**Taste Navigator of Bay area’s Asian food**

Problem Statement:

In the multicultural setting of the US, especially in areas with a high concentration of international students and residents such as the Bay Area, finding restaurants that cater to specific culinary tastes can be challenging. The goal is to develop a Java-based software system that allows users to discover and reserve restaurants based on personalized food preferences, with an emphasis on Asian cuisines.

1. Java Project Structure:

1. User Account Management:

- Classes: `User`, `UserProfile`, `Authentication`

- Functionalities:

- Register/Login system with username and password.

- Profile management for identity, address, and preferences.

2. Restaurant Information:

- Classes: `Restaurant`, `Menu`, `Review`, `Reservation`

- Functionalities:

- Store and display restaurant details, including cuisine type and menu items.

- Handle customer reviews, ratings, and reservation bookings.

3. Search and Recommendation Engine:

- Classes: `SearchEngine`, `RecommendationEngine`, `FilterCriteria`

- Functionalities:

- Allow users to search by price range, cuisine, number of people, tags related to cultural identity, and ratings.

- Generate personalized recommendations based on user preferences and past reviews.

4. Utility and Interface:

- Classes: `DatabaseConnector`, `UserInterface`, `ReservationSystem`

- Functionalities:

- Connect to a database to retrieve and store information.

- Provide an easy-to-use interface for account settings and restaurant searches.

- Manage reservations with time, date, and user details.

1. Software Design Principles to be used:

- Object-Oriented Programming (OOP):

- Encapsulation: Use private variables and public methods to access the data.

- Inheritance: Create a base class for common attributes and extend them in subclasses for specific functionalities.

- Polymorphism: Use interfaces and abstract classes to provide multiple implementations for the recommendation engine.

- Abstraction: Hide complex logic behind simple interfaces.

- Object-Oriented Design (OOD):

- Identify the system components and their relationships.

- Use UML diagrams to visualize classes, interfaces, and their interactions.

- Apply design patterns where appropriate, like Singleton for database connections and Factory for object creation.

To implement this project, we will use Java as the main programming language, with potential use of SQL for database management and Swing or JavaFX for the graphical user interface. The recommendation engine could use a simple algorithm based on our user preferences and ratings, or a more complex machine learning approach if desired.