**“the burger shot” RESTAURANT MANAGEMENT PLAN**

**Contents**

[1](#_heading=h.37m2jsg) OVERVIEW 4

[1.1](#_heading=h.1mrcu09) Purpose 4

[1.2](#_heading=h.46r0co2) Scope 4

[1.3](#_heading=h.111kx3o) Document Organization 4

[1.4](#_heading=h.206ipza) Changes and Revisions 5

[1.5](#_heading=h.4k668n3) Issues 5

[2](#_heading=h.2zbgiuw) ROLES AND RESPONSIBILITIES 5

[2.1](#_heading=h.3rdcrjn) Organization Overview 5

[2.1.1](#_heading=h.26in1rg) Role Breakdown 5

[3](#_heading=h.1egqt2p) REQUIREMENTS PROCESSES 5

[3.1](#_heading=h.3ygebqi) Overview 5

[3.1.1](#_heading=h.1ksv4uv) Phase One - Design 6

3.1.[2](#_heading=h.1ksv4uv) Phase Two - Flowchart 6

3.1.3 Phase Three - Pseudo Code 6

3.1.4 Phase Four - Coding 7

[4](#_heading=h.2dlolyb) TOOLS 7

[5](#_heading=h.z337ya) REQUIREMENTS DOCUMENTATION AND ORGANIZATION 7

[5.1](#_heading=h.sqyw64) Requirements Documentation 7

[5.1.1](#_heading=h.3cqmetx) Breakdown Structures 7

[5.1.2](#_heading=h.1rvwp1q) Associated Information 8

[5.2](#_heading=h.4bvk7pj) Organization 8

[6](#_heading=h.3q5sasy) MEASURES 9

[A.](#_heading=h.25b2l0r) Definitions, Acronyms, Abbreviations 10

B[.](#_heading=h.34g0dwd) Requirements Evaluation Checklists 11

C[.](#_heading=h.43ky6rz) Quality Standards 14

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **DATE** | **Version** | **Description** | **Author** |
| 10/03/2019 | 1.0 | First version of management document. | Arthur Poggenburg, Kenneth Chuson, Nathaniel Duya, Jessica Win |
|  |  |  |  |
|  |  |  |  |

# OVERVIEW

The project “The Burger Shot” Restaurant, is a restaurant that focuses on selling burgers and drinks. The software will provide advantages for both the customer and employees. The software will help organize the restaurant system to be as optimized as possible.

## PURPOSE

The purpose of this project is to have a restaurant software system that allows customers and employees to communicate easily, just using C++ software. This project helps to improve restaurant management, keeps track of customer orders and improves the working system for employees.

## SCOPE

* **What must be done**

The team must understand business ideas, software requirements, collecting and analyze data for the restaurant system. The team should be able to construct flowcharts in order to streamline the code. If necessary, the team also may have to do research in object orientation and data structures for C++.

* **How it shall be done**

To have a reliability and to maintain our code for the project requirements.

* **Who will perform various activities**

Not only will the leader perform various activities, but the team should keep track of the project whether there are issues, changes, repositories, and etc.

* **When they must be performed**

The team should be able to complete their parts earlier on so that there is time to correct bugs and errors.

* **What level of quality must be achieved**

In order to achieve peak quality levels, the team should be familiar with the project requirements, have good communication skills, critical thinking, experience in C++, and come up with ideas for the project.

## DOCUMENT ORGANIZATION

Organizing documents is important that the team should able to know what they are doing and to tackle the requirements for the project such as designing the project from scratch.

## CHANGES AND REVISIONS

Contact the team immediately if the project got ruined or if there are major changes. Have a clear explanation to the changes and revisions for the project.

## ISSUES

Issues may arise with a lack of communication or new iterations of the code without any patch logs. Communication will be very important in order to solve issues.

