CAPSTONE 4ZP6A PROJECT

Software Requirements Specification Document

Version 1

FINDDIT

GROUP 19

Prakarsh Kamal

Japnit Singh

Kanwar Sandhu

Hriday Jham

Table of Contents

Notations and Abbreviations

Terms/Notation/Abbreviation	Meaning
PR	Price range
Super Dislike	An option given to users allowing them to veto a restaurant while voting. Limited to 1 super dislike per user
Group Leader	The person who creates the group
Preferences	Likings a user has and can update in their profile. This includes Price Range, Cuisine Type, and Location
Voting Deadline	A hard limit before which each user in the group must have voted on the restaurant choices
Swipe	The action performed by a user indicating their choice. The left swipe means a dislike. The right swipe means it is a like.

Contributions

In this table, we outline the contributions and roles of each team member:

Member	Role	Contribution
Prakarsh	Frontend Dev	Added the Notations/Abbreviations Table. Created the use case diagrams and wrote out Non-Functional Requirements.
Japnit	Backend Dev	Added information to Functional Requirements. Wrote out the Purpose and Client/Stakeholders sections. Also wrote the Data & Metrics section.
Kanwar	Frontend Dev	Added information to the Functional Requirements. Wrote out the use cases for Functional Requirements. Created the Contributions table.
Hriday	Backend Dev	Wrote out the Project Constraints and Risks section.

Purpose of the Project

Purpose:

The purpose of building this application is to enhance the social experience and decision-making process for groups of users when it comes to choosing a restaurant. The app aims to increase end-user happiness when making a collective decision in a group by streamlining the decision-making process through an interactive stack-based swipeable UI and a cost optimization function considering user preferences.

Goals:

The goals of building this application are to increase user happiness, reduce decision-making fatigue, and facilitate enjoyable group outings through an interactive application.

Client and Stakeholders

Clients:

Our Supervisor: Mehdi Moradi would be our client.

Stakeholders:

- Developers (Group members of the Capstone Group)
- End Users of the application
- Domain experts (Teaching Assistants, Mentors, and McMaster Alumni)
- Restaurant Managers and Owners

Project Constraints

Solution & Data constraint

We are creating a cross-platform native application that will work on both IOS and Android Operating systems, requiring the users to have a device supporting these operating systems and to have an active internet connection.
Our application depends on data returned by the Google Maps Nearby API to retrieve restaurant data such as location, reviews, and ratings which is vital for creating the required user experience. Hence the user locations supported would be based on the supported Google Maps API locations.

Resource constraint

The resource constraint for the capstone project will be the number of developers i.e. 4 group members and the time available. The project would not involve any contribution from an external source or freelancers.

Schedule constraint

The schedule constraint for this project is the 8-month duration of the school
year since we have to deliver a working product within this period of time.

Budget constraint

No external funding is required for the execution of this project. Our service relies on Google backend services, specifically Firebase and Firestore, and utilizes Google Cloud APIs. We have opted for Google Cloud's Free tier, which offers a generous API limit, ensuring that it does not have any financial impact on us.

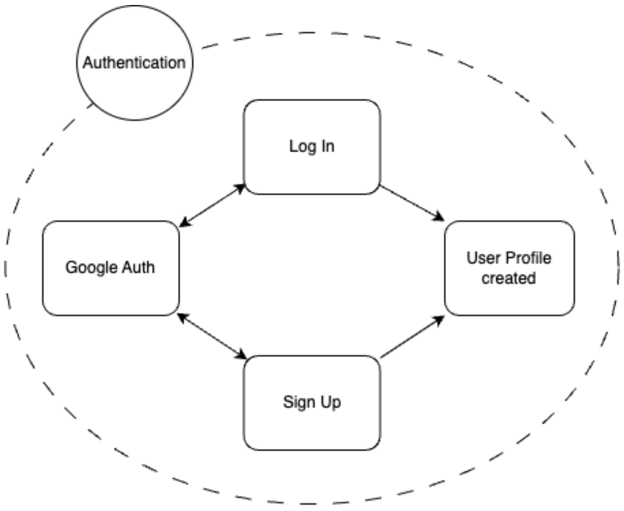
Enterprise constraint

No enterprise constraints for this project as it is self-funded and developed. There is no restriction on any technology stack being used.

