System/Subsystem Design Description

By: Dominic Crofoot, Travis Early, Jordan Fok, Sean Hill, & Nicholas Ho

Created November 10, 2018

Revisions:

- **20Nov2018: First draft**
    - First draft of SSDD and diagrams created.

Table of Contents:

1. Scope

1.1. Identification

1.1.1. SSDD: System/Subsystem Design Description

1.1.2. SRS: System Requirements Specification

1.1.3. Date planner: unnamed application team is designing

1.1.4. FR1,2,3: Functional requirements 1, 2, & 3. Further specified in SRS

1.1.5. EIR1,2,3: External Interface Requirements 1, & 2. Further specified in SRS

1.1.6. Table1: transition state diagram found in SRS

1.1.7. Table2: Qualification procedures for each requirement found in SRS

1.1.8. API: Application Program Interface

1.1.9. MIL-STD: Military Standard

1.2. System Overview

1.2.1. The system will provide users the ability to search through date ideas based on parameters (filters) they have chosen.

1.3. Document Overview

1.3.1. The SSDD will describe the design choices made by the team. Primarily, how the system wide design will look and how sub-parts will be designed as well (ie. login page, map, results tab, filters, and help page). The program will be able to save filters by users who choose to log in, and will ask for a username and password, but no other information will be saved or required of user.

2. Referenced Documents

2.1. Government-

2.1.1. MIL-STD 498, 5 December 1994

2.1.1.1. Template for entire SRS document

2.1.1.2. Link: MIL-STD 498

2.2. Non-Government-

2.2.1. Loopback

2.2.1.1. A node.js framework used for back-end services. Loopback will be used by the application for user authentication and retrieving of the users filter history.

2.2.1.2. Link: https://loopback.io/

2.2.2. Google Maps API

2.2.2.1. Data supplied by google that will be used for filter, results tab, and map.

2.2.2.2. Link: https://cloud.google.com/maps-platform/

3. System-Wide Design Decisions

To satisfy the requirement that users be provided a variety of date suggestions when supplying proper filters into the system, a map will be populated showing pins for each of the suggested date ideas within a specified area. Before the map will be populated the user must supply information into a set of filters. These filters are specific requirements about the date that the user does or doesn't want the date to meet. Which pins appear on the map for a user is a direct result of the information that was supplied by the user into the filters. A part of the page will also be dedicated to clickable objects that contain more information on each date suggestion.

To satisfy the requirement of stored and recallable search history, there will be account creation and login pages that allow the user to create and log into an account specific to that user. Search history for a logged on user will be automatically saved by the system. The user can then re-run searches that they have done in the past.

4. System Architectural Design

4.1. System Components

4.1.1. User SQL Database (CSCI) - Identifier C4.1.1

4.1.1.1. Purpose: Stores login and account information for users in a table. Stores the search history of a user in a table. Retrieves stored information for other components when needed and allows them to carry out their functions.

4.1.1.2. Requirement Association: Satisfies Functional Requirement 3 and and External Interface requirement 2 established in the SRS.

4.1.1.3. Relationships: Uses the Filters component to develop tuples in the database table that stores a user's search history. The tuples are constructed using data input by users into the Filters component.Uses the Login Page component to search the database table containing user account information to determine when an existing user logs in. (Refer to Figure 1 below)

4.1.1.4. Development Status: New Development

4.1.1.5. Hardware Resources: N/A

4.1.2. Login Page (CSCI) - Identifier C4.1.2

4.1.2.1. Purpose: Allows users to create new accounts or sign into existing accounts which allows them to store their search history and retrieve them at a later date.

4.1.2.2. **Requirement Association:** Satisfies External Interface Requirement 2 established in the SRS.

4.1.2.3. **Relationships:** Consists of the Loopback API (CSC) which provides functionality to the login page in order to allow users to create accounts or log in to existing accounts. Uses the User SQL Database component to determine existing users. (Refer to Figure 1 below)

4.1.2.4. **Development Status:** New Development

4.1.2.5. **Hardware Resources:** N/A

4.1.3. Map (CSCI) - Identifier C4.1.3

4.1.3.1. **Purpose:** Provides users with a visualization of the date location results based on the data in the Filters component. Allows users to obtain more information about each result by clicking on the pin associated with its geographical location on the map.

