Software Requirements Document

for

Restaurant POS System

Version 0.1

Prepared by

Group Name: Old Town Road

Group Members:

Sultan Al Bogami	**saalboga@uncg.edu**
Ian Wilhelmsen	**imwilhel@uncg.edu**
Ashim Chalise	**a_chali2@uncg.edu**
Raiana Zaman	**rszaman@uncg.edu**

University of North Carolina at Greensboro

May 2020

1 INTRODUCTION

1.1 PURPOSE

1.1.1 Restaurant POS System is a desktop application developed using Java language to serve as a point of sale software for a small pizza shop. By developing this software, the student group is practicing finding and applying solutions to problems encountered in delivering high quality, large-scale, real-world software systems in a timely and cost-effective manner.

1.2 DOCUMENT CONVENTIONS

1.2.1 This Document was created based on the IEEE Guide for Software Requirements Specifications referenced in section 1.8.1. of this document.

1.3 INTENDED AUDIENCE

1.3.1 This software is developed to be solely used as a means of evaluating the group members, whom are developing this software, on the following:

1.4 DEMONSTRATING KNOWLEDGE OF SOFTWARE ENGINEERING PRINCIPLES AND TERMINOLOGY.

1.4.1 Demonstrating knowledge of object-oriented modeling techniques (UML).

1.4.2 Applying knowledge in all of the requirements, analysis, design, implementation, and testing of a software system in a team project, and presenting project deliverables in written form.

1.4.3 Demonstrating knowledge of collaborative work practices and tools.

1.4.4 Demonstrating knowledge of software evaluation.

1.5 DEFINITIONS/JARGON

1.5.1 POS: An abbreviation for Point of Sale, which is the time and place where a retail transaction is completed.

1.5.2 UML: An abbreviation for Unified Modeling language.

1.5.3 IDE: An abbreviation for integrated development environment, which is a software application that provides comprehensive facilities to computer programmers for software development.

1.5.4 API: An abbreviation for application programming interface, which is a computing interface to a software component or a system defining how other components or systems can use it.

1.5.5 RAM: Random Access Memory.

1.5.6 JRE: Java Runtime Environment.

1.6 PROJECT SCOPE

1.6.1 This project will consist of creating a POS software for a small pizza shop that will assist the primary end user called server with several functionalities including but not limited to: Seating a guest, selecting a table, making an order, keeping a tab open, and logging off. The project will be completed by May, 2020.

1.7 TECHNICAL CHALLENGES

1.7.1 Using Google API for the Sign in feature of our software is one of the technical challenges that has been encountered. Another challenge is employing MVC best practices in the terms of passing data between views.

1.8 HONOR CODE

1.8.1 The group Old Town Road in developing this software abides by the Academic Integrity Policy of UNCG referenced in section 1.8.2. of this document.

1.9 REFERENCES

1.9.1 "IEEE Guide for Software Requirements Specifications," in IEEE Std 830-1984 , vol., no., pp.1-26, 10 Feb. 1984

1.9.2 "Academic Integrity Policy - Revised 5-8-2017.Pdf. " UNCG, drive.google.com/file/d/0B0rFGGhJvbDHUExSZmFFaWFmb00/view.

1.9.3 Grady Booch, James Rumbaugh, and Ivar Jacobson. 2005. Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series). Addison-Wesley Professional.

1.9.4 "Java® Platform, Standard Edition & Java Development KitVersion 10 API Specification." Overview (Java SE 10 & JDK 10), docs.oracle.com/javase/10/docs/api/overview-summary.html.

1.9.5 Pankaj. "CallableStatement in Java Example." JournalDev, www.journaldev.com/2502/callablestatement-in-java-example.

1.9.6 Quigely, Ike. Data Storage Classes. 10/9/2019

2 DESCRIPTION

2.1 PRODUCT FEATURES

2.1.1 Login with Google.

2.1.1.1 The software uses Google's Sign in feature as an API which helps users log in with their existing Google ID.

2.1.2 Create a food order or drink order.

2.1.2.1 The software allows the user to create a food or a drink order for the guest. The software calls the database for the Menu and the user can create a food or a drink order.

2.1.3 Edit an order.

2.1.3.1 The software allows the user to void an order. If the guest changes their mind, the user can void single items, multiple items or the whole order. Once the order is sent to the kitchen, it cannot be void.

2.1.4 Send and Print the order.

2.1.4.1 The software sends and prints the order. The order is then saved in a database, and is printed in the required places (i.e. In the kitchen, food orders only).

2.1.5 Cash out with the guest.

2.1.5.1 The software allows the user to accept a method of payment and process it. The software will agree upon cash or debit/credit/atm cards. Amex, Visa, Mastercard, Discover only accepted.

2.1.6 Ability to keep a tab open.

2.1.6.1 The software can keep a tab open if the guest plans on ordering more food, drink or dessert. You cannot void items that are sent in an open tab, but you can add items and close the tab out when necessary.