Use Cases

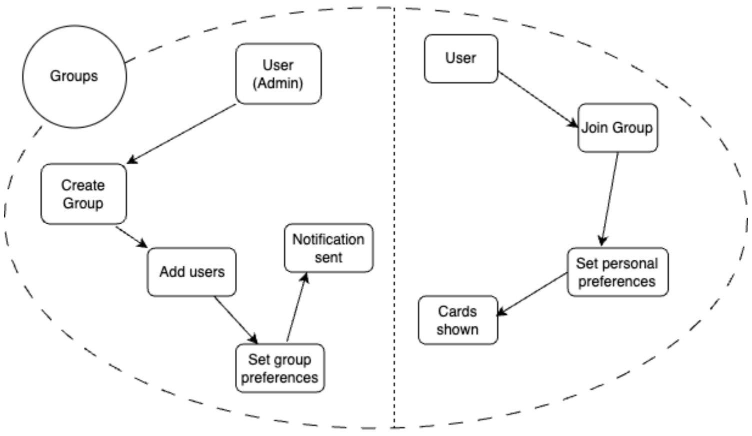
User Registration

The user will create an account on the app using the app's built-in sign-up method or by external providers such as Google Authentication. The app will ask the user about his/her username, password, name, and profile icon.



Group Creation

A user will act as an Admin and create a new group, add members, set group preferences for restaurants, and start a session for finding nearby restaurants.

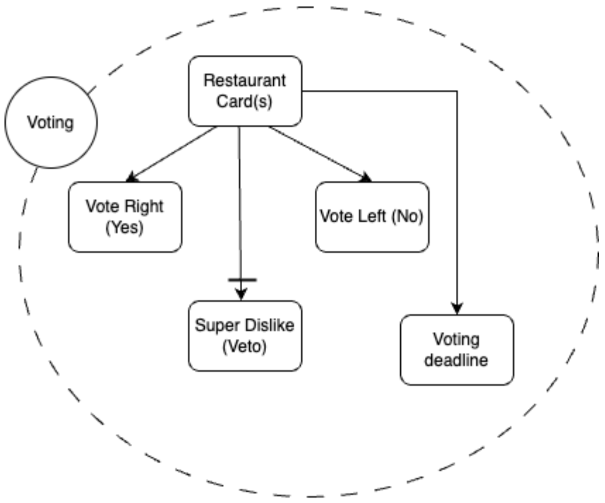


Join Group

If a user is not an admin, he/she can join a group as a member. The user will set personal preferences such as

- Distance they prefer to commute
- Price they are willing to spend

Swipe to Vote



After a session has started, a user can interact with the UI cards (of the restaurants generated by the app based on the set group preferences) to vote for a restaurant they prefer. The user can interact with the UI cards in the following ways:

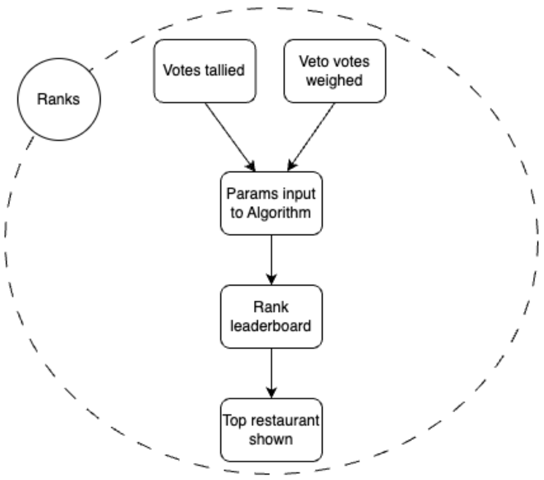
- Swipe Left to Downvote
- Swipe Right to Upvote
- Press the Super Dislike option to strongly reject the restaurant.

Switching between Groups

The user can switch between active groups if there is more than 1 group with live voting sessions.

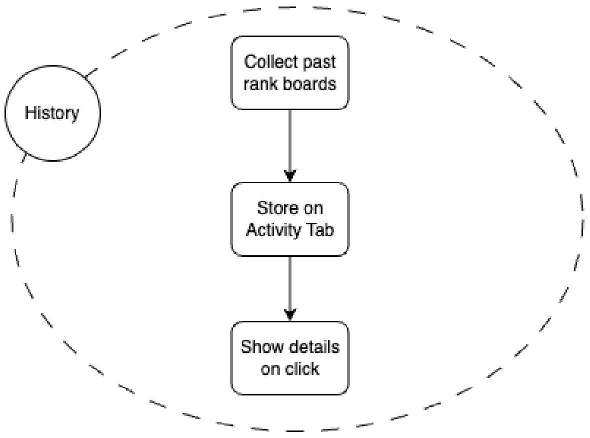
Ranking Leaderboard

After the voting is complete or on reaching the voting deadline, the group members will allow the app to make a decision for the group by displaying restaurants based on user votes, and user preferences optimizing the cost and increasing the happiness factor.



