# Professional Software Development (H)

Team I

# Project Specification for Resdiary.com

## Outline

The current resdiary.com portal allows diners to search for restaurants and available tables for dining. There is a system in place on the site for users to restaurants in their area but ResDiary would like a recommendation engine to be added to recommend new restaurants based on a diner's previous dining habits and their similarity to other diners. The goal of this project is to create a recommendation engine that accurately suggests places to eat based on previous restaurants and other people with similar preferences.

## Functional Requirements

- The  engine must accurately suggest places to eat based on the user's own previously visited restaurants and other users with similar eating preferences.
- Recommendations should be in close proximity to where the user typically eats or is currently searching.
- The engine should re-recommend previously visited restaurants provided the user selects this option.

## Non-Functional Requirements

- The engine should be written in Python for easy integration within existing Resdiary systems.
- Provided the data is not stored locally, the system should perform nightly updates to the recommendations made.
- New users should be presented an optional quick quiz/questionnaire to gather initial data.
- User locations and restaurant recommendations should be interpreted using coordinates rather than city/town name as those could be of arbitrary precision.
- Have the ability to "fine tune" the recommendation engine by altering the weighting of the significance different components of the recommendation such as distance, price, reviews, etc.

## Soft Goals

- Ultimately the goal of the system is to make good recommendations to users of the ResDiary service so that they can enjoy a new dining experience.
- The system itself (regardless of how the data is processed and stored) should provide the recommendations in a timely manner to users.

## Privacy, Security & Use of User Data

As brought up by one of the lecturers, due to the nature of this project the issue of data security, privacy and use of user data must be addressed formally in some regard.

Regardless of whether the issue falls on the customer’s part as they are giving us the data and so may be the ones required to ensure all laws are met the issue must still be addressed and the outcome clear.

We may be required to display a message to new users / existing users informing them that their booking history is being used to make recommendations but all data used is anonymous and secure.

The issue should be explained again at the start of the dissertation report and the outcome and measures taken within the project detailed in full.

Thus far the following have been taken:

- Initial small subset of data (csv files) were provided via email from the customer. Measures were taken on their end to ensure all the data was anonymous and did not contain any public information.

**User Stories**

| ID | TYPE | As a... | I want to... | So that I can... | Notes | Priority |
|---|---|---|---|---|---|---|
| 1 | Func. | Existing user | Have recommendations based off my similarity to other users | Be recommended restaurants from users with similar tastes to my own | | High |
| 2 | Func. | New user | Be able to take a questionnaire | Get an initial recommendation as a new user | | High |
| 3 | Func. | User | Be able to disable re-recommendations | Always be provided with unvisited restaurants | | High |
| 4 | Func. | Existing user | Be able to refine my recommendations settings | Restrict the types of recommendations made to me | (See example below) | Med |
| 5 | Func | User | Have recommendations take my budget into account | Restaurants are recommended within my price range | | Low |

NOTE: Regarding ID 4, an example of this implementation might be to allow users to turn on dietary restrictions such as only recommend me vegetarian or vegan restaurants as this might be an issue for some users.

**Split stories into tasks**

- Draw up a database schema for storing relevant, existing user data.

- Set up a database based on the created schema for storing the relevant, existing user data.

- Make a questionnaire for new users to generate initial data:
    - Determine the questions asked to new users in the new user questionnaire. Note: Preferred and non-preferred cuisine types should be less significant and not the focus as suggested by customer.
    - Create a basic front-end involving simple yes/no questions and location fields for new users of the site. (Customer stated Netflix new user quiz as an example of how they imagined it would be incorporated for the existing site)
    - Set up the back end for data persistence.
    - Process the gathered data and make some initial recommendations to new users.

- Make a refinement option for existing users to refine recommendations:
    - Determine the options which users are allowed to set in order to refine their recommendations.
    - Create a basic front-end including simple yes/no questions and location fields to present to new users.
    - Set up the back end for data persistence.
    - Process the gathered data to make refinements to the recommendations made to a particular user.

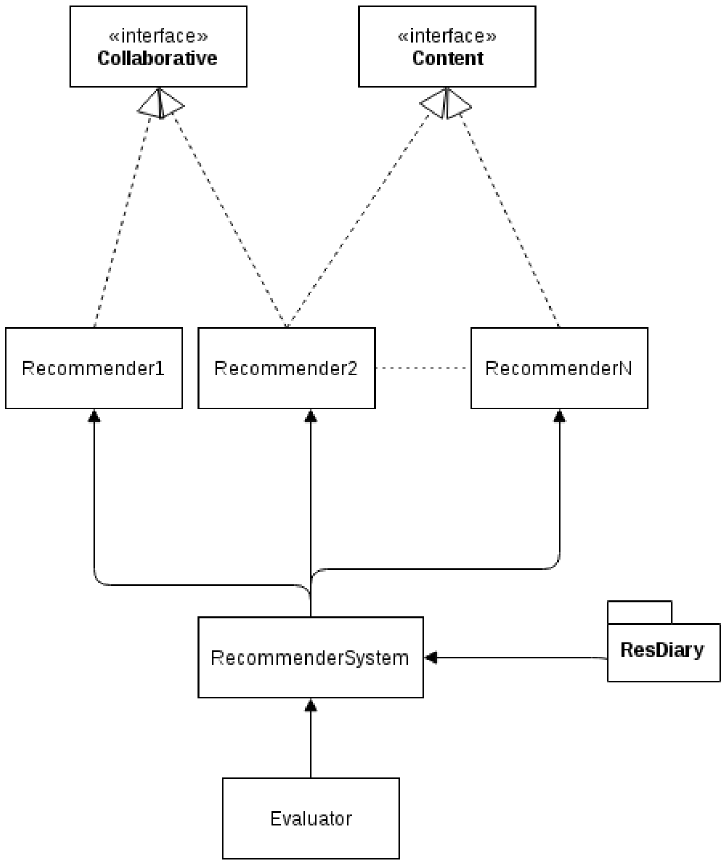
- Write the recommendations engine, taking the following parameters into account:
    - Similarity to user others -- customer stated this was key basis they wish recommendations to be made off
    - Other user reviews
    - Other user favourite restaurants
    - Current or typical location
    - Critic reviews
    - Restaurant availability

  Customer stated that “Cuisine Type” and “Price” should not be as significant a weighting as was originally suggested at the project proposal meeting. While still a feature of the recommendation it should not be as significant as we anticipated.

- Create a stand-alone front-end prototype for displaying recommendations

- Integrate the engine into resdiary.com

## High-Level System Design



## High-Level System Workflow

1. ResDiary sends a User object to our Recommender System, which contains all their previous reservations and reviews.
2. Recommender System sends that object to all Recommenders in the system.
    a. Some of the Recommenders are collaborative: they find similar users and recommend restaurants that those users liked.
    b. Some are content-based: they look for similarities between restaurants and recommend similar restaurants based factors such as price range and cuisine.
    c. Some Recommenders can combine both approaches.
3. Each Recommender returns the top N recommendations to the Recommender System.
4. Recommender System combines them based on some predefined coefficients and outputs the final recommendations.

Evaluator class can be used to judge system performance. For example, the restaurant review data could be split based on timestamps. Earlier data can be provided to our Recommender System. The System recommendations can then be checked against later bookings and reviews. The Evaluator outputs the percentage of bookings that our system guessed right.