
Display shopping list