4.1.3.2. **Requirement Association:** Satisfies Functional Requirements 1 and 2.

4.1.3.3. **Relationships:** Consists of the Google Maps API (CSC) which populates the contents of the map, filling it with queried information. Uses the Filters Component which passes input data to the Google Maps API to determine the specifics of the query. (Refer to Figure 1 below)

4.1.3.4. **Development Status:** New Development.

4.1.3.5. **Hardware Resources:** N/A

4.1.4. Filters (CSCI) - Identifier C4.1.4

4.1.4.1. **Purpose:** Receives input from the user which is used to query the Google Maps API for a list of possible date location results that align with the given inputs. Allows users to tailor their results to match their specific requirements for date locations.

4.1.4.2. **Requirement Association:** Satisfies Functional Requirements 1 and External Interface Requirement 1.

4.1.4.3. **Relationships:** Consists of the Google Maps API (CSC) which is queried using data obtained as input from the filters. Uses the User SQL Database component to populate filters based on a previous search stored in the database. (Refer to Figure 1 below)

4.1.4.4. **Development Status:** New Development

4.1.4.5. **Hardware Resources:** N/A

4.1.5. Results List (CSCI) - Identifier C4.1.5

4.1.5.1. Purpose: Displays the list of possible date location results provided as output from the Map component. Provides useful information about these locations such as open and closing times as well as phone numbers if they exist. Allows users the option to obtain additional information about a location by clicking on a result.

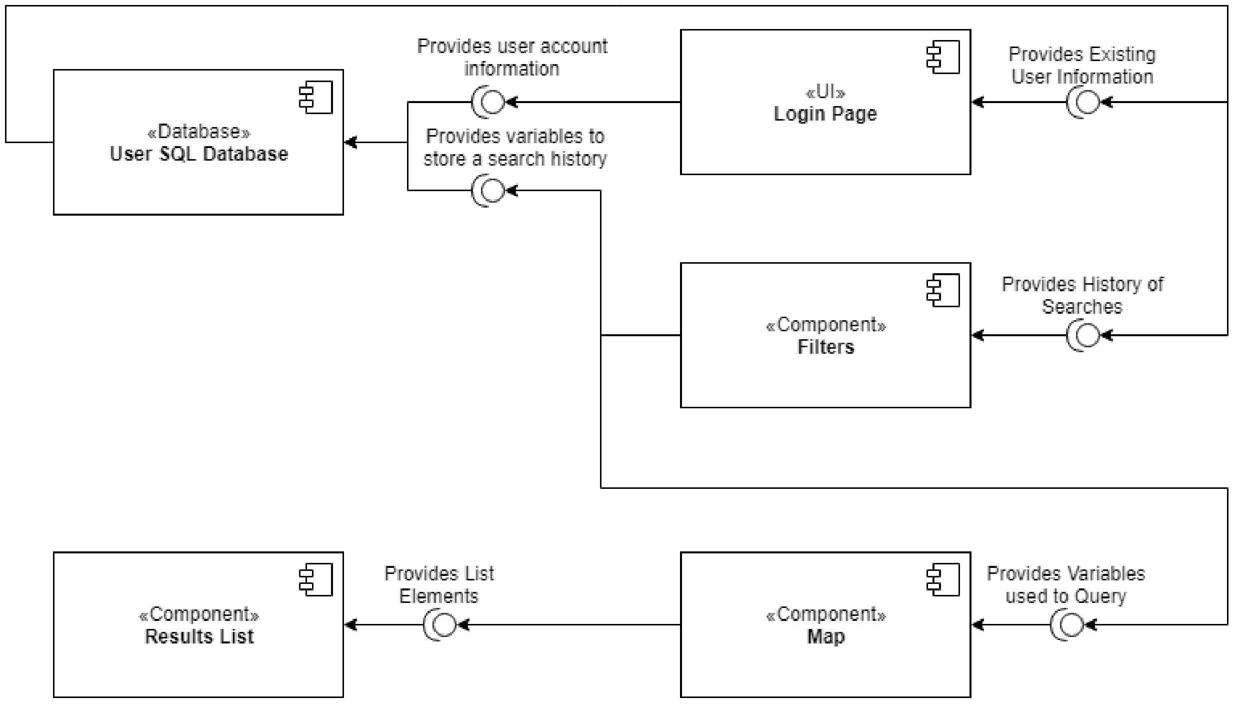
4.1.5.2. Requirement Association: Satisfies Functional Requirement 1