Accessing History

Users can access their previous group leaderboard results and view the leaderboard for any past expired session.



Functional Requirements

1. **Registration Page & User Profile**

   Requirement #: 1.1

   Requirement Type: Functional

   Event/use case: User Registration

   Description: The app shall allow users to create individual profiles with unique usernames and passwords for secure access. It shall also allow users to sign in through external providers such as Google

   Rationale: To ensure that user data and preferences are personalized and secured.

   Fit Criterion: Users can successfully register and create profiles with unique credentials, and user data is securely stored.

2. **Group creation**

   Requirement #: 2.1

   Requirement Type: Functional

   Event/use case: Group Creation

   Description: The app shall allow the user to create a new group as an admin/leader, set preferences & voting deadline for that group and add other logged in users to the group.

   Rationale: To create a session with different users and preferences for different occasions.

   Fit Criterion: The user can successfully create a group, add other signed-in users to the group, and set preferences.

3. **Users joining Active Sessions**

   Requirement #: 3.1

   Requirement Type: Functional

   Event/use case: Join Group

   Description: The app shall notify a logged-in user about their addition to the group.

   Rationale: Inform the user that they have been added to a group and direct the user to the group.

Fit Criterion: A logged-in user is informed about their addition to a group when they are added by a group leader.

Requirement #: 3.2
Requirement Type: Functional
Event/use case: Switching between Groups
Description: The app shall allow a user to choose a group out of multiple active groups with the user.
Rationale: Allow a user to navigate between multiple active groups with different preferences and interacting users.
Fit Criterion: The user is able to switch between different active groups through the UI.

Requirement #: 3.3
Requirement Type: Functional
Event/use case: Join Group
Description: The app shall give every user an option to set personalized preferences (scope: the current group) when added to a group that will be used by the algorithm.
Rationale: Get user preferences for each new group as user preferences which will later be used by the algorithm to rank restaurants.
Fit Criterion: A screen to set personal preferences for the group, that the user has joined.

4. Swipeable Stack UI to allow users to vote on options

Requirement #: 4.1
Requirement Type: Functional
Event/use case: Swipe to Vote
Description: The app shall create and show all restaurant cards to the users in the group via a Swipeable stack UI.
Rationale: Show the user all restaurant options with relevant information.
Fit Criterion: A logged-in user, who is a part of the group is shown restaurant options via a Swipeable stack UI when the preferences are set by the group leader.

Requirement #: 4.2

Requirement Type: Functional

Event/use case: Swipe to Vote

Description: The app shall allow users to vote on options with 3 main types of votes: Left swipe (decreasing the Restaurant score based on Algorithm), Right swipe(increasing the Restaurant score based on Algorithm) and one Super Dislike (to remove a restaurant from consideration).

Rationale: Use the UI to get the user's intention for each restaurant card shown to the user.

Fit Criterion: The user is able to provide intention for each of the restaurant cards through the UI. The user is also able to Super Dislike any one restaurant of choice.

5. Ranking Leaderboard

Requirement #: 5.1

Requirement Type: Functional

Event/use case: Ranking Leaderboard

Description: The app shall take user preferences, group leader preferences, and voting results into consideration and display a leaderboard to the users once the voting has finished or the voting deadline is met.

Rationale: Show restaurant ranking based on user voting results.

Fit Criterion: A logged-in user, who is a part of the group is shown restaurant leaderboards after voting has finished or the voting deadline is met.

6. Accessing History

Requirement #: 6.1

Requirement Type: Functional

Event/use case: Accessing History

Description: The app shall allow users to access up to 5 expired groups leaderboard results.

Rationale: Show expired leaderboard results for users' reference.

Fit Criterion: A logged-in user is able to see up to 5 previous groups' leaderboard results.

7. **Restaurant Cost Function Optimization Algorithm**

Requirement #: 7.1

Requirement Type: Functional

Event/use case: Create an algorithm to maximize happiness in the group.

Description: The app shall take in personal user preferences, group leaders' preferences, and voting results to assign a positive score to every Right Swipe/Upvote and a negative score to a Left Swipe/Downvote and rank restaurants on the leaderboard based on that score.

