Implementation plan

Melike Aytaç-21070006056
Sultan Esen Murat-21070006039
Yağmur Sabırlı-21070006017
Beren Elçin Polat-22070006064

Table of Contents

1 Overview

This document outlines the Implementation Plan for the Restaurant Automation System project, developed during the Spring 2025 semester. The purpose of the system is to simplify order processing and inventory management for small to medium-sized restaurants. By automating repetitive and error-prone manual tasks, the system enhances operational efficiency, reduces human errors, and supports smoother workflows.

Team Members:

- Beren Elçin Polat
- Melike Aytaç
- Yagmur Sabırlı
- Sultan Esen Murat

2 Milestones

Each milestone corresponds to a specific development goal and aligns with the planned schedule.

- **April 15, 2025:** Core backend system completed, including order processing and inventory logic.
- **May 11, 2025:** Persistent storage integration using MySQL, including secure data encryption mechanisms.
- **May 26, 2025:** Java GUI interface completed with full functionality for order creation, inventory tracking, and report generation.
- **June 4, 2025:** Comprehensive system testing finished, and final application delivery prepared for presentation.

3 Staffing

3.1 Requirements

Responsible for requirement gathering, analysis, and validation.

- Melike Aytac
- Sultan Esen Murat

3.2 Design

Responsible for system architecture, UML diagrams, and design principles.

- Yagmur Sabirli
- Beren Elcin Polat
- Melike Aytac

3.3 Iteration 1

Focus on the development of the core backend functionality and data model.

- Yagmur Sabirli
- Beren Elcin Polat
- Melike Aytac
- Sultan Esen Murat
-

3.4 Iteration 2

- Split into two sub-teams:

- **3.4.1 Interface Coding Group** Design and implement the graphical user interface.
- Sultan Esen Murat (Lead)
- Beren Elcin Polat
- Melike Aytac
- **3.4.2 Persistent Storage Group** Integrate and enhance database operations and implement data encryption.
- Yagmur Sabirli (Lead)
- Beren Elcin Polat

3.5 Iteration 3

Focus on integration testing, bug fixing, and visual/interface improvements.

- Melike Aytac (Lead)
- Yagmur Sabirli
- Beren Elcin Polat
- Sultan Esen Murat

3.6 Optional Iteration 4

Develop a web-based interface for additional accessibility.

- Entire team

3.7 Testers

Dedicated testers for each iteration to ensure software quality.

- Iteration 1: Melike Aytac
- Iteration 2: Sultan Esen Murat
- Iteration 3: Yagmur Sabirli

4 Process

We plan to use the following software process to manage the source code that we will create.

- **4.1 Version Control (Git + GitHub)** The project codebase is managed using Git for version control. GitHub is used as the remote repository platform, allowing for issue tracking, pull requests, and collaborative development. All contributors follow a branching strategy where each new feature or fix is developed in a separate branch and reviewed before being merged into the main branch.

- **4.2 Integrated Development Environments (IntelliJ IDEA / VS Code)** Development is carried out using IntelliJ IDEA and Visual Studio Code. These IDEs support Java development and allow integration with plugins for Git and Java GUI toolkits.
- **4.3 Build and Compile (OpenJDK)** The Java source code is compiled using OpenJDK 17. Builds are tested locally and deployed through GitHub-integrated workflows.
- **4.4 Team Communication (Microsoft Teams)** All communication and coordination are done using Microsoft Teams. Teams is also used for file sharing, organizing online meetings, and keeping development discussions transparent and accessible.
- **4.5 Issue Tracking and Task Management (Jira)** The Jira platform is used to log bugs, track tasks, assign responsibilities, and manage sprint progress. Tasks are broken down into user stories and distributed across iterations.

5 Coding Standard

5.1 Coding standard verification

Yağmur will perform periodic code checks to verify that the code meets the coding standard and report any standards violations to the programmer responsible for that violation.

5.2 Code reviews

As a team, we plan to meet for two hours once every other week to conduct code reviews on various modules.

5.3 Testing

5.3.1 Unit Testing

Unit testing will be performed by each programmer where appropriate. All test cases will be manually written to test the output of a particular unit. We may make use of a Junit test.

5.3.2 Integration Testing

Each Programmer will be responsible for thoroughly testing the subsystems when appropriate.

5.3.3 System Testing

The Testing group will be responsible for testing the system as a whole and verifying that the outputs meet the requirements.

5.4 Coding Standard applied for this project

5.4.1 Spelling and Capitalization

- Reserved words are all in lower case
- Declare constants and variables at the beginning of a file or a function definition
- Variables must be descriptive of their use. If the variable contains multiple terms, each new term be distinguished from the previous terms by using an uppercase letter or an underscore
- Functions must be descriptive of their use. If the variable contains multiple terms, each new term be distinguished from the previous terms by using an uppercase letter or an underscore

5.4.2 Punctuation and White Space

- Do not use preceding or trailing underscores in variable names.
- Curly braces {} are placed on their own lines to improve readability.