4.1.5.3. Relationships: Uses data stored in the Map component to populate values into the results table. (Refer to Figure 1 below)

4.1.5.4. Development Status: New Development

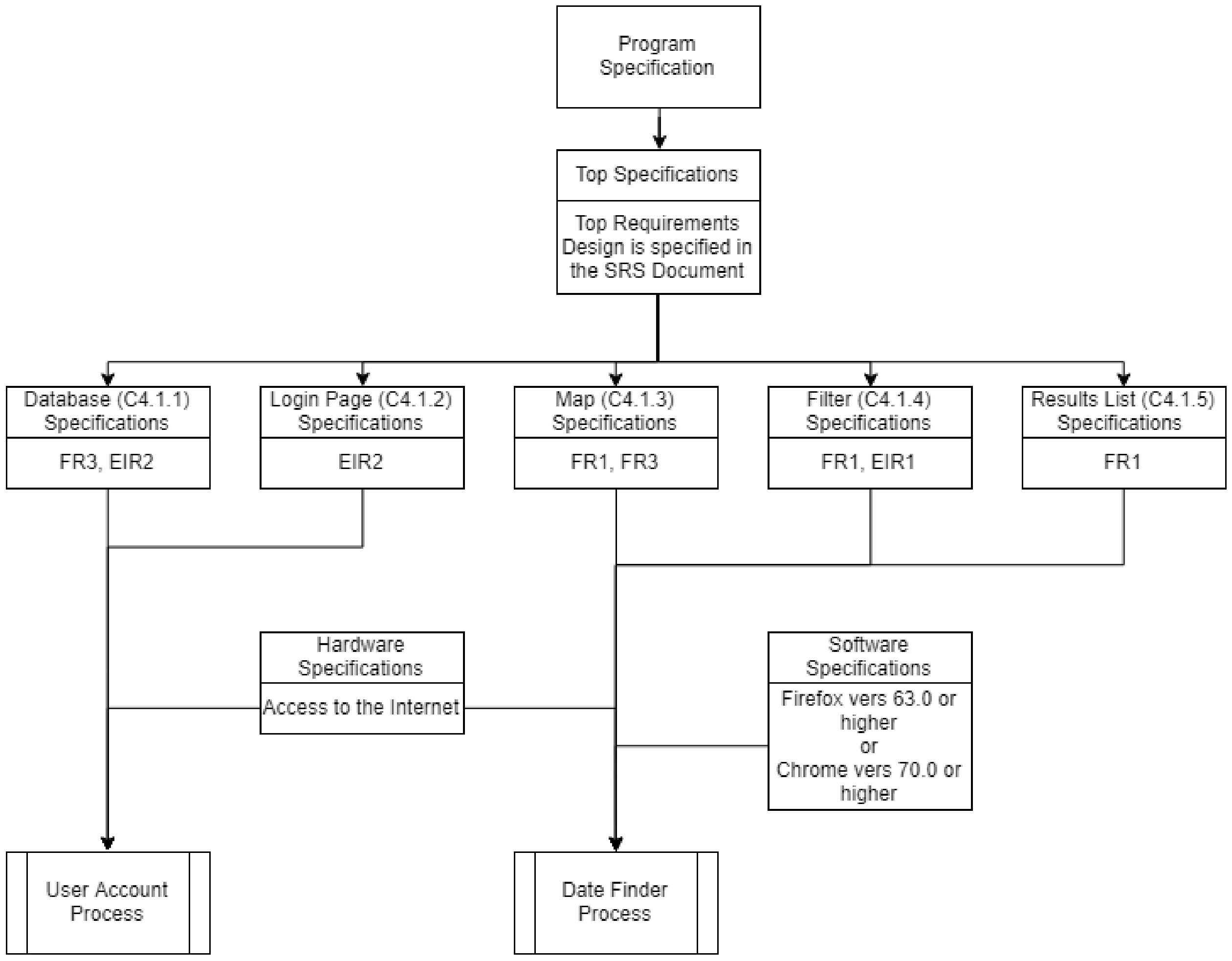
4.1.5.5. Hardware Resources: N/A

UML Component Diagram (Figure 1) : Below is the UML Component diagram that shows the relationships between components, specifically the required functionalities provided by other components.