# ROLES AND RESPONSIBILITIES

## ORGANIZATION OVERVIEW

|  |  |  |
| --- | --- | --- |
| Name | Role | Purpose |
| Kenneth Chuson | Leader | Management System |
| Nathaniel Duya | Code Support | Order Application System |
| Jessica Win | Bug Fixer | Kitchen System |
| Arthur Poggenburg | Code Design | Management System |

***Roles and Organization***

### Roles

The leader is expected to oversee the whole code project as well as write code for the project.

The bug fixer is meant to proofread the code and fix any issues that may arise.

The code support will help create the code as well as bug fix.

The code design will help create the code’s function, design, and the overall big picture.

# REQUIREMENTS PROCESSES

This section will review each step we will take to create the “Lots of Chicken” restaurant code.

## OVERVIEW

The first step for the project is designing the idea of our restaurant. Our goal was to make a simple restaurant that would meet all of the requirements given. After designing the concept of the restaurant, the goal is to turn it into a flowchart. Then the flowcharts will then turn into pseudo code. After that code design and creation will officially begin. The goal is to create the assignment in small chunks to avoid conflict and bugs from occurring.

### Phase One - Design

We first started out with a concept - chicken. The idea was that the restaurant would be themed around chicken to meet the requirements given. Chicken can be served in a variety of ways like nuggets, sandwiches, or even by itself!

### Phase Two - Flowchart

We then created a flow to map our ideas. Flowcharts help understand the goals and sets we will take towards the end project, the code itself.

### Phase Three - Pseudo Code

We then will convert our flowcharts into pseudo code so that we can get our designs and work closer to actual functioning code.

### Phase Four - Coding

The last step, we will then work on coding the project. By this step our designs and ideas will be polished and ready to be converted into code. This will be the phase that takes the longest.

# TOOLS

|  |  |  |
| --- | --- | --- |
| **Tool** | **Version** | **Use** |
| C++ Compiler (GDB) | C++ 17 | Used to compile and run the code. |
| Google Docs | Most Recent | Used to create the management document. |

1. ***List of Tools***

# REQUIREMENTS DOCUMENTATION AND ORGANIZATION

## REQUIREMENTS DOCUMENTATION

The requirements that will be produced are three different systems of code: Management System, Order Application System, and Kitchen System, that will work together to improve a restaurant environment.

### Breakdown Structures

ORDER APPLICATION SYSTEM:

* Ability to only view the active menus.

- Ability to view optional ingredients in a meal.

- Ability to view cooking preferences of ingredients that can be cooked diﬀerent ways.

- Ability to add a new order to a table.

- Ability to add a new order without deﬁning a table.

- Ability to add a suborder to an existing order.

- Ability to delete a suborder that has not yet been conﬁrmed within the order system.

- Ability to view the active suborder details

- Ability to view the status of all active suborders.

- Ability to alert the waiter when the drinks are complete.

- Ability to view transaction list of current order.

- Ability to delete an item or clear the transaction list.

- Ability to print customer receipts on order completion.

KITCHEN SYSTEM:

* Organised display of active orders.
  + View preferences and optional choices of every meal.
* Display elapsed time and progress of each order.
  + Inform waiter; update order to ‘in progress’.
  + Inform waiter; update order to ‘ready’.

MANAGEMENT SYSTEM;

* **MENU**
  + Ability to add ingredients, meals and menus.
  + Ability to edit ingredients, meals and menus.
  + Ability to remove ingredients, meals and menus.
  + Ability to alter the grid size of the meals, menu sections and menus.
  + Ability to change transparency of an image that represents a meal.
  + Ability to add time intervals to menus and oﬀers.
    - Ability to add, edit and remove oﬀers.
* **STOCK**
  + Ability to display the real time stock levels.
  + Means of stock checking.
  + Means of updating the minimum stock level.
  + Ability to alter the price of ingredients. (Note: Ingredient might still be in stock at a diﬀerent prices)
* **ETC.**
  + Ability to display statistics of complete orders.
  + Ability to add new waiters to the system.
  + Ability to alter the seating layout.

