
MASTER TEST DOCUMENT

for

Personal Dietary Application

Version 0.1

by Craig Boucher
Md Tanveer Alamgir
Fan Zou
Osman Momoh
Xin Ma

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 4 |
| 1.1 | Purpose | 4 |
| 1.2 | Scope | 4 |
| 1.3 | Document Terminology and Acronyms | 5 |
| 1.4 | References | 5 |
| 2 | Evaluation Mission Test Motivation | 6 |
| 2.1 | Background | 6 |
| 2.2 | Evaluation Mission | 6 |
| 2.3 | Test Motivators | 7 |
| 3 | Outline of Planned Tests | 8 |
| 3.1 | Unit Testing | 8 |
| 3.2 | Integration and Module Testing | 8 |
| 3.3 | Functional Testing | 9 |
| 3.4 | User Interface Testing | 9 |
| 4 | Test Approach | 10 |
| 4.1 | Unit Testing | 10 |
| 4.1.1 | Methods... | 10 |
| 4.2 | Integration and Module Testing | 10 |
| 4.2.1 | MVC... | 10 |
| 4.3 | Functional Testing | 10 |
| 4.4 | User Interface Testing | 13 |
| 4.4.1 | UI, buttons, etc... | 13 |

1 Introduction

This master testing document describes the strategies and tools we (Team 4) used to test the Personal Dietary Manager application. This testing document serves as the deliverable document for iteration 3 of the class project in COMP 5541 at Concordia University. Iteration 3 is the final iteration of Personal Dietary Manager, and implements database functionality using MySQL.

Associated with this test plan are documents delivered in iteration 1 and iteration 2 of this project. These are, respectively, the Software Requirements Specifications (SRS) and Design Documents (DD). This test plan will refer to these other documents for the purpose of traceability.

1.1 Purpose

The purpose of this test plan is to verify the quality and correctness of the Personal Dietary Manager application. Correctness is the state in which the application satisfies the SRS and delivers a product that can accurately track a dietary regimen. Quality is the state in which the application delivers a user-friendly experience that is free of bugs, runs fast, and runs smoothly.

As no software project is perfect, any outstanding bugs and issues will also be discussed.

1.2 Scope

The scope of this test document are the tools and strategies that are used for testing. These are:

- Unit testing
- Integration and Module Testing
- Functional Testing
- User Interface Testing

Tests that are outside of the scope of this document are load, configuration and installation, business analysis, network security, and stress testing.

1.3 Document Terminology and Acronyms

| Abbreviation | Term |
|--------------|---|
| MVC | Model View Controller |
| GUI | Graphical User Interface |
| UML | Unified Modeling Language |
| PDA | Personal Dietary Application |
| GRASP | General Responsibility Assignment Software Patterns |

1.4 References

- Dr. Nora Houari, 2020, “COMP 5541 Course Notes” including:
 - Introduction to testing
 - Tutorial: Testing
 - Project Information
 - Montrealopoly Master Test Plan
- Pressman, R.S. (2009). Software Engineering: A Practitioner’s Approach, 7Th Edition.
- ISO/IEC/IEEE International Standard - Software and systems engineering – Software testing –Part 3: Test documentation,” in ISO/IEC/IEEE 29119-3:2013(E) , vol., no., pp.1-138, 1 Sept. 2013

2 Evaluation Mission Test Motivation

The purpose of this test document is to ensure the personal dietary desktop application meets the design requirements mandated by the course objectives outlined in the project specification document and system requirements document. This document will be a tool to outline test cases and keep a quality assurance guideline for the software development team. Lastly, the 'team 4' group will deliver an exceptional piece of quality software.

2.1 Background

The graduate diploma course COMP 5541 mandated a personal dietary application be developed as part of a group project for the semester. The third increment of this project involved implementing a database for use with the desktop application. A remote database server offered by Microsoft, Azure, is utilized. The Java programming language and GUI library JavaFX. A comprehensive testing environment has been created to make sure the software is completed without any major issues or bugs.

This test document outlines several testing methods and processes that were used to verify the software is without serious bugs. The previous document included, the design document, had the software architecture and design pattern specifications. The initial document, the requirements document, contained the class diagrams, use case diagrams, and the purpose and demands the software is intended to fulfill.

2.2 Evaluation Mission

The principal goals of this document are evaluated as the following:

- Making sure that the design document quality is adhered to.
- Guaranteeing the software requirement document specifications are completed.
- That any major software bugs are not found in the application.

The test plan document will be used as a tool to make sure these missions are accomplished.

2.3 Test Motivators

The areas explored in this testing document are the following:

Unit Testing: A set number of methods are tested to verify their inputs and outputs meet the specifications. White and black box testing methods are used.

Integration and Module Testing: The classes and objects are tested together to see if behavior remains consistent.

Functional Testing: Makes sure that use cases objectives are accomplished.

User Interface Testing: The GUI that the user interacts with is checked for malfunctions.

3 Outline of Planned Tests

This section provides an overview of the four types of tests that we will do: unit testing, integration and module testing, functional testing, and user interface testing. A brief description of each is described.