«Database»
User SQL Database
Provides user account information
Provides variables to store a search history
«UI»
Login Page
Provides Existing User Information
«Component»
Filters
Provides History of Searches
«Component»
Results List
Provides List Elements
«Component»
Map
Provides Variables used to Query

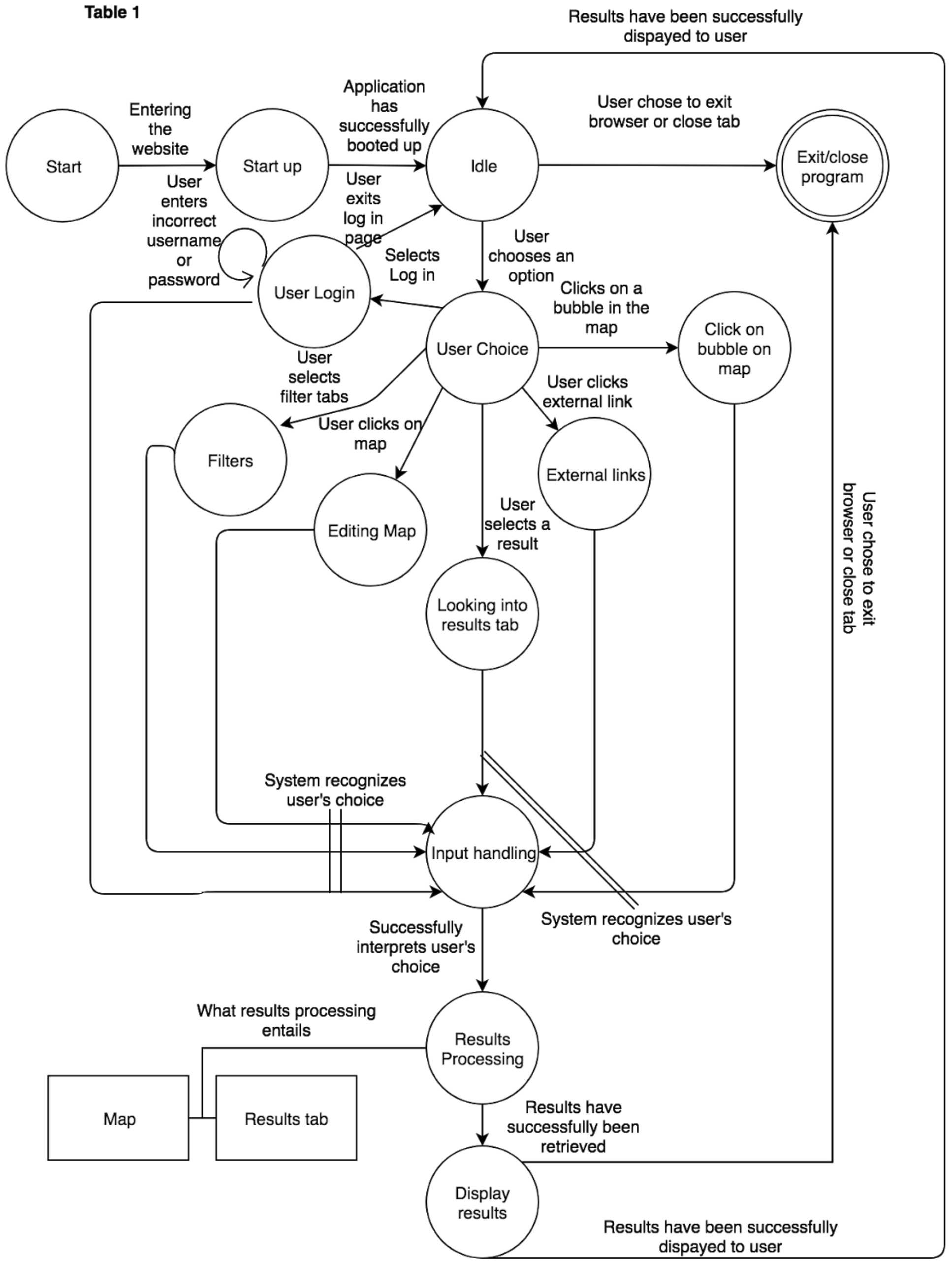
Specification Tree (Figure 2): Below is the specifications tree that identifies the relationships among the planned specifications and requirements for the system components.


