Software Testing and Validation (ENSE 375)

Recipe Management System

From,

- Ramsha Naeem(200507574)
- Pratik Gadhiya (200518183)
- Kristina Langgard (200323529)

Agenda

- Introduction
- Problem Definition
- Design Requirements
- Solutions
- Testing and Demonstration
- Project Management
- Conclusion and Future Scope

Introduction

This project focuses on designing and developing a Recipe Management System, a Java-based desktop application that allows users to create, manage, and search their personal recipes.
We aim to simplify the process of storing and accessing culinary information with mindfulness of a user-friendly experience. In traditional alternatives, there is a lack of organization and limited accessible tools for individuals who enjoy cooking and want to manage their recipes. Existing solutions are often either overly complicated or too limited in functionality. We wish to modernize the experience to be convenient and approachable.

Rationale:

- The need for this project became clear after observing the difficulties faced by home cooks and hobbyists in managing recipes effectively.
- Paper-based systems are prone to damage and disorganization, while online platforms often come with distractions or require internet access.
- Our proposed Recipe Management System addresses these issues by offering a simple, efficient, and accessible way to store and retrieve recipes, without unnecessary complications.

Problem Definition

Issues with Traditional Methods:

1. *Lack of Centralized Storage*
2. *Complexity of Existing Platforms*
3. *Limited Customization Options*

Benefits of a Digital Solution:

1. *Streamlined Organization*
2. *Intuitive User Experience*
3. *Offline Functionality*

Design Requirements - Functions

- **Add Recipes:** Allows users to input recipe details (title, ingredients, steps, categories, and author) and save them to the local database.
- **View Recipes:** Displays a list of all stored recipes with options to view details like ingredients, steps, and categories.
- **Delete Recipes:** Enables users to remove unwanted recipes from the local database permanently.
- **Search/Filter Recipes by Category:** Provides search functionality and filtering based on recipe categories (e.g., breakfast, dessert) for quick access.
- **Export & Import Recipes (CSV, PDF):** Allows users to export recipes as CSV or PDF files and import recipes from compatible files for easy sharing and backup.
- **Optional Login for GUI:** Provides a simple login screen in the GUI mode to secure access and maintain personalized recipe management.

Design Requirements - Objectives

- **User-Friendly:** Create a simple and intuitive interface (Console & GUI) for effortless recipe management.
- **Offline Functionality:** Ensure the app works completely offline without dependency on cloud services.
- **Lightweight & Efficient:** Keep the software resource-efficient with minimal dependencies and fast performance.
- **Data Security & Privacy:** Protect user data by storing recipes locally and allowing controlled export/import.
- **Cross-Platform Support:** Make the solution portable and compatible across different operating systems.
- **Reliable:** The system should function correctly and consistently under normal usage conditions, maintaining data integrity and accuracy.
- **Maintainable:** The system should be designed with clean and modular code, facilitating easy maintenance and future enhancements.

Design Requirements - Constraints

- **Local Storage Only:** All data must be stored in a single JSON file, with no external cloud database.
- **Limited Budget:** Development must be cost-free, using open-source libraries and tools.
- **Time-Bound Development:** The project must be completed within the allocated academic schedule.
- **No Internet Dependency:** All features (add, view, delete, export/import) must work fully offline.
- **Minimal Hardware Requirements:** The system must run on standard computers without high-end specifications.

Solutions – 1 & 2

Solution 1: Console-Only Application
✅ **Simple to Implement**
Very lightweight with minimal dependencies.
Runs on any machine with Java installed.
❌ **No Graphical Interface**
- Not user-friendly for non-technical users.
- Requires command-line knowledge.

❌ **Lacks Advanced Features**
- No recipe import/export functionality.
- No support for organizing or filtering recipes.
- Limited interactivity and user experience.

Solution 2: Cloud-Based Web App
✅ **Access from Anywhere**
- Can be used from any device with a browser and internet connection.
- Ideal for remote collaboration.

❌ **Internet Dependency**
- Cannot be used offline.
- Breaks if the server goes down.

❌ **Data Privacy Concerns**
- Requires backend to store data on a server.
- Users' recipe data is exposed to network vulnerabilities.

Transition Statement:
While both options had their strengths, they also came with significant limitations. We needed a solution that maintained accessibility and usability while addressing performance, security, and offline capabilities.

Solutions – Final Solution

Hybrid GUI + Console Application

- Combines both GUI (JavaFX) and ConsoleUI for flexible user interaction.
- Recipes stored locally in recipes.json (no internet required).
- GUI includes an optional login feature for enhanced usability and privacy

Features include:

- Add, view, delete recipes via GUI or Console.
- Search and filter recipes by category or keywords.
- Export recipes to CSV and PDF formats and Import recipes from compatible files.
- No cloud dependency works entirely offline.

Comparison Table:

Feature	Solution 1 (Console)	Solution 2 (Cloud)	**Solution 3 (Final)**
Offline Access	✅ Yes	❌ No	✅ Yes
GUI	❌ No	✅ Yes	✅ Yes
Data Privacy	✅ Local JSON	❌ Cloud storage	✅ Local JSON
Export/Import	❌ No	❌ Limited	✅ Yes
Testing	❌ Limited	❌ Limited	✅ Full Test Suite

Testing and Demonstration

Test Requirements for Each Testing Technique:

1. **Boundary Value Testing:** Checked limits (e.g., max ingredients, empty titles).
2. **Decision Table Testing:** Verified combinations of valid/invalid inputs.
3. **State Transition Testing:** Ensured proper navigation between menus and states.
4. **Unit Testing:** Validated individual methods (add, delete, search, export).

