CSEN 275 Recipe Application

Team members:

Sathyasai Ajitesh Apparao Tamirisa - W1652100
Neeraj Menon - W1650116
Isha Pednekar - W1650945

Requirement Analysis for Recipe Sharing Application

1. User Registration and Profiles:
   - Description: Users should be able to register for an account and create personalized profiles.
   - Functional Requirements:
     - Users can sign up with a unique email and password.
     - Users can provide additional details for their profile, such as dietary preferences and allergies.

   - Non-functional Requirements:
     - User passwords should be securely hashed and stored.
     - User registration forms should include validation checks for input data.

2. Recipe Creation and Editing:
   - Description: Users can create, edit, and share their recipes.
   - Functional Requirements:
     - Users can create new recipes by providing details such as title, ingredients, measurements, cooking steps, and images.
     - Users can edit existing recipes to update details or make corrections.
     - Users can share their recipes with other users of the platform.
   - Non-functional Requirements:
     - Recipe creation forms should be user-friendly and intuitive.
     - Images uploaded with recipes should be properly compressed and optimized for web viewing.

3. Recipe Categorization and Tags:
   - Description: Recipes can be categorized into different types and tagged with relevant attributes.
   - Functional Requirements:
     - Recipes can be categorized into predefined types such as appetizers, main courses, and desserts.

- Users can tag recipes with attributes such as dietary preferences, cuisine types, and occasions.
- Non-functional Requirements:
  - Categories and tags should be managed efficiently to ensure consistency and ease of navigation.

4. Search and Filter:
   - Description: Users can search for recipes based on keywords, ingredients, or categories, and apply advanced filtering options.
   - Functional Requirements:
     - Users can perform keyword searches across recipe titles and descriptions.
     - Users can filter recipes based on dietary restrictions, cuisine types, cooking time, and other attributes.
   - Non-functional Requirements:
     - Search and filter functionalities should provide accurate and relevant results on time.

5. Personalized Feed and Recommendations:
   - Description: Users receive personalized recipe recommendations based on their preferences, follows, and activity.
   - Functional Requirements:
     - The system analyzes user preferences, follows, and activity to generate personalized recipe recommendations.
     - Users can discover new recipes through a personalized feed tailored to their interests.
   - Non-functional Requirements:
     - Recommendation algorithms should be efficient and scalable to handle large user bases.

6. User Ratings and Reviews:
   - Description: Users can rate and review recipes, with aggregated ratings highlighting popular and well-reviewed recipes.
   - Functional Requirements:
     - Users can rate recipes on a scale (e.g., 1 to 5 stars) and provide written reviews.
     - The system aggregates ratings to highlight popular recipes and calculate average ratings.
   - Non-functional Requirements:
     - Ratings and reviews should be moderated to prevent abuse and ensure authenticity.

Overall, the Recipe Sharing Application should provide a seamless user experience for discovering, sharing, and collaborating on culinary creations while ensuring the security and integrity of user data and interactions.

Object Oriented Concepts Used in the project

The project incorporates numerous object-oriented programming principles, pivotal to its architecture and functional design, particularly evident within a Spring Boot framework context. Here is a detailed elucidation of the object-oriented concepts manifested within the project:

Encapsulation: The project's classes, such as Recipe and User, encapsulate their respective data and the methods to manipulate this data. This encapsulation ensures that object data is shielded and interacted with only through defined interfaces, enhancing data integrity and security.

Inheritance: This project leverages inheritance extensively, particularly through the Spring framework. For instance, service classes may extend or implement Spring's classes or interfaces, enabling shared behavior and properties across various entities and services.

Polymorphism: The application showcases polymorphism through interfaces like UserRepository. This abstraction layer allows the application to utilize any implementing object interchangeably, facilitating a flexible code structure that can accommodate various implementations without necessitating core changes.

Abstraction: Interfaces within the project, exemplified by UserRepository, embody abstraction by defining a set of methods that any implementing class must execute. This design abstracts the what from the how, focusing on the operations required without delving into their specifics.

Association: Relationships between entities, such as between Recipe and User, illustrate association, delineating how objects can relate and interact within the application, thereby facilitating coherent data navigation and manipulation.

Dependency Injection: Utilizing this design pattern, the project employs @Autowired annotations to inject dependencies, emphasizing reliance on abstractions rather than concrete classes. This approach enhances modularity and testing ease, key for maintainable code.

Composition: Should there be closely interlinked entities where one cannot exist without the other, the project demonstrates composition, a strong "has-a" relationship, further emphasizing the intertwined nature of certain objects and their dependencies.