Self-service Micro Market Task RESTful API

## Introduction

## Why a programming task?

To get an impression of your coding skills we are asking you to do what you do best - write some code. Solving the following task will give us a basic understanding of your style and will be used as a basis for further questions and open technical discussion. There is no right or wrong solution, we just want to see how you will approach the problem.

## General overview of the task

Plan, document and develop an AI-enabled Self-Service Micro Market system named “mMarketAI” that allows clients to buy food and beverages using banknotes, coins, card transactions or cryptocurrencies. Please, make sure you test your services thoroughly to ensure correct functionality and proper error handling.

## Useful tips

1. Make sure you have all the required IDEs JDKs, and tools installed.
2. Start by planning the tasks and separate them into smaller manageable tasks with clear implementation details for each task.
3. Focus on the RESTful API first has the highest priority.
4. Choose coding standards for your project and stick to them. (Naming conventions, etc.)
5. Do not commit unused or ide specific files to the version control system (You can use gitignore file search in GitHub and find one suitable for your IDE)
6. Format the code consistently throughout the project. (You can implement editor config)

## Requirements

## B. I. Functional requirements

### Must-have:

1. The system must have CRUD API for all the stored items (food or drinks).
2. API for checking out of the items.
3. The mini market must have the quantity for each product and have a fixed quantity for each item.
4. The system must store each order of each user or anonymous user for statistical purposes.
5. The system must accept the following payment methods:
   * Coins 10, 20, 50 st , 1BGN coin, 2BGN coin;
   * Banknotes 5BGN, 10BGN, 20BGN 50BGN;
   * If using coins and banknotes it shall calculate and return change to the customer in the form of coins only starting by the biggest possible monetary value available in the coin changer;
   * Accepts payments from all the currencies using card payment equivalent to Bulgarian lev with exchange rate for euro to BGN 1,92;
6. Implement login and register. (cloud be a separate microservice)

### Should-have:

1. Implement food AI recommendations form the available items.
2. Implement food plan with the available meals for the following fitness goals: (Use AI to generate food plan based on users’ weight height goal, calorie intake):
   * weight loss, muscle building, endurance.
3. Introducing subscription plans including variety of personalized meals for breakfast, lunch and dinner yearly, monthly, weekly
4. Functional tests, unit tests (Use JUnit) add at least 50% test coverage of the services and Integration tests of the APIs.
5. API that provides: Statistics of most profitable subscription plan for a given month and year.
6. API that provides: Statistics of most profitable food item for a given month.
7. Roles and Permissions for each endpoint (Roles: Administrator, Client, Anonymous).
8. Add User management API for the administrators to check food items availability.
9. Implement reset password flow.
10. React responsively with design / or Thymeleaf

### Could-have:

1. Implement a point reward system for each purchase you gather points (if you are registered) to exchange for voucher for 5 BGN or 10 BGN.
2. The system could have barcode scanner functionality to register the shopping cart items.
3. Implement a Notification system (cloud be a microservice) that can:
   * Notifies the administrator that micro market is low on certain items;
   * Send email registered users for promotions;
   * Send email to the registered users for new available items.

## B. II. Non-functional requirements:

### Must-have:

1. Spring Boot 3.x or higher.
2. Java 21 or Higher.
3. Maven 3.8.x or higher / Gradle 7.6.x or higher.
4. Database compatible with hibernate (PostgreSQL).
5. Hibernate ORM.
6. Lombok.
7. MapStruct.
8. Configured Spring security.
9. Database migration implementation and scripts (Flyway or Liquibase)
10. Version control (GitHub, Gitlab, Bitbucket, etc.).
11. Create main branch and development branch for the new stories/features and bug branches for the new bugfixes).
12. Keep the commit messages relevant to the tasks / bugs / features that they worked on. Start with the Jira story number then the story title example:
    * + - PROJECT-342-Implement-admin-panel.
13. Docker or Kubernetes.
14. Documentation – in the form of ReadMe file containing how to build, start, test the project including prerequisites and tools required.

### Should-have:

1. Description of Request / Response bodies and URI Paths and the DTOs [For example: Swagger Open API specification].
2. Example / Test data in the database for demonstration purposes.
3. Prepared demonstration of all the functionality.
4. Prepared demonstration of the functional tests.
5. Prepared demonstration of the performance tests (If implemented)

### Could-have:

1. Performance tests (for example use k6) (Load test, Spike test, Stress test, Soak test).
2. Kubernetes / Helm chart for the whole infrastructure of the system (If you chose this option do not add docker swarm or docker-compose.yaml).
3. Pipeline (Jenkins/Gitlab Runner etc.).
4. Agile Board with the task separated in 2 sprints for the 2 weeks on the project

**Resources :**

* + - 1. AI <https://www.baeldung.com/spring-ai-model-context-protocol-mcp>
      2. Gradle <https://docs.gradle.org/current/userguide/installation.html>
      3. Maven <https://maven.apache.org>
      4. Java 21 <https://jdk.java.net/java-se-ri/21>
      5. RestFUL <https://restfulapi.net/>
      6. SWAGGER <https://swagger.io/specification/>
      7. Spribng io <https://start.spring.io/>
      8. K6 <https://k6.io/>
      9. React <https://react.dev/reference/react>
      10. Spring <https://docs.spring.io/spring-boot/documentation.html>
      11. Spring security <https://spring.io/projects/spring-security>
      12. Hibernate <https://hibernate.org/orm/documentation/6.6/>
      13. Kubernetes <https://github.com/canonical/microk8s>
      14. IDE for Java [https://www.jetbrains.com/idea/download/](https://www.jetbrains.com/idea/download/?_gl=1*1lij7dt*_gcl_au*MjQ0MjIxMTcyLjE3NDMxNTMwNDQ.*FPAU*MjQ0MjIxMTcyLjE3NDMxNTMwNDQ.*_ga*MTMzNDA2NDY5MC4xNzQzMTUzMDQy*_ga_9J976DJZ68*MTc0MzE1MzA0MS4xLjEuMTc0MzE1MzA5NC45LjAuMA..)
      15. IDE for React <https://code.visualstudio.com/download>
      16. Mapper <https://mapstruct.org/>
      17. Lombok <https://projectlombok.org/>
      18. MVN repo <https://mvnrepository.com/>