**CS673 Software Engineering** 

**Team 1 - MenuMatch**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Dirgha Jivani | Team Leader |  | 09/25/2024 |
| Roshni Dodhi | Requirement Leader |  | 09/25/2024 |
| Prayushi Khandelwal | Security Leader |  | 09/25/2024 |
| Jiho Cheon | QA Leader | *JihoCheon* | 09/25/2024 |
| Pratyush Patel | Design Leader | *Pratyush Patel* | 09/25/2024 |
| Vignesh S | Configuration Leader | *Vignesh* | 09/25/2024 |
| Anshul Raj | Design and Implementation Leader | AR | 09/26/2024 |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| v1 | Dirgha Jivani, Roshni Dodhi, Prayushi Khandelwal, Jiho Cheon, Pratyush Patel, Vignesh S, Anshul Raj | 09/26/2024 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#_heading=h.gjdgxs)

[Related Work](#_heading=h.30j0zll)

[Proposed High level Requirements](#_heading=h.1fob9te)

[Management Plan](#_heading=h.3znysh7)

[Objectives and Priorities](#_heading=h.2et92p0)

[Risk Management (need to be updated constantly)](#_heading=h.tyjcwt)

[Timeline (need to be updated at the end of each iteration)](#_heading=h.3dy6vkm)

[Configuration Management Plan](#_heading=h.1t3h5sf)

[Tools](#_heading=h.4d34og8)

[Deployment Plan if applicable](#_heading=h.2s8eyo1)

[Quality Assurance Plan](#_heading=h.17dp8vu)

[Metrics](#_heading=h.3rdcrjn)

[Code Review Process](#_heading=h.26in1rg)

[Testing](#_heading=h.lnxbz9)

[Defect Management](#_heading=h.35nkun2)

[References](#_heading=h.1ksv4uv)

[Glossary](#_heading=h.44sinio)

# Overview

Our project focuses on creating a restaurant review platform specifically designed to cater to users with special dietary needs. The motivation behind this idea arose from the challenges faced by international students in Boston, many of whom adhere to specific diets such as vegan, gluten-free, or halal. Finding restaurants that meet their dietary requirements can be difficult, especially since general restaurant reviews do not provide specific insights into the suitability of certain dishes.

The purpose of this platform is to simplify the process of finding suitable dining options. Users will be able to filter restaurant reviews based on their preferences. The platform will display not only the overall rating of the restaurant but also a detailed rating for the specific food category the user is interested in, making it easier for them to choose the best dining option that fits their needs.

The primary users of this system are individuals with dietary restrictions or preferences, including international students, health-conscious individuals, or anyone looking for specific types of food. The platform could also serve as a valuable tool for tourists or locals exploring new food options in the area.

# Related Work

Google Maps is currently one of the most widely used platforms for finding restaurant reviews. While it provides reliable information on the overall quality of a restaurant, it lacks the ability to offer detailed insights into specific cuisines or dishes, particularly those that cater to special dietary needs.

Our system addresses this gap by offering curated reviews based on users' dietary preferences. Unlike Google Maps, which focuses on general restaurant ratings, our platform provides tailored reviews for specific food categories, allowing users to easily find restaurants that meet their dietary requirements.

# Proposed High level Requirements

* 1. Functional Requirements  
     (For each functional requirement, please give a feature title and a brief description using the following format: As (a role), I want to (action), so that (value).)

Essential Features

1. Food Type Selection
   * *As a user, I want to choose the type of food or dietary preference (e.g., vegan, halal, gluten-free), so that I can find restaurants offering food that matches my needs.*
   * Estimated effort: 15-20 person hours
2. Overall Restaurant Review
   * *As a user, I want to see the overall rating of a restaurant, so that I can quickly gauge the general quality of the establishment.*
   * Estimated effort: 10-12 person hours
3. Curated Reviews Based on User Choices
   * *As a user, I want to see curated reviews for the specific type of food I selected, so that I can assess how good the restaurant is for my dietary preference.*
   * Estimated effort: 20-25 person hours
4. User-Submitted Reviews for Specific Food Types
   * *As a user, I want to add reviews for a specific type of food at a restaurant, so that others can benefit from my experience.*
   * Estimated effort: 15-20 person hours
5. Mobile & Web Accessibility
   * *As a user, I want to be able to access the platform via both web and mobile devices, providing a responsive and user-friendly interface on each.*
   * Estimated effort: 15-20 person hours

6. User Registration & Profile Management

* + *As a user, I want to be able to sign-up, login, and manage the profile and update it anytime I want.*
  + Estimated effort: 15-20 person hours