We frequently distinguish our testing strategies between white- and black-box testing. White-box testing tests the internal structures of an application (e.g. source code, classes, methods). Conversely, black-box testing tests the external functions that the software is supposed to perform, and does not necessarily examine the underlying source code.

3.1 Unit Testing

The primary focus of unit testing is performing isolated tests on specific methods in the system. We will be using white box testing techniques to iterate through the paths the methods take to execute their instructions. The following list of methods is not every method present in the software application however they are the most crucial. For a complete list of the methods please see the design document. Here are the methods included in this document which are tested:

- Method createDining
- Method deleteDining
- Method findDiningByName
- Method createMeal
- Method createRetailer

3.2 Integration and Module Testing

The integration and module testing will verify that the classes and methods work together as a larger system. There is only one main window for the personal dietary application. Both bottom up and top down testing are used when describing the test cases and their results for the functionality available to the user in the main window. From the main window, every piece of the model view controller (MVC) model is coordinated to deliver the appropriate software functionality.

3.3 Functional Testing

Functional testing is a form a black-box testing in which the application use cases are tested. These use cases can be found both in the SRS and the Project Infromation class notes document. In other words, the functional requirements of the project are tested for correctness.

3.4 User Interface Testing

User interface (UI) testing involves testing the various parts of the GUI for correctness and quality. These GUI parts include the text fields, comboboxes, observable lists, list views, table views, buttons, etc. UI testing is a form of black-box testing.

4 Test Approach

4.1 Unit Testing

4.1.1 Methods...

4.2 Integration and Module Testing

4.2.1 MVC...

4.3 Functional Testing

In this section we outline the tests done for the functional requirements (a.k.a use cases). A list of these use cases can be found in the SRS and Project Information class notes.

4.3.1

| | |
|-----------------------|---|
| Use Case | View a list of all foods in diet |
| Description | The user is able to view a list of foods (both consumed and not consumed) |
| Preconditions | The user has entered food items into list |
| Postconditions | A list of foods is displayed on the center portion of the application |
| Comments | Test successful |

4.3.2

| | |
|-----------------------|---|
| Use Case | Add a new food item |
| Description | A new food item is added to the list of foods |
| Preconditions | The user has inputted all text fields with appropriate data |
| Postconditions | A new food item appears at the end of the displayed food list |
| Comments | Test successful |

4.3.3

| | |
|-----------------------|--|
| Use Case | Mark a food group as eaten or not eaten |
| Description | A food group will be marked as eaten if a food item is consumed of that group. It will be marked not eaten if there are no consumed food items in that group |
| Preconditions | The presence, or not, of food items in the food list |
| Postconditions | The appropriate food group box will light up at the bottom of the application |
| Comments | Test successful |

4.3.4

| | |
|-----------------------|--|
| Use Case | Mark a food group as eaten or not eaten |
| Description | A food group will be marked as eaten if a food item is consumed of that group. It will be marked not eaten if there are no consumed food items in that group |
| Preconditions | The presence, or not, of food items in the food list |
| Postconditions | The appropriate food group box will light up at the bottom of the application |
| Comments | Test successful |

4.3.5

| | |
|-----------------------|--|
| Use Case | Remove a food item |
| Description | A food item in the food list is removed |
| Preconditions | One or more food items are in the food list |
| Postconditions | The selected food item is removed from the food list |
| Comments | Test successful |

4.3.6

| | |
|-----------------------|---|
| Use Case | Sort the food list based on a food attribute (e.g. serving, time, food group) |
| Description | The list of foods is sorted based on a given food attribute |
| Preconditions | Two or more food items are in the food list |
| Postconditions | The foods in the food list are sorted in ascending or descending order based on the attribute |
| Comments | Test successful |

4.3.7

| | |
|-----------------------|--|
| Use Case | Set a food item as consumed or not consumed |
| Description | The user is able to select in the list of foods which ones are consumed or not consumed |
| Preconditions | One or more food items are in the food list |
| Postconditions | A food item is marked as consumed or not consumed. The total nutrients in the consumed food list is appropriately updated. |
| Comments | Test sucessful |

4.3.8

| | |
|-----------------------|--|
| Use Case | Choose a food item as either indining or outdining |
| Description | The user is able to select if the food item has either an indining or outdining food attribute |
| Preconditions | The user has entered corrected all values in the text areas |
| Postconditions | A food item is entered into the food list with either an indining or outdining food attribute |
| Comments | Test sucessful |

4.3.9

| | |
|-----------------------|--|
| Use Case | View only consumed food items, or view all food items |
| Description | The user is able to view either consumed food items, or a list of all food items (consumed and not consumed) |
| Preconditions | Two or more food items are in the food list |
| Postconditions | Either all food items will be displayed, or only consumed food items |
| Comments | Test sucessful |

4.3.10

| | |
|-----------------------|--|
| Use Case | The food list will persist after application closes |
| Description | The food list is automatically saved in a database after a food item is added or deleted. The food list is restored upon opening the application |
| Preconditions | The user has previously added or removed food items |
| Postconditions | The food list which existed before closing the application is restored upon opening the application. |
| Comments | Test successful |

4.4 User Interface Testing

4.4.1 UI, buttons, etc...