**Inventory Management Documentation**

*Cover Page*

Project Name: Grocery and Finance Manager for Home Kitchens

Team Members:

Nathan Moore

Cody Bracewell

Johnathan Lewis

Joseph Lambo

This is a course requirement for CS321- Object Oriented Java at the University of Alabama in Huntsville

Date of the release: April 20th, 2021

*Intellectual property*

Copyright 2021

Nathan Moore

Cody Bracewell

Johnathan Lewis

Joseph Lambo

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3, (3 November 2008) or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license can be found at [https://www.gnu.org/licenses/fdl.html.](%20https://www.gnu.org/licenses/fdl.html.)

*Table of Contents*

Each section should be included along with the page number. Example:

Table of Figures ……………………………………………. i

1.0 Project Description ………………………………………1

1.1 TBD ……………………………………………...1

2.0 Project Management ……………………………………..2

*Table of Figures*

Provide a **list** of the figures in your document. These page numbering use the i, ii, iii format. Prior to the document pages.

***Followed by an INTENTIONALLY LEFT BLANK page before PAGE 1 of the Document.***

*1.0 Project description*

This program will be used to create weekly expense reports/grocery lists for at home chefs in order to keep up with incoming and outgoing food product expenses. These reports will be displayed as graphs as well as lists of flagged items either being under or overused by the user. Alongside generating expense reports program will allow user to check if available inventory exists to build a recipe, if not a grocery list will be generated for the chef with a prices based off of most recent bulk cost price.

*2.0 Project management*

This is the description of the work that went into the project.

*History*

This section will provide a description of what you have done during the semester. You should provide a brief description of the previous submissions and what improvements were made over the semester.

*Personnel*

Nathan Moore

Cody Bracewell

Johnathan Lewis

Joseph Lambo

*Effort*

Indicate approximately how much time the team as a whole as spent on the project. Also indicate the number of meeting that occurred during the semester. A time line would be helpful.

*3.0 Use Cases*

1. Login
2. Input Inventory
3. Check Recipe Readiness
4. Save Current Progress
5. Request for Finance Report
6. Generate Grocery List

*4.0 Requirements*

The description of the requirements should be based on your project description and the use cases.

*Functional requirements*

If you use special words with specific meanings, define them first. This should be from the user or client point of view. Use the use cases as a guide. These will be listed in summary form in the *Summary* section below.

Describe:

* R1 – Prompt Login – ask for numeric pin
* R2 -- Import all item amounts from file to objects in program
* R3 – GUI action update inventory – used to compile grocery lists if not enough of item available, add/remove items
* R4 – GUI inventory amount type – drop boxes beside item name that allow selection of: oz, lb, tablespoon, etc. -- will need some math behind this
* R5 -- Import new item amounts to recipes and calculate recipe readiness – background process.
* R6 – GUI action display report/grocery list – user will use a “generate shopping list button”.

*User interface requirements (diagrams/sketches of the envisioned UI are very useful here)*

Describe:

Dry Ingredients – any items that aren’t liquid ingredients

Wet Ingredients – liquid ingredients

* Program Sketch
* Input Elements
  + - * Item name, bulk price, bulk weight
      * Search button
      * Add/Delete selection
      * Choose to generate report(s)
* Output Elements
  + - Financial Reports
    - Grocery List
    - Updates to Ingredient File

If invalid input from the user is detected, program should alert user with a pop up and highlight the field in which the invalid entry was made – this can be a nonexistent item name in the search field, an invalid cost (cost < $0.00), or invalid amount input (weight < 0.00) if user inputs an existing item into add should show that item already exists and prompt user to update existing item.

*Future modification and extensions (Optional in Requirement Phase, Required in Final Project)*

Describe how the program could be extended in the near future. These are the things that will not appear in your implementation, but you will take these things into consideration in your design. You may use additional scenarios, window sketches, etc., to illustrate the future extension.

*Summary*

This should be a list of statements that indicate what the software shall do. These should have a clear numbering and be clearly linked to the scenarios. For example, “The software shall accept a user selection of direction and produce the appropriate scene as presented in Use Case 1.4”. You might consider a table since you will need to show how the requirements are linked to the scenarios. Consider this to be the general contract for your system. You will satisfy the contract if you satisfy each requirement. This does not mean that you have satisfied the contract well or that the contract was interesting. These last two items will also be used in evaluating your performance

*Associated tests (Optional in Requirements Phase, Required in Final Project Document)*

In this section describe the test you will use to determine whether the requirement is satisfied. There will probably be a bit of text but there should also be a clear summary (perhaps a table) that indicates the requirement and the test.

*5.0 Design*

This section describes the static and dynamic elements of your system and the relation of the model to the user interface (views and controls). Descriptions of the final CRC cards may be used here. The actual CRC cards should be placed in an appendix. At least some of the design should be documented with appropriate UML class diagrams. Focus on areas that have classes and interactions specific to your project. In other words, no not provide diagrams for common data structures or utilities.

Since the MVC architecture is being used in the project, describe your data model, the views, and the controls. Indicate the communication between the parts. Once again this should not have a great amount of detail. This section is about design and not implementation.

*Model design*

What is in the model and what are the ways in which it can change?

*View design*

What are the views of the data and how are they generated?

*Control design*

What are the elements by which the user can request a change to the mode?

*6.0 Implementation*

This section provides descriptions of how the design is implemented. Do not provide excessive detail. The details should be in the javadocs and other code comments. You should indicate which, if any, design patterns you have implemented.

*Packages and classes*

Describe the packages, classes, components, interfaces, and the relationships that implement the design. Make sure you explicitly indicate how a package, class or group of classes relates to or satisfies a design specification in the previous section. Use text here, but make clear references to the javadocs. For each public class indicate in what package it is contained. For each public class clearly indicate if it is abstract or concrete, whether it extends another class, and whether it implements an interface. For each public class describe the API for that class.

*Utility classes and packages*

Describe the packages and classes that are used to satisfy design specifications but are not in the actual design. This would be the placed to indicate the way in which you satisfied the four component requirements that are not explicitly examined above. You can consider the packages and classes that deal with the views and controls as satisfying the GUI component.

*Tested functionality*

Describe functionalities and the outcome of the tests, i.e., list functionalities that have been tested and present the results of the tests. Note that all the use cases/functionality listed in the requirements document should be tested.

*Untested functionality*

Describe the functionalities that have not been tested, and include an explanation of why the tests have not been completed.

*7.0 Discussion*

Provide a discussion of any items that you think are important. This should include anything that you want to say about trade-offs that you have made, particular implemented features that you want to highlight, and any lessons learned.