Sprint One:

Requirements Engineering:

Part A) User and System Requirements:

User Requirements:

The YelpHelp system will recommend a bar that is safe (where the number of occupants is not too high) and cool (where the number of occupants is not too low).

System Requirements:

1. Given some location L, the system will compile and store a collection of bars within some radius R of location L.
2. The system will read relevant data and assign "coolness" and "safety" factors to each bar in the predefined collection.
3. A list of bars will be sorted by "coolness" factors in descending order.
4. The system will provide a bar recommendation given some "coolness" and "safety" constraints.
5. If no constraints are provided, the system will automatically recommend the bar where the safety and coolness are as optimal as possible (following some internal algorithm).

Part B) Functional and Non-Functional Requirements:

Functional Requirements:

The bar will follow reasonable restrictions with respect to occupancy or density and implement certain safety measures such as social distancing, and these will apply to all customers and staff.

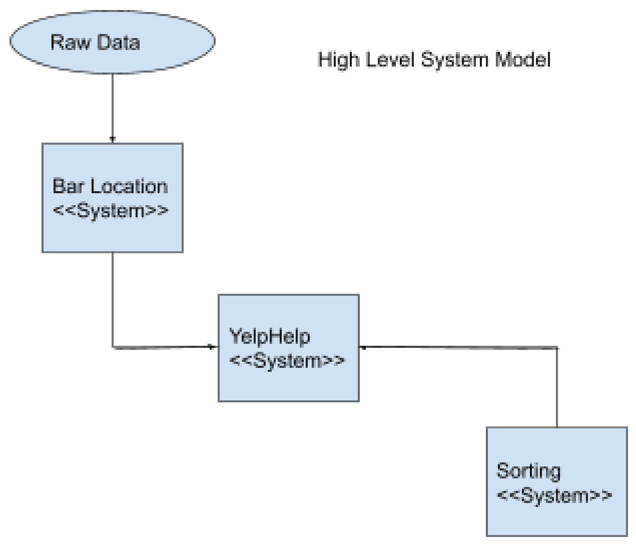
1. A user shall be able to search for a bar given some custom inputs (specific coolness and safety factors).
2. The system will generate a bar recommendation that fits the user's inputs.
3. Each set of unique inputs will correspond to a unique bar recommendation (i.e changing the inputs will change the output).

Non-Functional Requirements:

1. Product Requirements:
    a. Performance: The system shall return a bar recommendation within some specified time constraint. The system must be organized so performance is maximized, as the users will expect multiple recommendations within a reasonable timeframe
    b. Space: The system should minimize its space requirements, however, space considerations are considered secondary to performance.
    c. Dependability/reliability: System must be designed to be dependable given varying inputs. Should account for a wide range of test cases and incomplete data.
2. Organizational Requirements:
    a. Environmental: The system shall be hosted in an environment that is able to fetch and interact with real-time data.
3. External Requirements:
    a. Ethical considerations: The system shall not access personal data from the user without permission.

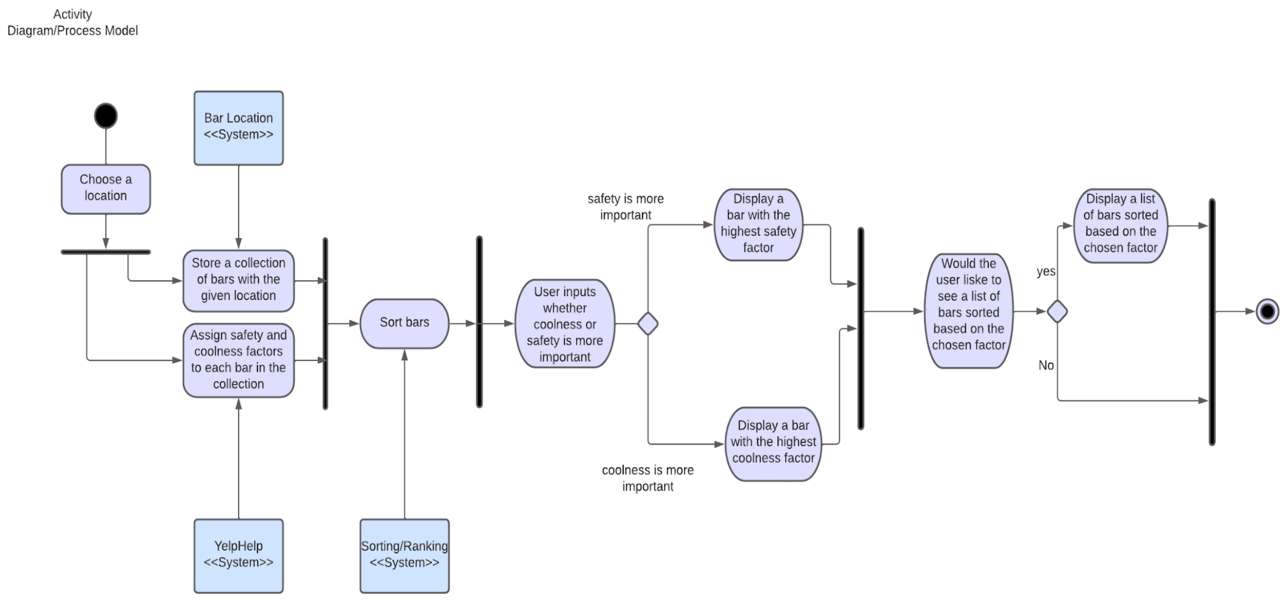
System Models:

Context Models:



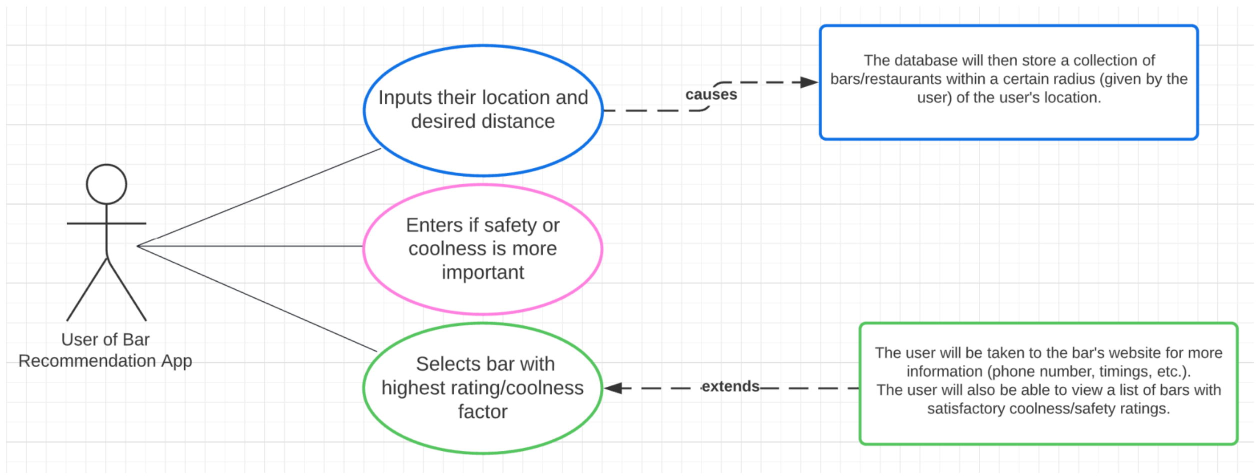
The system will be composed of multiple interacting systems external to its context. The main system which will be implemented is named YelpHelp and will host most of the functionality of the entire system. The Bar location system shall coordinate with raw data (gathered from API) to store relevant data and information, the sorting system will gather all relevant data and sort it via whatever metric is defined internally (by safety or coolness).

Activity Diagram/Process Model:


Activity Diagram/Process Model
Bar Location <<System>>
Choose a location
Store a collection of bars with the given location
Assign safety and coolness factors to each bar in the collection
Sort bars
User inputs whether coolness or safety is more important
safety is more important
Display a bar with the highest safety factor
coolness is more important
Display a bar with the highest coolness factor
Would the user liske to see a list of bars sorted based on the chosen factor
yes
Display a list of bars sorted based on the chosen factor
No
YelpHelp <<System>>
Sorting/Ranking <<System>>