Program Specification
Top Specifications
Top Requirements Design is specified in the SRS Document
Database (C4.1.1) Specifications
FR3, EIR2
Login Page (C4.1.2) Specifications
EIR2
Map (C4.1.3) Specifications
FR1, FR3
Filter (C4.1.4) Specifications
FR1, EIR1
Results List (C4.1.5) Specifications
FR1
Hardware Specifications
Access to the Internet
Software Specifications
Firefox vers 63.0 or higher or Chrome vers 70.0 or higher
User Account Process
Date Finder Process

4.2. Concept of Execution

4.2.1. As the user opening the application through the clients web browser, the user can choose to login with a username and password. Upon logging in the client will receive a token. This token can be used to then retrieve a list of the users filter history which can then be used when searching for date locations. When a search request is made, the request is sent to the google maps API to find locations relative to the user’s search. The list will then be filtered through by applying the filters selected by the user.
The table shown below is the state machine from the SRS. Table1 best shows the overall design flow of the program. Given this overview of the flow of the program and how it is designed, the following diagrams will dive into sub-components and how each interacts with the program.

Table 1


Results have been successfully dispayed to user
Start
Entering the website
Start up
Application has successfully booted up
Idle
User chose to exit browser or close tab
Exit/close program
User enters incorrect username or password
User exits log in page
User Login
Selects Log in
User chooses an option
Clicks on a bubble in the map
User Choice
Click on bubble on map
User selects filter tabs
User clicks external link
Filters
User clicks on map
External links
Editing Map
User selects a result
Looking into results tab
User chose to exit browser or close tab
System recognizes user's choice
Input handling
System recognizes user's choice
Successfully interprets user's choice
What results processing entails
Results Processing
Map
Results tab
Results have successfully been retrieved
Display results
Results have been successfully dispayed to user

4.3. Interface Design

4.3.1. Interface Identification and Diagrams

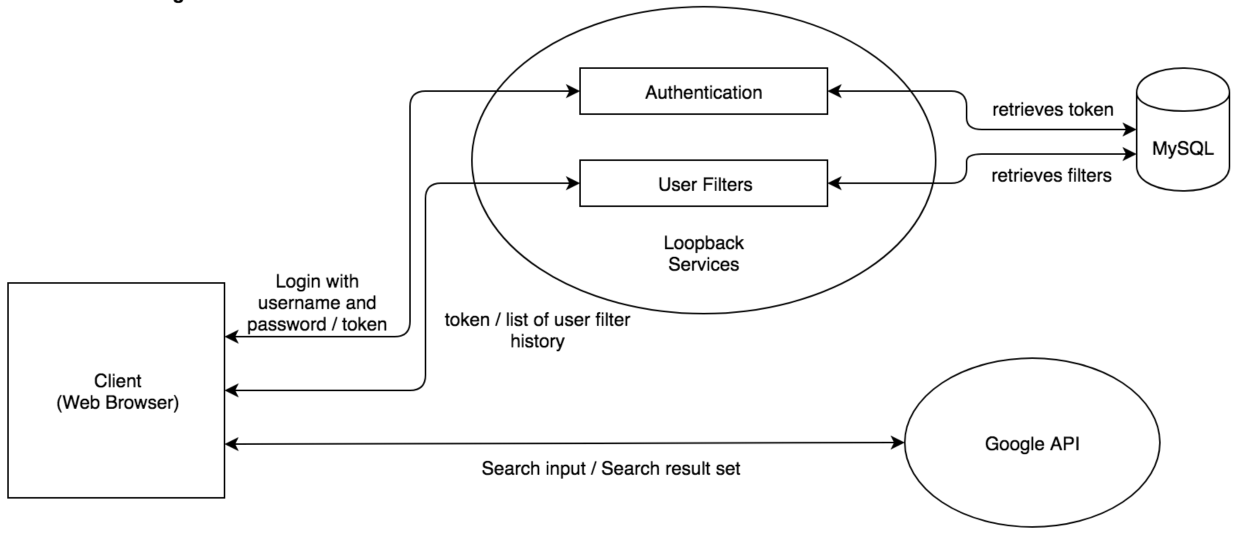
4.3.1.1. The diagrams are organized as such:

4.3.2: External entities interface diagram (ID1)
4.3.3: Loopback Interface Diagram (ID2)
4.3.4: MySQL Interface Diagram (ID3)
4.3.5: Client (Web Browser) Interface Diagram (ID4)
4.3.6: Google Maps API interface (ID5)

4.3.2. External Systems Interface Diagram (ID1)

For the system to operate properly, the use of MySQL and Google Maps API is required. This diagram (ID1) gives a broad overview of how the client pulls from MySQL and Google Maps API.