- Indentation is managed using tabs in IntelliJ IDEA (Java default convention).

5.4.3 Documentation and Style

- Each Java file starts with a structured header:

/**

- File: RestaurantManager.java
- Author: Melike Aytac
- Created on: March 1, 2025
- Purpose: Handles restaurant order processing and report generation.
-
- Modification Log:
- 04 Mar 2025 - Updated stock deduction algorithm.
- 15 Mar 2025 - Integrated login validation.

*/

- Each method is preceded by JavaDoc-style comments:

/**

- Calculates total order value.
- @param orderId The ID of the order
- @return Total value as a double

*/ public double calculateOrderTotal(int orderId) { // ... implementation

}

- All function definitions include parameter validation.

- `void` is explicitly used for methods with no return values.
- Inline comments are used to describe non-obvious logic.

Formatting

- Code is formatted to maximize readability.
- Tab-based indentation is used to ensure consistency with Java IDE conventions.

5.4.4 Comments

- Comments should be used to explain key program segments and segments whose purpose or design is not obvious to a reader of your program
- When the header file of a less commonly used class library is inserted in a program using "import ...", a comment should be used to explain its purpose
- Block comments should be used throughout the body of a program to describe the purpose of logical sections of code
- Meaningful identifier names should always be used. All variables and constants should be explained preferably by using self-explanatory names
- Programs should be formatted to maximize readability
- Indentation managed by tabs not spaces

6 Measurements

6.1 SLOC – Source Lines of Code

Each weekly build will include a SLOC count, helping us monitor development progress and identify potentially excessive complexity. This count will be automated in the build script for consistency.

6.2 Classes

The number of Java classes will be tracked across iterations. Our final design currently includes both backend (logic and data) and frontend (GUI) classes. At the end of development, we anticipate at least 14 core classes:

- **Backend Classes (9):**
  - RestaurantSystem
  - Menu
  - MenuItem
  - Ingredient
  - Inventory
  - Order
  - OrderItem
  - OrderManager
  - ReportGenerator
- **Frontend (GUI) Classes (5):**
  - LoginScreen
  - MainMenuScreen
  - OrderScreen
  - InventoryScreen
  - ReportScreen

This metric helps ensure that our object-oriented architecture is modular and follows best practices.

6.3 Methods/Class

For each method, we will take a metric about how many methods that class contains. From these data we will calculate total number of methods, average number of methods per class and track most and least complex classes (complexity being equated with method count for the sake of this measurement).

7 Risks

This project has risk associated with it. Items that may prevent the project from reaching the successful completion include:

7.1 Support personnel

We are relying on the support personnel to complete certain tasks in a timely fashion to allow the team to perform coding activities. If key personnel cannot perform the requested work in a timely fashion we may be pushed behind schedule.

7.2 Tools usage

The team has not used Jira before, the learning curve for this tool might provide steeper and longer than expected, pushing us off schedule.

7.3 Tool functionality

If any of the selected tools (e.g., IntelliJ IDEA, Java GUI libraries, GitHub CI/CD integration) fail to meet expected functionality — such as plugin compatibility issues, IDE crashes, or integration failures — we may need to reconfigure our development workflow or switch tools entirely. This could cause interruptions or delays in the development cycle.

8 Tools

The following tools were selected based on their compatibility with the Java development stack and the needs of the project:

- **IntelliJ IDEA / Visual Studio Code**: Primary IDEs used for writing and debugging Java code and designing the GUI.
- **Git**: Version control system used for managing changes to source code.
- **GitHub**: Used for hosting the project repository, managing issues, and team collaboration.
- **MySQL**: The database system used to store user data, orders, menu items, and inventory.
- **Java GUI Toolkit (Swing/JavaFX)**: Used for developing the desktop graphical user interface.
- **Microsoft Teams**: Utilized for group communication, file sharing, and meeting coordination.
- **Jira**: Agile project management tool used for issue tracking, sprint planning, and task assignment.

These tools were chosen for their availability, ease of use for students, and robust community support.

9 Support

9.1 Hardware Support

- Each team member has access to a personal computer capable of running Java-based development tools.
- Internet access is available for remote collaboration and access to version control and documentation.
-

9.2 Software Support

- All development software tools are open-source or free for student use.
- Java Development Kit (JDK 17) is used to compile and run the application.
- IntelliJ IDEA Community Edition and Visual Studio Code are used for programming.
- GitHub provides integrated version control and collaboration features.

9.3 Personnel Support

- **Prof. Dr. Mehmet Süleyman Ünlütürk**, the course instructor, provides guidance, evaluates deliverables, and facilitates requirement validation.
- Peer review within the development team ensures that implementation tasks are double-checked and that quality standards are maintained.
    - This support structure ensures that the team has the necessary infrastructure, tools, and mentorship to complete the project successfully.

10 Class Diagrams