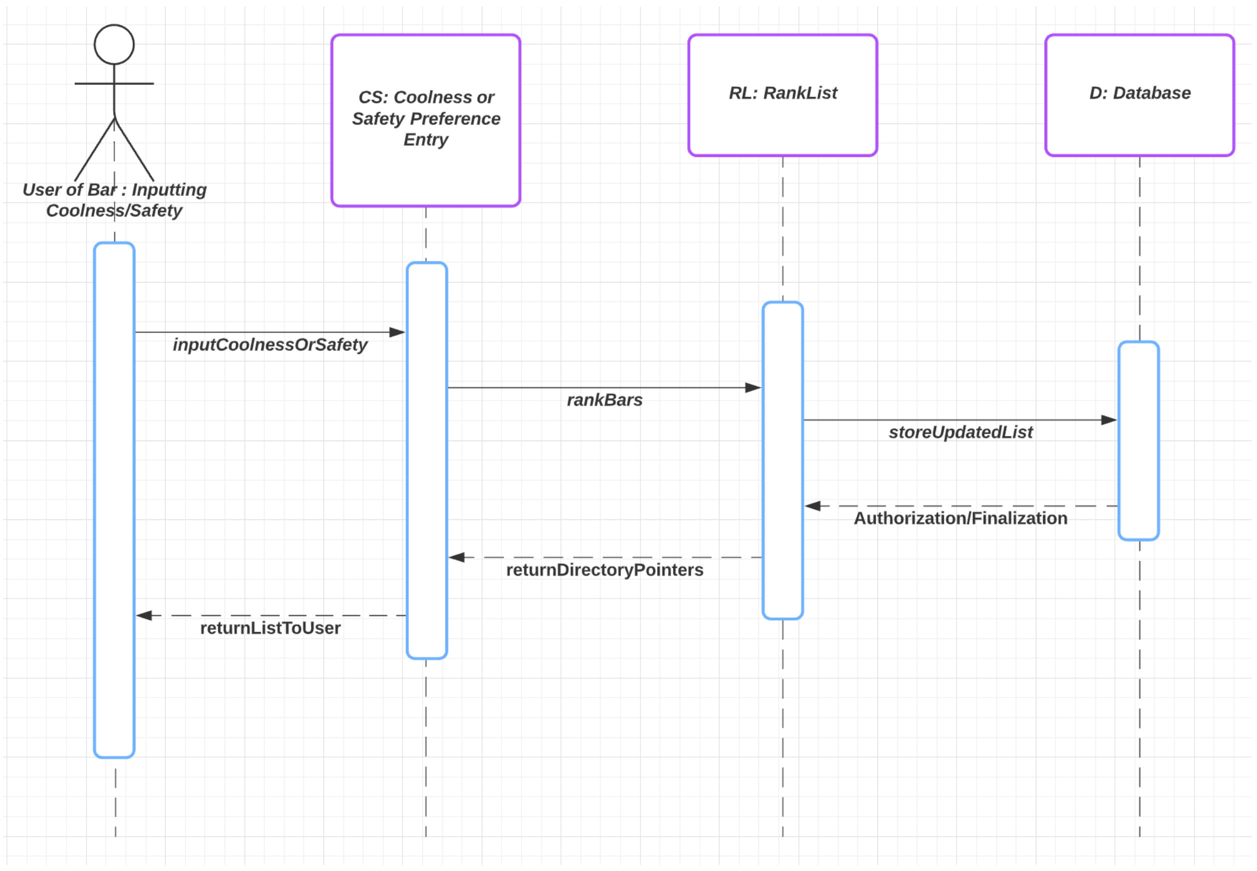
The filled circle indicates the start of the process. The bar customer provides his/her location. Then a collection of bars will be stored within a certain radius of the user's location. Each bar in the collection will be assigned safety and coolness factors. After that, bars are sorted. Then the system will ask the user whether safety or coolness is more important. If the user chooses safety, the system will output the bar with the highest safety factor. If the user chooses coolness, the system will output the bar with the highest coolness factor. Then the user will be asked if he/she wants a list of bars sorted based on the chosen factor. If the user enters yes, then a list of sorted bars will be displayed.

Use Case Diagram:


Inputs their location and desired distance
causes
The database will then store a collection of bars/restaurants within a certain radius (given by the user) of the user's location.
Enters if safety or coolness is more important
User of Bar Recommendation App
Selects bar with highest rating/coolness factor
extends
The user will be taken to the bar's website for more information (phone number, timings, etc.).
The user will also be able to view a list of bars with satisfactory coolness/safety ratings.

The use case diagram lists three specific cases of how the user will interact with the YelpHelp system. The user will first input their current location, causing the database to store a list of bars that are located within a certain desired distance from the user's location. This ensures that the bar is not very far away. Second, the user inputs if they prefer coolness or safety to be the main feature of the bar. Third, the user inputs any preferred safety factors such as mask and vaccine regulations. Lastly, a bar with the highest rating will be given; the user then will be able to access the bar's site for further information. The user will also have access to a list of bars with high coolness/safety ratings.