Figure 3

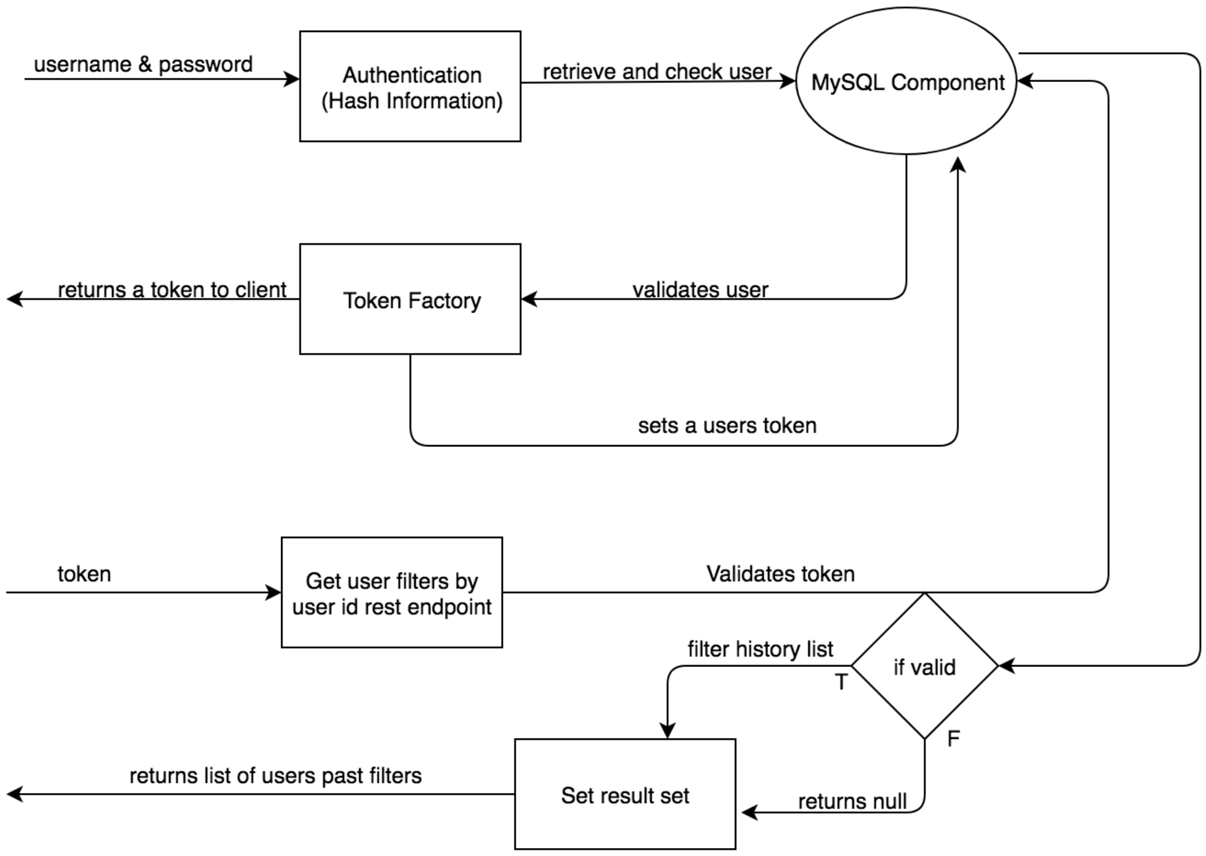


The Loopback Services are included in this diagram because the filters from the browser are used and saved. Further details of how this works will be in the next diagram.

4.3.3. Loopback Interface Diagram (ID2)

Loopback is a service being used to save users and their information (more detailed information can be found in the SRS). The process is started the user submitting a username and password (logging in or signing up). Then that information is authenticated, and passed on to MySQL. From MySQL's components, it validates the user. Then the token factory will return a token to the client and sets a user token, which is sent back to MySQL. The token is sent back in order to get filters. Once the token is validated, the system returns to MySQL. If the token is valid it returns a filter history, if the user has one.