Desirable Features

1. User Nationality Input
   * *As a user, I want to add my nationality to my profile, so that others with a similar taste palate can find relevant reviews.*
   * Estimated effort: 8-10 person hours

Optional Features

1. Food Flavor Descriptions
   * *As a user, I want to describe the flavors of a dish (e.g., sweet, spicy, savory), so that others can get a more detailed idea of the food’s taste.*
   * Estimated effort: 12-15 person hours

2. Food Photo Uploads

* *As a user, I want to upload photos of the food I ordered, so that others can see what the dishes look like before trying them.*

* 1. Nonfunctional Requirements
     1. Security & Privacy
        1. *The platform must implement standard security protocols to protect user data, including dietary preferences and location data.*
        2. *The platform must implement strong security protocols such as HTTPS for all communications to ensure data encryption.*
     2. Performance Requirements
        1. *The platform should be able to handle up to 10,000 concurrent users without a noticeable decline in performance.*
        2. *Search and filtering operations should return results within 2-3 seconds under normal load conditions.*
        3. *The platform should be optimized for fast loading times, with a page load time of less than 3 seconds for most users.*
     3. Scalability
        1. *The platform must be able to scale horizontally and vertically to accommodate an increasing number of users, restaurants, and reviews as the user base grows.*
        2. *The system architecture should support adding new features or expanding functionality without significant rework.*
     4. Availability
        1. *The platform must maintain a minimum uptime of 99.9%, ensuring high availability to users.*
        2. *In case of system downtime for maintenance, users should be notified in advance.*
     5. Usability
        1. *The user interface should be intuitive, with a clear and easy-to-navigate design.*
        2. *Both web and mobile versions should provide a consistent user experience, with a responsive design for varying screen sizes.*
     6. Compliance
        1. *The platform must adhere to data protection regulations such as GDPR for users based in the European Union, ensuring the proper handling of personal data.*

# Management Plan

### Objectives and Priorities

* + 1. Complete All Essential Features  
       The highest priority is to fully implement all essential features, including food type selection, overall restaurant reviews, curated reviews based on user choices, and the ability for users to submit reviews for specific food types. These features form the core functionality of the platform.
    2. Deploy the Software Successfully  
       Deploying the platform to a stable, user-accessible environment is critical. Ensuring that the system runs smoothly and is accessible across devices will allow users to benefit from its functionality.
    3. Ensure No Known Bugs  
       Delivering a bug-free experience is a top priority. We will focus on thorough testing to ensure the platform operates smoothly without any major issues.
    4. Maintain High Quality  
       The platform should offer a clean, intuitive user experience with fast, reliable performance. This includes high-quality UI design, seamless navigation, and accurate, relevant results for user queries.
    5. Implement Desirable and Optional Features  
       While not essential, implementing desirable features like user nationality input and optional features like flavor descriptions and food photo uploads will further enhance the user experience. These will be developed after the core functionality is stable.

## Risk Management (need to be updated constantly)

The major risk associated with this project is team management and assigning the tasks efficiently. Another risk is related to the requirement of the project. If the requirements change or are not listed properly, then we will have to accommodate accordingly and that might be costly.

**Risk Management Sheet Link:** [CS673\_SPPP\_RiskManagement.xlsx](https://docs.google.com/spreadsheets/d/1R8j3Obsf0ZWHe-G8EJi553TDFSyxVDhu/edit?gid=23484764#gid=23484764)

## Timeline (this section should be filled in iteration 0 and updated at the end of each later iteration)

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | Get restaurants info(menu, reviews, etc) and enables recommendation based on food type selection | Get restaurant informations using Google Maps API and add missing informations manually / categorize menus and restaurants based on type / embed reviews as numerical value | 80-100 hrs |
| 2 | Mobile & Web Accessible | Build our own website and make it mobile accessible if possible | 80 hrs |
| 3 | Profile Management & add desired features | Enable user sign-in and make the web to remember one’s previous inputs. Also add desired features such as nationality input and photo reviews | 80-100 hrs |

# Configuration Management Plan

## Tools

* Git/Github: We would be using Git/Github for version control or repository management
* JIRA: We will be using JIRA for task management and time tracking
* AWS: We will be using AWS for hosting our application
* Jenkins: We will be using Jenkins for deployment of our application
* Docker: We will be using docker for containerization of our application
  1. Code Commit Guideline and Git Branching Strategy  
     For Git Branching Strategy we will use the Github Flow since it is more suited for smaller teams. According to this strategy we will have a main branch that should be deployable at all times. Developers can create a new feature branch from this main branch and work on their feature isolated from the main branch and other branches. We will also have a peer review before merging the branch to the main branch. Instead of having a fixed reviewer we can be a reviewer for someone else, but having it reviewed before merging is necessary. We will also have one or two dedicated repository owners who will only have the access to write to the main branch. They can review the comments by the reviewer and merge it back to the main branch if they feel everything is okay. Also there will be an automated test stage in the CI/CD pipeline which the feature branch must pass before being merged to the main branch. This way we can protect the main repository from bugs and it will be deployable always.