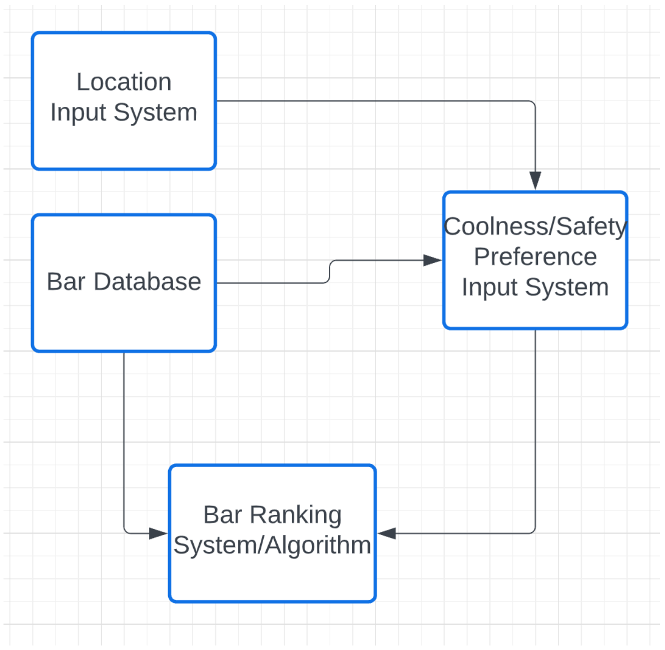
Sequence Diagram:



This sequence diagram illustrates the use case of the customer entering if they place more importance/preference on general coolness or safety. As soon as they input this, bars are ranked based on the coolness/safety rating (out of 100). This list is stored into the database and once finalized, the directory pointers are given and the final list produced to the user.

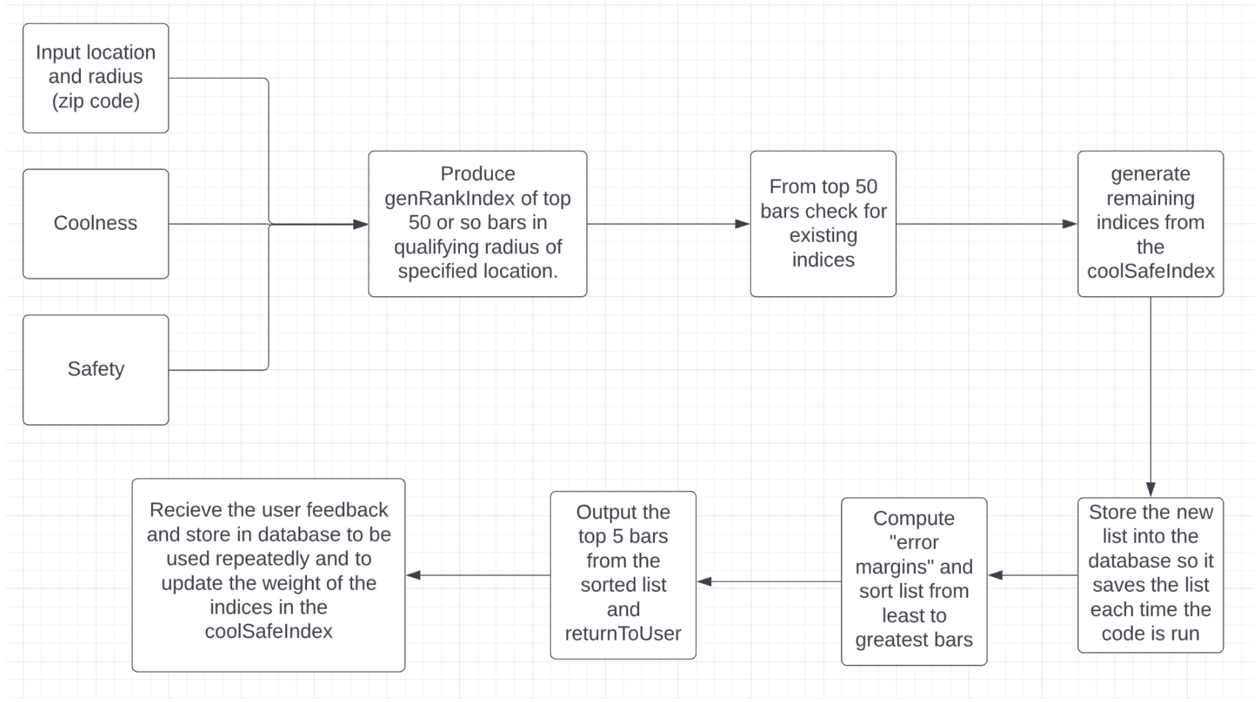
Architectural Designs:

Conceptual View:


Location Input System
Bar Database
Coolness/Safety Preference Input System
Bar Ranking System/Algorithm

The conceptual diagram shows the system units/components such as the components in which the user inputs the location and preferences and the algorithm that creates the ranked list of bars.

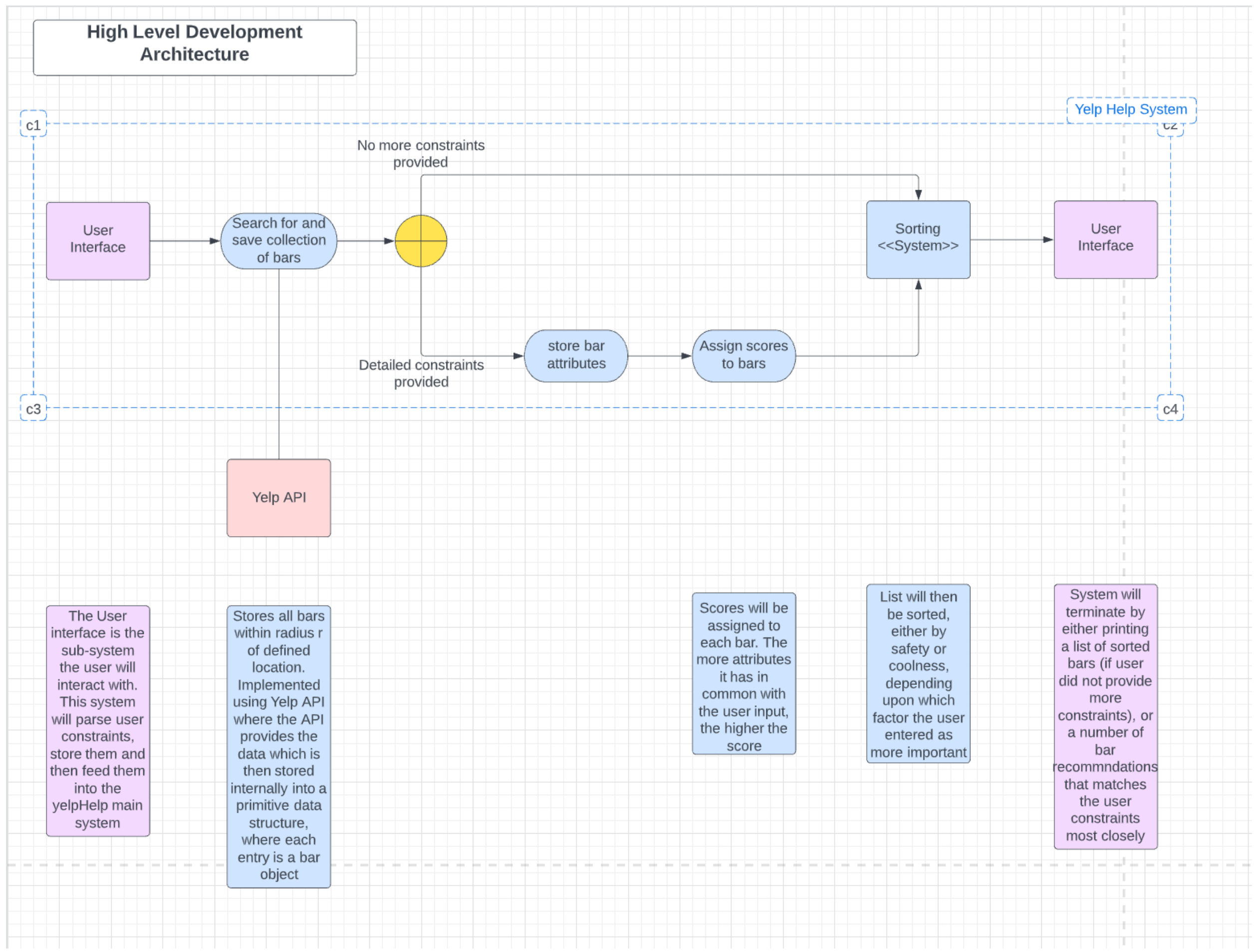
Behavioral View:



Input location and radius (zip code)
Coolness
Safety
Produce genRankIndex of top 50 or so bars in qualifying radius of specified location.
From top 50 bars check for existing indices
generate remaining indices from the coolSafeIndex
Store the new list into the database so it saves the list each time the code is run
Compute "error margins" and sort list from least to greatest bars
Output the top 5 bars from the sorted list and returnToUser
Recieve the user feedback and store in database to be used repeatedly and to update the weight of the indices in the coolSafeIndex

This is the behavioral diagram of how the overall code will be structured and operated. The essential structure of the code as displayed by the diagram involves the intake of user inputs and manipulation/indexing of bar data. The code uses a general rank provided by the API and extracts the top bars so that low ranking bars are not prioritized. It then checks a database to see if an index exists, and makes one if not and stores it in said database. This index produces a custom ranking and compares it to the respective user inputs. It then displays the top 5 bars and requests/stores user input to improve the algorithm.

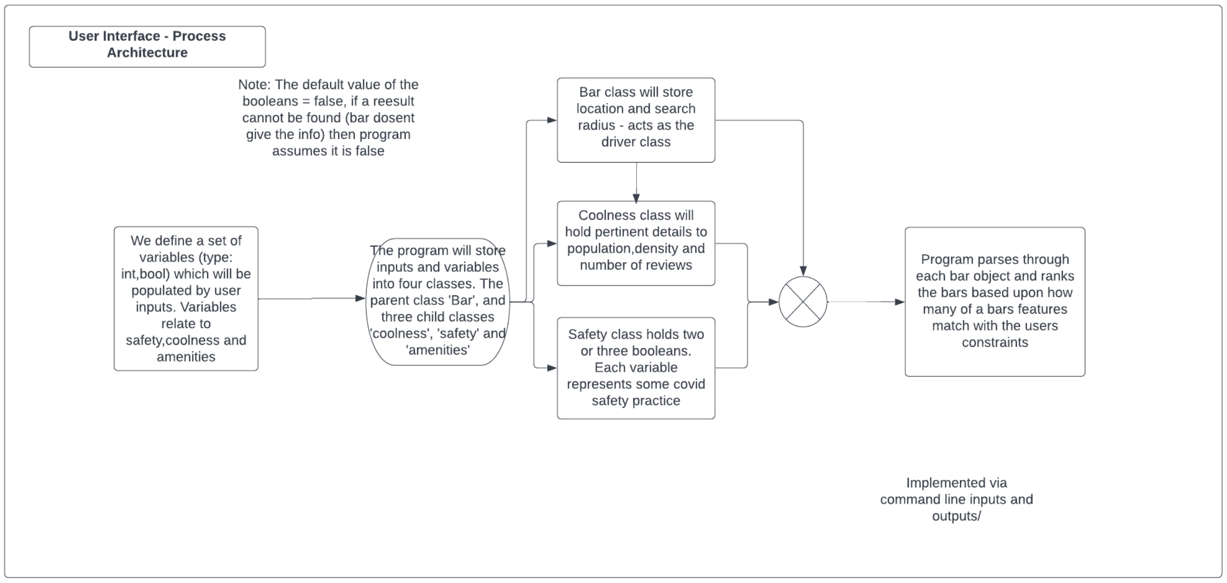
Development View:



High Level Development Architecture
Yelp Help System
c1
c2
c3
c4
User Interface
Search for and save collection of bars
No more constraints provided
Detailed constraints provided
store bar attributes
Assign scores to bars
Sorting <<System>>
User Interface
Yelp API
The User interface is the sub-system the user will interact with. This system will parse user constraints, store them and then feed them into the yelpHelp main system
Stores all bars within radius r of defined location. Implemented using Yelp API where the API provides the data which is then stored internally into a primitive data structure, where each entry is a bar object
Scores will be assigned to each bar. The more attributes it has in common with the user input, the higher the score
List will then be sorted, either by safety or coolness, depending upon which factor the user entered as more important
System will terminate by either printing a list of sorted bars (if user did not provide more constraints), or a number of bar recommndations that matches the user constraints most closely

The architecture given above focuses on the overall system from a development point of view. The architecture will consist of a user interface to gather inputs, using those inputs the system will interact with the Yelp API to generate a search request and store all relevant information. From there the system will go into one of two branches - detailed constraints or no detailed constraints. If the user-provided more constraints (to be defined later) then the system will store bar attributes gathered from data from the Yelp API (that data is stored locally at this point) and then the system will follow some algorithm and assign scores to each of the bars in the database. Finally, both branches will merge and we enter the sorting system where the list of generated bars is sorted by some metric (coolness or safety), lastly, the architecture culminates by providing this information to the user via a command-line interface. The details for each of the stages are also given in the architecture document.