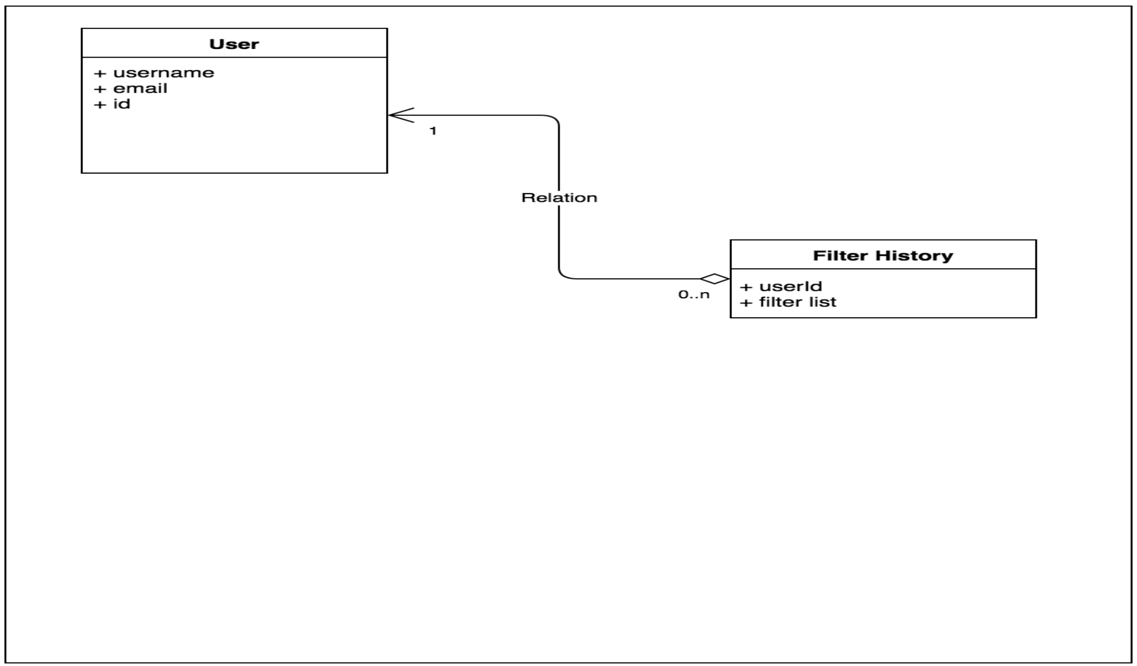
Figure 4



Loopback is a seperate system that is being utilized to store user information and filter history.

