ENSE 375 – Software Testing and Validation

Kitchen App Unit Converter

Alvin John Tolentino (200123456)

Quing Rosete (200449566)

Taiwo Akinwale (200430975)

**Table of Contents**

1. Introduction
2. Design Problem
   * Problem Definition
   * Design Requirements
     + Functions
     + Objectives
     + Constraints
3. Solution
   * Solution 1
   * Solution 2
   * Final Solution
     + Components
     + Features
     + Environmental, Societal, Safety, and Economic Considerations
4. Test Cases and Results
   * Path Testing
   * Data Flow Testing
   * Logic Coverage Testing
   * Testing Methodologies Used
     + Boundary Value Testing
     + Equivalence Class Testing
     + Decision Tables Testing
     + Use Case Testing
   * Additional Tests
     + Recipe Tests
     + Timer Tests
   * Limitations
5. Team Work
   * Meeting 1
   * Meeting 2
   * Meeting 3
   * Meeting 4
   * Meeting 5
   * Meeting 6
   * Meeting 7
   * Meeting 8
   * Meeting 9
6. Project Management
7. Conclusion and Future Work
8. References
9. Appendix

**Introduction**

The Kitchen App aims to assist users in managing their kitchen tasks efficiently by providing functionalities like recipe management, a unit converter, and a timer. This documentation focuses on the unit converter component, which allows users to convert various units of measurement commonly used in the kitchen. The unit converter is designed to be user-friendly, accurate, and efficient, facilitating seamless conversions between different units such as cups, tablespoons, teaspoons, kilograms, grams, milliliters, and more.

**Design Problem**

**Problem Definition**

The primary problem addressed by the unit converter is the difficulty faced by users in converting between various kitchen measurement units accurately and quickly. Inaccurate conversions can lead to incorrect ingredient proportions, affecting the quality of the recipes prepared.

**Design Requirements**

**Functions**

* Convert values between different units of measurement.
* Allow users to select the unit they are converting from and the unit they are converting to.
* Display the conversion result clearly.

**Objectives**

* Ensure accuracy in unit conversions.
* Provide a simple and intuitive user interface.
* Allow quick and easy unit selection and conversion.

**Constraints**

* The app must perform conversions without requiring an internet connection.
* The user interface must be responsive and work on various device sizes.

**Solution**

**Solution 1**

The first solution involved using a basic user interface with text input for the value and two dropdown menus for unit selection. However, this solution had usability issues, including resetting selections after each conversion.

**Solution 2**

The second solution improved on the first by maintaining user selections and providing better error handling. However, it still lacked some usability enhancements.

**Final Solution**

The final solution provides a refined user interface with persistent selections and comprehensive error handling. It includes:

**Components**

* Eclipse
* A map for storing conversion rates

**Features**

* Accurate unit conversions
* Persistent unit selections
* Clear error messages

**Environmental, Societal, Safety, and Economic Considerations**

* The app is digital, reducing the need for physical conversion charts.
* Accurate conversions help in reducing food waste.
* It is safe to use and does not involve any harmful processes.
* The app is economical as it is free to download and use.

**Test Cases and Results**

**Path Testing**

Path testing ensures that all possible paths through the code are tested. This was used to verify the navigation within the app, ensuring all user paths from entering values to seeing conversion results work correctly.

**Data Flow Testing**

Data flow testing checks the flow of data through the application to ensure it is processed correctly. This was used in testing the conversion logic, verifying that inputs are correctly processed and outputs are accurately produced.

**Logic Coverage Testing**

Logic coverage testing ensures that all logical conditions and branches in the code are tested. This was employed in validating the conditional logic for error handling and unit selection.

**Testing Methodologies Used**

* Boundary Value Testing: Used to ensure the edge cases of input values (e.g., minimum and maximum values) are handled correctly. For instance, testing the conversion logic with values like 0 and large numbers.
* Equivalence Class Testing: Applied by grouping similar inputs that should produce the same output to reduce the number of test cases. This was useful in testing various units that share similar conversion logic.
* Decision Tables Testing: Used to represent combinations of inputs and their corresponding outputs to ensure all possible scenarios are tested. This method was employed in testing the user interface for various input combinations and conversion results.
* Use Case Testing: Testing based on user scenarios to ensure the application behaves as expected in real-world usage. This included testing common user actions such as converting common units like cups to tablespoons and verifying the results.

**Additional Tests**

Recipe Tests

These tests ensure that recipes are loaded correctly and that ingredients and steps are displayed as expected. Path testing and use case testing were particularly relevant here to verify the flow from selecting a recipe to viewing its details.