### Associated Information

Describe the information that will be associated with each requirement and who is responsible for collecting the information.

|  |  |  |
| --- | --- | --- |
| **Associated Information** | **Use** | **Captured By** |
| Change history | Change control and audit | RM Tool |
| Priority | Implementation planning | Analyst |
| Unique ID | Traceability matrix | RM Tool |

1. ***Associated Information***

## ORGANIZATION

### Numbering Convention

I - Management System

II - Order Application System

III - Kitchen System

# MEASURES

There are three sections of requirements for our restaurant: the order application system, the kitchen system, as well as the management system. Kenneth and Arthur will be working on the order application system, Jessica will be working on the kitchen system, and Nathaniel will be working on the management system.

By working individually on each system, we will be able to work efficiently. At the same time, there will be recurring meetings to avoid miscommunication. Specifically at the beginning of the project, we will be focusing our meetings on the overall plan of the project. We will figure out the restaurant one step at a time, and breaking down these components even further. For instance, a restaurant theme will be determined, each food item on the menu will be agreed on according to the theme, a list of ingredients for each of these food items will be written down, etc.

This process will be repeated at the meetings for each requirement in each system, regardless of whether or not we are the one coding the system. This ensures the team members are always on the same page, and we are making decisions for our restaurant as a group rather than an individual.

In person meetings will occur often in order to avoid miscommunication, but there will be multiple online sources we share to store our information. Utilizing Google Docs for our written plans and pseudocode, a group chat as well as drives specifically for .cpp files with a notice of when it was last updated and who it was by, and flowcharts created through Google Drive to summarize our plans from our meetings. There will be notes taken at our meetings and shared with the team members as well, to be able to look back on in case of miscommunication.

Our meeting procedure of breaking down each requirement’s components as well as utilizing multiple online sources will ensure that all team members know exactly how we will be approaching and creating each requirement for our restaurant, as well as optimal communication within the team to avoid missing any requirements in our code.

**Definitions, Acronyms, Abbreviations**

|  |  |
| --- | --- |
| Associated information | Information associated with a requirement, including traceability information. If a requirements management tool is used, the requirements database or repository usually has more associated information than hardcopy documents such as the SRS. |
| Child | Child requirements are decomposed from parent requirements. For example, A is the child of the requirement ABC. |
| Compliance matrix | RTM |
| Constraint | Boundary conditions on how the system must be constructed and implemented, for example, how a COTS package might be selected. |
| Derived | New requirements identified during the development process that trace back to a driving requirement. |
| Goal | States the desired result, not the way to reach it. For example, they system shall reduce operating costs by 10% of 2001 costs. All changes in requirements and design should be passed through stated goals. If they are outside the goals, they should be rejected. |
| Information | Any communication or representation of knowledge such as facts, data, or opinions in any media or form. |
| Non-functional requirement | Relate to characteristics of a system such as performance, reliability, security, accuracy, and so forth. |
| Non-technical requirements | Agreements, conditions, or contractual terms that affect and determine the management activities of a project. |
| Parent | Child requirements are decomposed from parent requirements. See child requirement. |
| PMO | Project management office |
| Requirement | A condition or capability that is wanted or needed. |
| Requirement repository | COTS providing a database or spreadsheet in which the requirements and associate information are stored. |
| RM | Requirements management |
| RTM | Requirements traceability matrix |
| SME | Subject matter expert in one or more areas of the client’s business. |
| SRS | System requirements specification. |
| System functional requirements | Include functional and non-functional requirements on the system. |