Testing and Demonstration

Test Cases to Satisfy the Test Requirements:

1. **Boundary Value Testing**:

- Objective: Ensure that all user inputs fall within expected limits.
- Example: Title field length (min: 1 character, max: 100 characters), ingredient/step limits.
- Result: Application correctly accepts valid inputs and rejects invalid ones

2. **Decision Tables Testing**:

1. Objective: Validate rule-based operations like saving, deleting, or searching recipes based on user conditions.
2. Example Table:

Condition	Input Valid	Recipe Exists	Action
Add Recipe	Yes	No	Save recipe
Add Recipe	No	-	Show error
Delete Recipe	Yes	Yes	Delete
Delete Recipe	Yes	No	Show message

3. Result: Application behavior matches the expected actions for each condition.

Testing and Demonstration

Test Cases to Satisfy the Test Requirements:

3. State Transition Testing:

1. Objective: Verify correct navigation and state change between UI states.
2. Example:

State 1: Main Menu → Add Recipe → Save → Back to Main Menu
State 2: Main Menu → View Recipes → Back
Result: All UI states change appropriately and no broken transitions found

4 .Unit Testing:

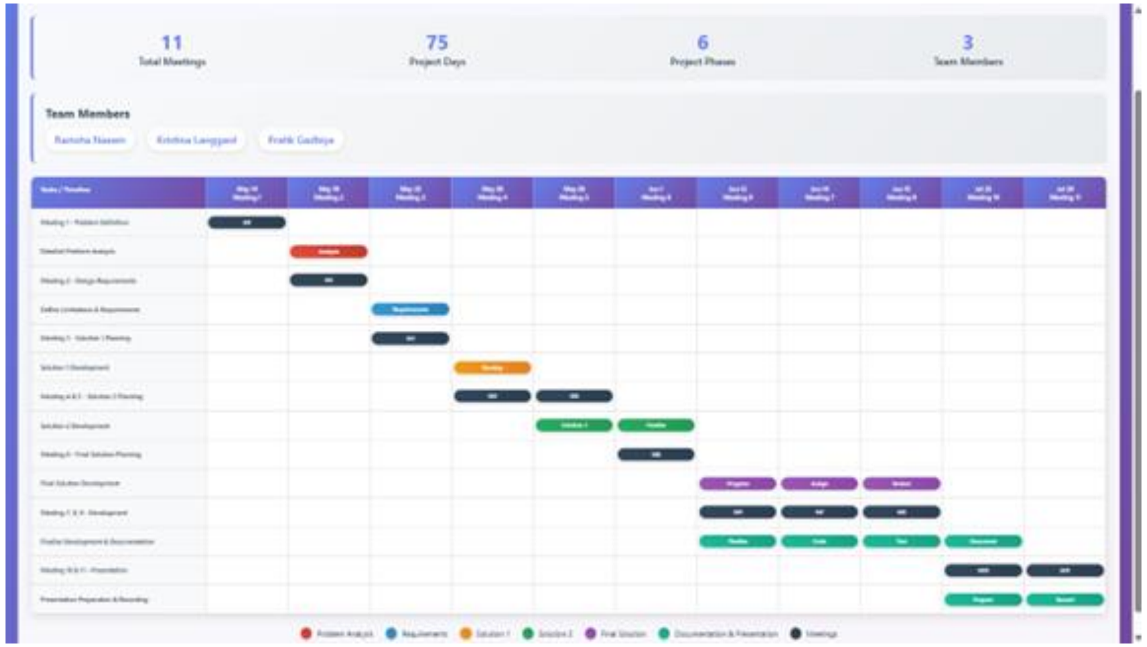
1. Objective: Validate each method’s functionality in isolation.
2. Tools Used: Junit
3. Example: save(Recipe), loadAll(), delete(String title)
4. Result: All unit tests passed successfully with 100% method coverage.

Testing and Demonstration

DEMO

Project Management

- Gantt Chart



- We followed an incremental development process, starting with a console-based prototype (Solution 1), then adding GUI features (Solution 2), and finally combining both into a stable hybrid solution (Solution 3).
- Used version control (Git/GitHub) for collaboration and tracking changes
- .Regular test-driven development ensured code quality and minimized regressions.

Conclusion and Future Work

Conclusion:

- Solution 3 successfully **meets all functional requirements**, offering a **user-friendly hybrid interface (Console + GUI)**.
- **Comprehensive testing** (Boundary Value, Decision Table, State Transition, Unit Tests) improved reliability and robustness.
- Offline operation, simple design, and export/import features make the application **practical, secure, and resource-efficient**.
- The system is modular and **future-proof**, allowing future enhancements like multi-user accounts, advanced search, and cloud sync.
- We achieved all project objectives **within the set timeline and constraints**.

Future Work:

- Add image attachments to recipes.
- Cloud sync (optional).
- Advanced search (by ingredients).

Thank you!