Timer Tests

These tests verify the functionality of the timer, including adding, starting, stopping, resetting, and deleting timers. Additional tests included ensuring negative time values were not allowed. Boundary value testing and equivalence class testing were used to validate these functionalities.

Limitations

* The app does not support all possible units.
* Limited to predefined conversion rates.

**Team Work**

Meeting 1

Time: May 28, 2024, 5:00 PM - 6:00 PM  
Agenda: Discussion of Project Vision and Initial Task Allocation

* Alvin John Tolentino: Initial research on unit conversion logic
* Quing Rosete: Preliminary UI design concepts
* Taiwo Akinwale: Backend architecture planning

Meeting 2

Time: June 1, 2024, 5:00 PM - 6:00 PM  
Agenda: Distribution of Project Tasks

* Alvin John Tolentino: Task 1 - Implementing basic unit conversion logic
* Quing Rosete: Task 2 - Designing initial UI mockups
* Taiwo Akinwale: Task 3 - Setting up the backend infrastructure

Meeting 3

Time: June 8, 2024, 5:00 PM - 6:00 PM  
Agenda: Review of Individual Progress

* Alvin John Tolentino: Task 1 (80%) -> Task 1, Task 5 - Refining conversion logic and error handling
* Quing Rosete: Task 2 (50%) -> Task 2 - Completing UI design
* Taiwo Akinwale: Task 3 (100%) -> Task 6 - Integrating backend with front end

Meeting 4

Time: June 15, 2024, 5:00 PM - 6:00 PM  
Agenda: Review Progress and Discuss Issues

* Alvin John Tolentino: Task 1, Task 5 (100%) -> Task 7 - Implementing persistent selections
* Quing Rosete: Task 2 (90%) -> Task 8 - UI improvements based on feedback
* Taiwo Akinwale: Task 6 (100%) -> Task 9 - Backend optimizations

Meeting 5

Time: June 22, 2024, 5:00 PM - 6:00 PM  
Agenda: Final Review and Testing

* Alvin John Tolentino: Task 7 (100%) -> Review - Final code review
* Quing Rosete: Task 8 (100%) -> Testing - User interface testing
* Taiwo Akinwale: Task 9 (100%) -> Documentation - Completing project documentation

Meeting 6

Time: June 29, 2024, 5:00 PM - 6:00 PM  
Agenda: Discuss Java Learning Resources

* All team members: Learn extra Java components

Meeting 7

Time: July 6, 2024, 5:00 PM - 6:00 PM  
Agenda: Review of Learning Progress

* All team members: Discuss what we have learned about Java

Meeting 8

Time: July 13, 2024, 5:00 PM - 6:00 PM  
Agenda: Code Review and Bug Fixes

* Alvin John Tolentino: Fix unit conversion logic bugs
* Quing Rosete: Refine UI design
* Taiwo Akinwale: Ensure backend stability

Meeting 9

Time: July 20, 2024, 5:00 PM - 6:00 PM  
Agenda: Final Adjustments and Documentation

* Alvin John Tolentino: Final testing and validation
* Quing Rosete: UI polishing
* Taiwo Akinwale: Compile final documentation

**Project Management**

**Gantt Chart Mapping**

* Initial Research: May 28 - June 1
* UI Design: June 1 - June 8
* Backend Development: June 1 - June 15
* Integration and Testing: June 15 - July 6
* Extra Java Learning: June 29 - July 6
* Code Review and Bug Fixes: July 6 - July 13
* Final Adjustments and Documentation: July 13 - July 20

**Conclusion and Future Work**

The unit converter component of the Kitchen App successfully achieved its design functions and objectives, ensuring accurate and user-friendly conversions. Future improvements could include adding more units and enhancing the user interface further. Additionally, implementing an account creation system where users can make groups to share recipes would be beneficial for teaching classes or simply sharing recipes with others.

**References**

IEEE Citation Format: Used for all references in this document.

**Appendix**

**Gantt Chart**

|  |  |  |
| --- | --- | --- |
| **Task** | **Start date** | **End Date** |
| **Initial Research** | **May 28** | **June 1** |
| **UI Design** | **June 1** | **June 8** |
| **Backen Development** | **June 1** | **June 15** |
| **Integration and Testing** | **June 15** | **July 6** |
| **Extra Java/Flutter Learning** | **June 29** | **July 6** |
| **Code Review and Bug Fixes** | **July 6** | **July 13** |
| **Final Adjustments and Documentation** | **July 13** | **July 20** |