4.3.4. MySQL Interface Diagram (ID3)

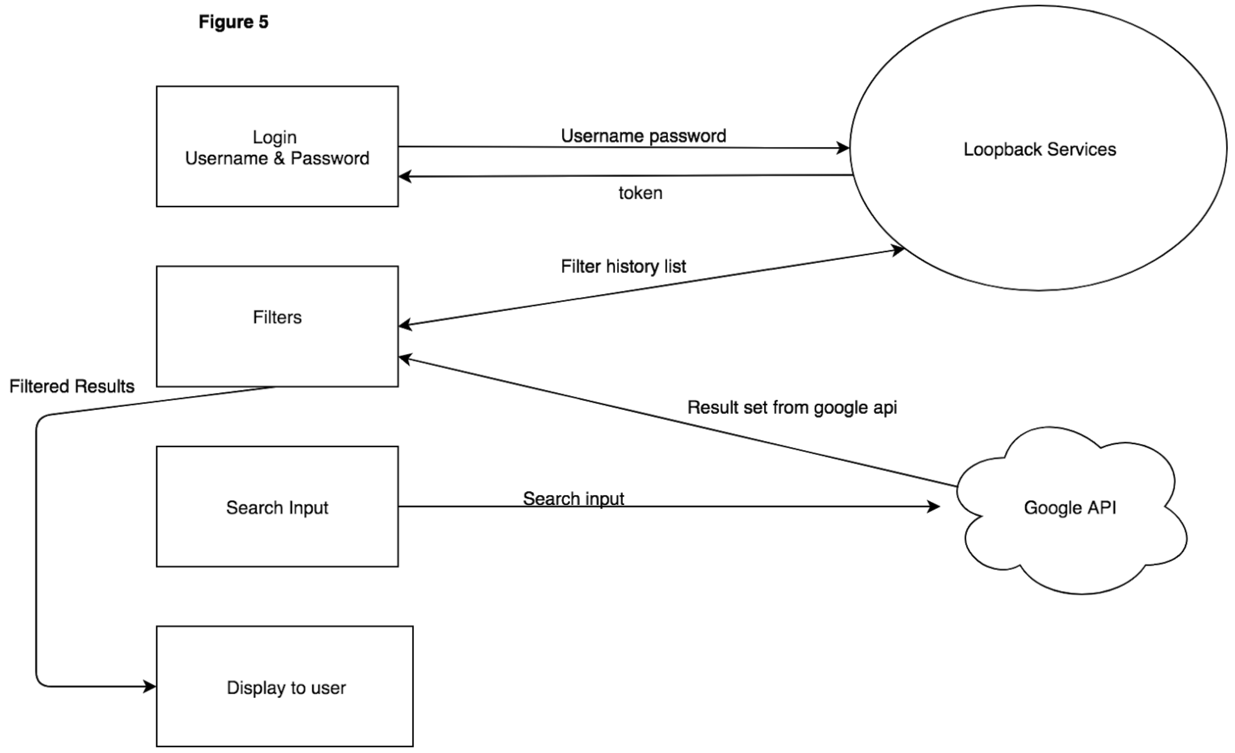
This represents the relationship between our data we will provide to the users. The data stored by the MySQL system will be relative slim and small, this is by design. Most of our data will take use of the external apis to reduce the amount of data stored by the MySQL system. The system will store the users and their filter history which has a relation of one to many.



4.3.5. Client (Web Browser) Interface Diagram (ID4)

The system has been proven to be operational on the Web Browsers Google Chrome version 70.0.3538.110 / November 19, 2018, and Mozilla FireFox 63.0.3 / November 15, 2018. This diagram shows how the application will interact with inside the browser.

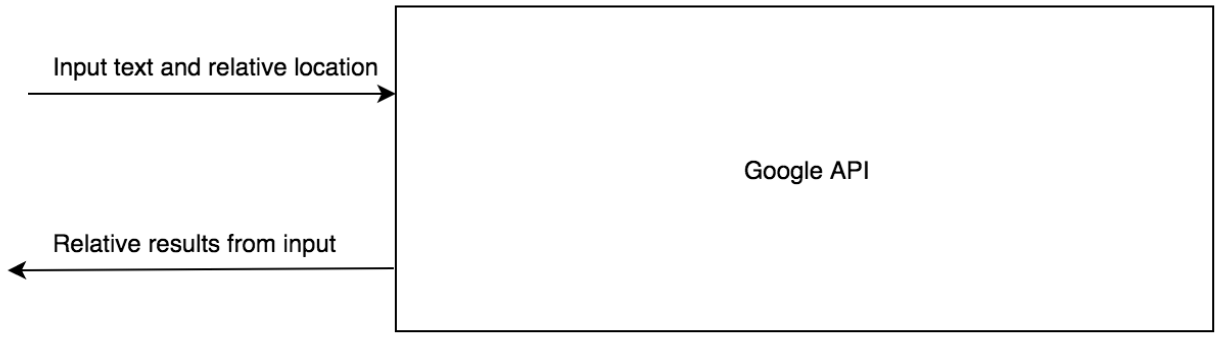
Figure 5



4.3.6. Google API Interface Diagram (ID5)

The inner workings of Google Maps API is unknown to the team, so the diagram addresses what the application sends to the API and what the system receives from it.

Figure 6



5. Requirements Traceability

The following table describes the traceability between the Functional Requirements as described in the SRS and the CSCIs as described in this document.

Requirement	Capability	Related CSCIs
FR1	Provide search results based on user filters	**C4.1.3, C4.1.4**
FR2	Provide a user-friendly location display	**C4.1.3**
FR3	Allow registered users to store searches	**C4.1.1**
EIR1	Allow the user to customize search options	**C4.1.4**
EIR2	Store and authenticate user-specific accounts	**C4.1.1, C4.1.2**

6. Notes

6.1. Diagrams are subject to change as new features are added or new information about external services are found out.