We will also have a pre-commit hook which would be required to run before committing any code or else the test pipeline will fail and the feature branch could not be merged. This pre-commit hook will have code quality checks.

## CI/CD Plan if applicable

We are planning to deploy our application on an AWS ec2 instance and we are also planning to automate the deployment using Jenkins. Jenkins can even test the code and then deploy automatically if the test stage has passed or notify the developers otherwise. We can also run the pipeline on the main branch at a fixed interval like once a week or twice a week.

# Quality Assurance Plan

## Metrics

(Describe the metrics to be used in the project to measure the quality of your software. Each metric should be measurable and quantifiable. Examples of metrics include product complexity (LOC, # of files, # of classes, # methods, cyclomatic complexity, etc.) , defect rate (# of defect per KLOC), # of test cases, test case pass rate, cost (# of person hours used), # of user stories completed, etc. **The result of these metrics should be reported in the progress report/ iteration summary sheet.**)

| Metric Name | Description |
| --- | --- |
| Lines of code (LOC) | Measures the total number of lines in the source code. This helps assess the size and complexity of the software. |
| Cyclomatic Complexity | Measures the complexity of the program by counting the number of independent paths through the code. A higher value indicates more complex code, which could lead to more defects. |
| Defect Rate | Measures the number of defects per 1,000 lines of code (KLOC). This helps track how many bugs are reported during the project. |
| # of test cases | Track the total number of test cases created for the project to ensure adequate test coverage of all features and functionalities. This helps measuring how well the software is validated and ensuring that critical paths are thoroughly tested, minimizing the risk of defects in production. |
| Test Case Pass Rate | Measures the percentage of passed test cases out of the total number of test cases. Indicates how stable and reliable the software is after testing. |
| Cost(# of person hours used) | Tracks the total number of hours spent on development and testing. Helps measure productivity and cost of the project. |

* 1. Coding Standard

We will use Python as our main programming language, and we will adhere to the PEP8 standard for Python coding. This includes following conventions for naming variables, classes, and methods, as well as maintaining proper indentation, commenting, and documentation throughout the codebase. Also, all code will be committed to the team’s GibHub repository. Team members must regularly push their code and use clear, descriptive commit messages. Branching strategies will be used to ensure smooth collaboration and avoid conflicts.

## Code Review Process

We will follow the peer review process where everyone can review each other’s code. We are still deciding on the checklist which should be used by the reviewer. But we will have designated persons who would be responsible for merging the code back to the main branch who in turn can review the code reviewers’ notes and decide whether we can merge this to the main branch or it needs anything else. Also with the pipeline there will be an automated test run on the branch and it is required to pass the test stage of the pipeline for the code to be accepted.

## Testing

Both manual and automated testing will be used in this project. Developers will write and run unit tests using pytest to ensure that individual components function correctly, while the QA leader will focus on integration testing to verify that different modules, such as Google Maps API and the restaurant reviews system, work together successfully. Manual testing, using Selenium, will be conducted to ensure the website’s interface and user experience operate as intended across devices and browsers. Unit tests will be executed continuously during development, and integration tests will be performed after major features are implemented. The main testing objectives are to ensure the accuracy of the recommendation system, maintain code quality, and verify that all components function cohesively.

## Defect Management

We will use GitHub Issues to manage defects throughout the project. Defects to be tracked include functional bugs (e.g., incorrect restaurant recommendations or Google Maps errors), UI/UX issues (e.g., poor layout or navigation problems), and performance defects (e.g., slow loading times or API response delays).

Each team member will be responsible for reporting defects they encounter during development or testing. The QA Leader will triage and assign defects based on their severity and priority. Once a defect is assigned, the developer responsible will work on resolving the issue and submit a fix via a pull request. The QA Leader will verify that the defect is resolved before closing the issue.

# References

* GithubFlow - <https://www.abtasty.com/blog/git-branching-strategies/>

# Glossary