Rationale: Reduce unhappiness in the group by ranking restaurants higher with higher votes and closer to user preferences.

Fit Criterion: A logged-in user is able to see the ranked restaurant output by the algorithm.

Data & Metrics

Algorithm to Rank Restaurants to Maximize Group Happiness:

The cost of ranking restaurants in a big group is the unhappiness that is associated with the selection of a restaurant that does not align with personal user preferences. We will be creating an algorithm that aims at maximizing happiness in the group. This will be done as follows:
Preconditions:

- The group leader inputs preferences for the group which will be used to query the Google Maps API to find all restaurants matching the criteria.
- Each user in the group selects their personal preferences.

Algorithm:

- Every **restaurant (R)** in the pool will be initially assigned a **score (Z)** of 0.
- We will average out the personal **preferences of all users (X)** in that group.

- We will then assign a **Right Swipe (Upvote) Score (RS)** and **Left Swipe (Downvote) (LS)** score to each restaurant in the pool which will be unique to the restaurant.
- Every Right swipe **increases Z** by **RS** and every Left swipe **decreases Z** by **LS**.
- The RS & LS values will be calculated as follows:
    - RS is an arbitrary positive value that is **in-directly** related to the **(R-X) value** and **directly** related to the **ratings** and reviews of the restaurant. (the value determined by how close the average user preference is to the restaurant data)
    - LS is an arbitrary positive value that is **directly** related to the **(R-X) value** and **in-directly** related to the **ratings** and reviews of the restaurant. (the value determined by how close the average user preference is to the restaurant data)
- In the end, restaurants on the leaderboard will be ranked by the highest value of **Z**.

Google Maps Nearby Search API

We will be utilizing data returned by the Google Maps Nearby Search API to search for restaurants based on users input preferences.
This would be our primary data source which feeds the frontend cards.

Non-functional Requirements

- Look and Feel Requirements
    - Appearance:
        - The app will have an intuitive interface for all users
        - The app will employ an appealing design with enriching user experience elements and components such as buttons, modals, text fields, and cards.
    - Style:
        - The app's design will use a consistent color scheme.
        - The color scheme will be non-offensive
        - The app will be developed with modern minimalist styles and aesthetics
- Usability and Humanity Requirements
    - Ease of Use:

    - The app must be easy to use for any and all types of users
    - UI components shall be adequately sized to avoid misclicks on the screen
  - Personalization and Internationalization:
    - The app must allow the user to personalize their profile and update their picture and preferences
    - The app's text and language will be written in English (EN-US)
    - The price range will use the dollar sign ($)
    - The distance will be measured in kilometers (km)
- Performance and Speed Requirements
  - Speed and Latency:
    - The app must have fast response and load times aiming to respond under 100 milliseconds
  - Reliability and Availability:
    - The app must be available for the maximum amount of time and aim to reduce maintenance downtime as much as feasible
- Operational and Environmental Requirements
  - Requirements for Interacting with Adjacent Systems:
    - The app must be compatible and work across iOS and Android devices
- Security and Privacy
  - Access requirement:
    - Only the user will have access to their personal data. The developers will not have access to their personal information
  - Privacy requirement:
    - The app must conform with the Personal Information Protection and Electronic Documents Act (PIPEDA)
- Legal
  - Standards:
    - The app must follow the fail-safe-defaults design principle meaning if an error occurs, it will propagate into a safe state
    - The app must comply with the development standards for iOS and Android software development

Risk & Issues

- **Google server uptime:** Since our app's database and authentication is heavily dependent on Google servers, any server downtime on would negatively affect the functioning of our app.
- **Google Maps Nearby API deprecation:** As we are utilizing the Google Nearby API to search for restaurants in a location, the deprecation of the API would have adverse effects on the functionality.
- **Project incompletion due to time constraints:** This project is limited to a time frame of 8 months with loads of deliverables in that period. Any issues or roadblocks faced would have a snowball effect on the completion of the project.
- **Data accuracy:** Sometimes data from the API call can be outdated and cause a negative experience for users. For example, a restaurant that went out of business or menu changes are not reflected promptly in the API data.
- **Algorithm Unhappiness:** The algorithm we have created may sometimes have induced bias or incorrectness in the optimization calculation causing a user to occasionally experience unhappiness in the decision.

References

Office of the Privacy Commissioner of Canada. (2019, May 31). *Pipeda in brief.* https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/pipeda_brief/