Technologies Used:

The backend services are written in Spring Boot with Java and the user progress is stored in a MySQL database.

Notes:

I had to download the whole set as I didn't know which jar files were suitable.

Classes and Packages:

The application is structured into the following packages and classes:

Here are the packages and class names I use:

1. **com.rowmatch.model**
   * **User**
   * **Tournament**
   * **Leaderboard**
2. **com.rowmatch.repository**
   * **UserRepository**
   * **TournamentRepository**
   * **LeaderboardRepository**
3. **com.rowmatch.controller**
   * **UserController**
   * **TournamentController**
   * **LeaderboardController**
4. **com.rowmatch.service**
   * **UserService**
   * **TournamentService**
   * **LeaderboardService**
5. **com.rowmatch.util**
   * **TimeUtil** (for handling time-limited tournaments)
   * **CoinUtil** (for handling coin rewards)

com.rowmatch.model

User: This class represents the user and contains information such as user id, level, and coins.

Tournament: This class represents the tournament and contains information such as the start and end time of the tournament.

Leaderboard: This class represents the leaderboard for a specific tournament group and contains information such as the user id, username, and tournament score.

com.rowmatch.repository

UserRepository: This interface extends JPA repository and is responsible for performing CRUD operations on the User entity in the database.

TournamentRepository: This interface extends JPA repository and is responsible for performing CRUD operations on the Tournament entity in the database.

LeaderboardRepository: This interface extends JPA repository and is responsible for performing CRUD operations on the Leaderboard entity in the database.

com.rowmatch.controller

UserController: This class exposes API endpoints for creating a user, updating the user's level, and retrieving the user's progress.

TournamentController: This class exposes API endpoints for entering a tournament, claiming rewards, getting the rank of a specific player, and retrieving the full leaderboard data.

LeaderboardController: This class exposes API endpoints for retrieving the leaderboard data of a specific tournament group.

com.rowmatch.service

UserService: This class implements business logic for creating a user, updating the user's level, and retrieving the user's progress.

TournamentService: This class implements business logic for entering a tournament, claiming rewards, getting the rank of a specific player, and retrieving the full leaderboard data.

LeaderboardService: This class implements business logic for retrieving the leaderboard data of a specific tournament group.

com.rowmatch.util

TimeUtil: This class is responsible for handling time-limited tournaments.

CoinUtil: This class is responsible for handling coin rewards.

Design Decisions:

Database: MySQL is used to store the user progress as it provides good performance and reliability for many users. JPA is used to interact with the database and perform CRUD operations.

Exception Handling: Custom exceptions are used to handle exceptional cases in the application, such as entering a tournament before reaching level 20 or claiming a reward for a tournament that has not been entered.

Security: Spring Security is used to secure the API endpoints and prevent unauthorized access.

Performance: Caching is used to improve the performance of the application, especially for frequently accessed data such as the leaderboard data.

Unit Testing:

JUnit is used to write unit tests for the User, UserController, and UserService classes to ensure the functionality of these components. These tests cover cases such as creating a user, updating the user's level, and retrieving the user's progress.

Conclusion:

In conclusion, the implementation of Row Match's backend system requires careful consideration of various factors such as performance, security, concurrency, and readability. The classes and packages have been organized in a logical and easily understandable structure, with each component serving a specific purpose. The user operations have been tested using JUnit tests, ensuring the functionality and reliability of the system. It is also important to continuously monitor and improve the system, especially when dealing with a high volume of users and data. Additionally, proper documentation of the design decisions and implementation can greatly aid in maintaining and updating the system. Overall, the goal is to provide a fast and secure API that enhances the user's experience in playing Row Match.