2.1.7 Log off.

2.1.7.1 The software allows the user to log off for privacy concerns. To prevent the unwanted misuse of the credentials, the software lets the user logout or automatically logs off in 3 minutes of inactivity.

2.2 USER CHARACTERISTIC

2.2.1 Single Sign On.

2.2.1.1 The software allows a single user to sign on and do the tasks required. The user does not require to sign in every time he/she completes an order and needs to start another order.

2.2.2 Task Oriented Progression

2.2.2.1 The software helps the user navigate through a single task. It progresses with the task. It is very user intuitive.

2.3 Operating Environment

2.3.1 An Operating System with Java IDE

2.3.1.1 This software is built with Java programming language, so, in order to run it, an Operating System with Java IDE is required. Minimum requirements must be met on the operating environment (See below).

2.3.2 A form of input (Keyboard & Mouse or Touchscreen)

2.3.2.1 The main purpose of this software is to serve as a Point Of Sales software in a Pizza restaurant. So, to be able to select the menu, edit an order, cater to the guests needs, a form of input is vital. A team of keyboard and mouse or touchscreen is necessary for this program to run successfully.

2.3.3 A Monitor

2.3.3.1 In order to see the menu, the options and the software itself, a monitor is vital to this software. The clients can use touch screen monitor so that they don't have to use a keyboard and mouse.

2.4 Design and Implementation Constraints

2.4.1 Limitation in available skills

2.4.1.1 There were some skill limitations in this project for the graphics department. Due to not having graphically experienced personnel, the project could not attain a certain industry standard with graphics.

2.5 Assumptions and Dependencies

2.5.1 User Knowledge and Capabilities

2.5.1.1 It is assumed that the user has a good knowledge of using a computer and is capable of reading the prompts on the screen and following the instructions to complete the task. It is assumed that he/she can turn on/off a computer and run the program.

2.5.2 Environment Conditions

2.5.2.1 It is assumed that the computer running the software will run in a proper indoor or controlled outdoor environment. This software is recommended to be used in a restaurant (preferably Pizza restaurant) which can also be put as a fast food restaurant with seating. The computer depends on the temperature not being too hot or too cold, hence a room temperature would be perfect for the software to run.

2.5.3 Assumed Utilities

2.5.3.1 It is assumed that the computer is receiving energy(electricity) to run and is connected to a stable internet connection with adequate speed (1 Mbs required).

3 FUNCTIONAL REQUIREMENTS

3.1 PRIMARY FUNCTION

3.1.1 Synopsis

3.1.1.1 The Primary Functional requirements of the Point of Sales (POS) are to assist the primary end user, called the server, in facilitating their customer-oriented workflow in a commercial capacity. The first initial case that the POS will handle is a server working in a counter style pizza shop that will eventually evolve into a full-service pizza restaurant. The server has a limited number of responsibilities including:

3.1.2 Logging in

3.1.2.1 The users will be able to log into the POS via the Google SSO API. The Google SSO will facilitate the more diverse functionality and increase user ease of access. Future administrators of shops and their systems will be able to update authorized user lists without having to manage passwords, salts, regular expressions, or users across multiple groups and/or roles.

3.1.3 Seating Guests

3.1.3.1 On operations of scale, the ability to divide the server workflow onto multiple end users or collapse the workflow to a single stationary employee like a cashier diversifies the user of the application. Seating guests is a task that has several implications. The start of a workflow, or series of actions, where, in the case of this application, goods or services are exchanged to a client. It signifies a placeholder where to find a client for delivery of goods. It creates a chain of responsibility on a high level through a workflow and more importantly to client satisfaction.

3.1.4 Taking an Order:

3.1.4.1 At the very base level, POS will specify a set of products and their specifications for production and delivery. POS will eventually be able to disperse a collated set of items among several end user distributors (cooks, stockers, etc.).

3.1.5 Accepting a Pseudo Payment:

3.1.5.1 Without specifying the type or vendor of microtransactions, POS will catalogue a pseudo payment type with a datetime, server ID, a reference to the items ordered by the table. The list of tickets and tables serviced should be made available at the end of day sales report, also known as closing out.

3.1.6 Closing out:

3.1.6.1 At the end of the work day, customers facing employees are expected to account for their portion of the sales for that business day. That action, called closing out, requires that the server turn over money to the restaurant for all amounts of tender clients have turned over throughout the shift. Most clients expect a transaction list with payments listed by a datetime stamp. This close out list is ends with amount to be returned to

3.2 SECONDARY

3.2.1 Dining Room View:

3.2.1.1 An optional view to look at a purely numbered view of table laid out in a grid. The view will put in place "ownership" of table or order by marrying the server user and a table/ticket. Selecting a table on this view will create a ticket with a status of "open." While an asynchronous response on this page is the end goal, this advancement will change the amount of time that users will be allowed to be idle on this page and others.

3.3 BENCHMARK SCHEDULE:

3.3.1 Rollout 1: 3/11/2020

3.3.1.1 By the date specified in the above rollout, the following will be finished: Seating Guests, Taking an Order and Accepting a Pseudo Payment.