1. **Requirements Evaluation Checklists**

Enter the unique ID of the problem requirement(s). Explain in Remarks the reason if “No” is checked. Attach additional sheets if needed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Evaluation Criteria** | **Yes** | **No** | **ID** | **Remarks** |
| A test case is associated with the requirement. |  |  | I,II,III |  |
| Ability to view active suborder details |  |  | I |  |
| Ability to add a new order to a table |  |  | I |  |
| Ability to add a new order without deﬁning a table. |  |  | I |  |
| Ability to add a suborder to an existing order. |  |  | I |  |
| Ability to delete a suborder that has not yet been conﬁrmed within the order system. |  |  | I |  |
| Ability to view optional ingredients in a meal. |  |  | I |  |
| Ability to view cooking preferences of ingredients that can be cooked different ways. |  |  | I |  |
| Ability to only view the active menus. |  |  | I |  |
| Ability to view the status of all active suborders. |  |  | I |  |
| Ability to print customer receipts on order completion. |  |  | I |  |
| Ability to alert the waiter when the drinks are complete. |  |  | I |  |
| Ability to delete an item or clear the transaction list. |  |  | I |  |
| Organised display of active orders. |  |  | II |  |
| View preferences and optional choices of every meal. |  |  | II |  |
| Inform waiter; update order to ‘in progress’. |  |  | II |  |
| Inform waiter; update order to ‘ready’. |  |  | II |  |
| Display elapsed time and progress of each order. |  |  | II |  |
| Ability to display the real time stock levels. |  |  | III |  |
| Ability to add ingredients, meals and menus. |  |  | III |  |
| Ability to edit ingredients, meals and menus. |  |  | III |  |
| Ability to remove ingredients, meals and menus. |  |  | III |  |
| Ability to display statistics of complete orders. |  |  | III |  |
| Ability to add new waiters to the system.  Ability to add, edit and remove offers. |  |  | III |  |
| Ability to add time intervals to menus and offers. |  |  | III |  |
| Ability to alter the grid size of the meals, menu sections and menus. |  |  | III |  |
| Ability to alter the seating layout. |  |  | III |  |
| Ability to alter the price of ingredients. (Note: Ingredient might still be in stock at a different price) |  |  | III |  |
| Ability to change transparency of an image that represents a meal. |  |  | III |  |
| Means of stock checking. |  |  | III |  |
| Means of updating the minimum stock level. |  |  | III |  |

1. ***Checklist for Individual Requirements***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Evaluation Criteria - All Requirements** | **Yes** | **No** | **IDs** | **Remarks** |
| Requirements are consistent with each other. |  |  |  |  |
| Requirements are complete: every case or scenario is addressed. |  |  |  |  |
| Requirements address user interfaces. |  |  |  |  |
| Non-functional requirements are addressed. |  |  |  |  |
| Assumptions and dependencies for requirements are stated. |  |  |  |  |
| Requirements address system and user error conditions. |  |  |  |  |
| All requirements are traced to their parent or driver (no dropped traceability). |  |  |  |  |
| Interfaces are specified (internal/external). |  |  |  |  |
| Inputs and outputs are specified. |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. ***Checklist for All Requirements*Quality Standards**

Good quality code is code that has been tested for all possibilities, easy to read and/or consistent, organized, efficient, and is able to accommodate to changes.

The code must have been tested for as many situations as possible. Pertaining to the course project, the code must accommodate for customers with food allergies, ability to tell the customer when there are not enough ingredients to create their order rather than entering negative ingredients, etc. The code must be ready and tested for any situation, as it would be in real life.

Readability, consistency, and organization are similar to one another in that there must be clarity for any reader who is reviewing the code. In a real world situation, our code will be passed along to multiple people, and therefore must be understandable to anyone. Though any computer will be able to compile and run code if there are no errors, the human must be able to read the code to complete necessary tasks or add any changes.

Efficiency is important, as it’s essentially the reason why concepts in coding such as functions, classes, etc., exist to avoid hard coding. Having inefficient code could also affect readability and organization, making it difficult for one who didn’t work on the code to add necessary changes.

The code must be able to accommodate to changes. Pertaining to the course project, if the team members wanted to replace specific ingredients for a specific dish, we should be able to do so without facing difficulty and re-writing lines of code. Being able to accommodate to changes also affects reusability, which would be important in the real world.