Process View:


User Interface - Process Architecture
Note: The default value of the booleans = false, if a reesult cannot be found (bar dosent give the info) then program assumes it is false
We define a set of variables (type: int,bool) which will be populated by user inputs. Variables relate to safety,coolness and amenities
The program will store inputs and variables into four classes. The parent class 'Bar', and three child classes 'coolness', 'safety' and 'amenities'
Bar class will store location and search radius - acts as the driver class
Coolness class will hold pertinent details to population,density and number of reviews
Safety class holds two or three booleans. Each variable represents some covid safety practice
Program parses through each bar object and ranks the bars based upon how many of a bars features match with the users constraints
Implemented via command line inputs and outputs/

The architecture given above expands upon the specifics of accessing data from the API and storing it (from a process view and with a focus on the user's perspective via UI). Tracing the architecture we begin by defining some set of variables that will be populated by the data supplied from the API (a series of attributes for each bar class - phone number, rating, safety value, etc.). Then the program will parse through the fed data and begin storing all relevant data into one of the four defined classes (bar, amenities, coolness, safety). From there each class will follow its own set of protocols concurrently, where the bar class will store location and radius data, and the remaining classes will store attributes that are relevant to their overall nature (safety class stored safety attributes, etc.). Finally, we terminate the architecture by the final phase where the system parses through all objects and access attributes to rank all of the bars in the collection (based on user's inputs).

The Testing section is elaborated on in a separate document.

Evaluation:

Requirements:

- Is it straightforward to understand the requirements?

  The requirements are stated clearly and simply, and they are understandable. Each requirement expresses one thought per statement.

- Do the requirements meet the intended functionalities of the system?

The basic functionality of the system is to recommend a cool and safe bar. All the stated requirements in the sprint lead to the intended functionality of recommending a cool and secure bar.

- Are there any conflicting requirements?

All the requirements are stated consistently without contradictions between the requirements.

System modeling:

- Are system models available?

  The sprint includes different types of system models. It includes a context model, activity/process model, use case diagram, and sequence diagram. Each model presents the system from a different point of view. The context model includes a high-level block diagram that shows different systems used with YelpHelp. A process model is also included to show the processes in which the systems are used. The use case diagram and sequence diagram are used for interaction modeling. Use case diagram shows the system's actors and their interactions with the system. Sequence diagram shows the interactions between the user and the objects in the system.

- Do system models successfully provide a high-level description of the system from different views?

  Each model successfully provides an abstract description of the system from different aspects of the system. Each model has an explanation to help clarify and understand the model. For example, the activity/ process model describes the system process. First, the bar customer will input the location. Then a collection of bars will be stored within a certain radius of the user's location. Each bar in the collection will be assigned safety and coolness factors. After that, bars are sorted. Then the system will ask the user whether safety or coolness is more important. If the user chooses safety, the system will output the bar with the highest safety factor. If the user chooses coolness, the system will output the bar with the highest coolness factor. Then the user will be asked if he/she wants a list of bars sorted based on the chosen factor. If the user enters yes, then a list of sorted bars will be displayed.

<u>Architectural Design:</u>

- Is architectural design available?

  The architectural design is available, and each architectural model shows the system from a different point of view. The architectural views used in this sprint are conceptual view, behavioral view, development view, and process view.

- Do the architectural models provide an adequate description of how the system is organized as a set of communication components?

  Each architectural model shows how the system should be organized and designed from a different point of view, and each model is provided with an explanation to clarify how the system will be organized and designed. For example, the development view shows how the software is broken into multiple components for development. The architecture consists of a user interface to get inputs from the user. Then, the system will interact with the Yelp API to search for bars and store all relevant information. Then there are two branches, detailed constraints or no detailed constraints. If the user provided detailed constraints, then the system will store bar attributes and assign scores to each of the bars in the database. Finally, both branches will merge, and the sorting system will sort bars by coolness or safety. At the end, the system provides the user with bar's information