3.3.2 Rollout 2: 3/24/2020

3.3.2.1 By the date specified in the above rollout, the following will be finished: Dining Room View, Closing out and Logging In.

4 TECHNICAL REQUIREMENTS

4.1 OPERATING SYSTEMS/COMPATIBILITY:

4.1.1 Target Environments

4.1.1.1 The target environment for this application is any environment that runs both a Java Virtual Machine and an instance of a MySQL database if the intent is to run this app on a singular machine. However, this application can be run in a distributed scenario where multiple instances can be at the same time working with only a single instance of the MySQL Database.

4.2 INTERFACE REQUIREMENTS:

4.2.1 User Interface

4.2.1.1 A mouse and keyboard are needed at a base with a card reader for debt/credit payments for registering inputs. A screen the size of a medium tablet would be optimal for the display of information. Alternatively touch screens in an android or apple OS setting would substitute nicely.

4.2.2 Hardware Required

4.2.2.1 Current iterations of the 5th generation commercially tablets will end up being the ideal platform for this application. Smaller mini personal computers are currently the ideal hardware build complete with a network card accepting a wired input, a base HDMI port and 3 USB ports are ideal. Something along the lines of the Dell Nuc paired with a small display.

4.2.3 Software Interfaces:

4.2.3.1 This application will require an instance of MySQL, which is freely available, but must be maintained by the customer. The application does not currently perform any sort of basic task to ensure the data in the database remains uncorrupted, secure, and readily available. Given the initial size of the database the demands for a single store with moderate amount of transactions. Serious consideration must be made to keep this instance well protected and backed up.

4.2.4 Communication Interfaces:

4.2.4.1 Google API is required for POS to function. The application utilizes the Google API to authenticate users without the use of magnetic strip cards, key fobs or a username/password specific to this application. This way the application can utilization the authentication and role hierarchy that the customer specifies. This authentication will later be used in employee time management utility to process punch times and hourly pay for employees.

5 NON-FUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENT:

5.1.1 Operating System

5.1.1.1 Windows 7, 8/8.1, 10; MacOS X; Linux.

5.1.2 RAM

5.1.2.1 128 MB.

5.1.3 Disk space

5.1.3.1 124 MB for JRE; 2 MB for Java Update.

5.1.4 Processor

5.1.4.1 Minimum Pentium 2 266 MHz processor.

5.1.5 Browsers

5.1.5.1 Internet Explorer 9 and above, Firefox, A 64-bit browser (Safari, for example) is required to run Oracle Java on Mac.

5.2 SAFETY / RECOVERY

5.2.1 Disclaimer

5.2.1.1 Any breakdown or damage to the product's code-base, either due to user-interference, application crash, desktop crash etc, could cause save file corruption if done in the middle of a save. It can also cause the application to become unstable, and may require a full reinstallation to reset or fix any lingering issues. Any such reset should not negatively affect the ability to log back into a previously created account or access previous account cloud-stored save.

5.3 SECURITY REQUIREMENTS

5.3.1 Base Requirements

5.3.1.1 User requires an account from the user's end in order to use the application. Any user will require a username and password, and this information would need to be stored securely on the cloud or via a separate database. Users are unable to sign up. Only admin is authorized to add new users.

5.4 POLICY REQUIREMENTS

5.4.1 Base Requirements

5.4.1.1 By using this application, you agree that you may have to use your Gmail credentials.

5.4.1.2 This application will not disclose any of your information to any other party.

5.5 SOFTWARE QUALITY ATTRIBUTES

5.5.1 Availability

5.5.1.1 The application is limited to employees only. The application should always be able to be operate by the user once installed on the target machine.

5.5.2 Correctness

5.5.2.1 The measure of correctness is not life threatening but is pertinent. The amounts saved and moved about in the databased for transactions must have some precision. Inaccuracies in these amounts would render the application useless. If the application could not handle this kind of precision then its users would need an application to supplant the primary purpose of this one. It must correctly calculate the ticket price always.

5.5.3 Maintainability

5.5.3.1 The developers will need to maintain the cloud service that stores the relevant account and save data information.

5.5.4 Reusability

5.5.4.1 This application is designed and engineered so that it can be reusable. This means, the code is encapsulated in such a way that makes it easier for future re-usability for other purposes. It can be installed and uninstalled as many times as needed.

5.5.5 Portability

5.5.5.1 The application should be able to be installed and run on any computer or laptop that meets the Performance Requirements specified within this document.

5.6 PROCESS REQUIREMENTS

5.6.1 Methodology Requirement

5.6.1.1 This application was developed using the Agile method.

5.6.2 Time Constraints

5.6.2.1 The product will be developed for approximately 4 months, starting from an initial date of 1/13/2020, to 05/01/2020.

5.6.3 Cost & Delivery Date

5.6.3.1 This application is free to use. Delivery date is 05/01/2020.