| Pizza POS System    **Software Requirements Specification**  **Version 2.0**  **October 23, 2023**  **Alex, Ariat, Malik, Garrett** |
| --- |

**Table of Contents**

[**1. Introduction 2**](#_ohz7r2ywjvt5)

[**1.1 Purpose 2**](#_70w109vf2pwk)

[1.2 Scope 2](#_py5kuzfvpn5i)

[1.3 Definitions, Acronyms, and Abbreviations 3](#_dcu76zs58zz8)

[1.4 Document References 3](#_26n0nyzatcv2)

[1.5 System Overview 3](#_q9z2wjvo2dc2)

[**2. Functional Description 3**](#_bu835va74c0c)

[2.1 Main Screen Requirements 4](#_m6uh7abbdkdc)

[2.1.1 Main Screen readability 4](#_4hn8vucp1udc)

[2.1.2 Main Screen content 4](#_6ti5ggo093yi)

[2.1.3 Main Screen content 4](#_ss3c0aik5kl)

[2.1.4 Return to main page 4](#_tccbkx9lquxy)

[2.2 Menu 4](#_9iq5152rhere)

[2.2.1 Main Menu Pizza Option 4](#_ijnlrzj2iz6m)

[2.2.2 Main Menu Build Your Own Pizza Option 5](#_q1nehbkxwyr)

[2.2.3 Cart and check out 5](#_uipe1dpxii5o)

[2.2.4 Admin Functions 5](#_v6spairlw200)

[2.2.6 Compatibility 5](#_7ut4gaxc7mqj)

[2.3 Use case 6](#_olas7zylx5tx)

[2.3.1 Placing order 6](#_j7imvchglm4j)

[2.3.2 Process payment 6](#_bi3tcjqjvyxx)

[2.3.3 Manage menu items 6](#_6ccf4rd26fa1)

[2.4 User stories 7](#_8cw9auethhza)

[2.4.1 Placed order 7](#_da9fmgjbubft)

[2.4.2 Payments 7](#_k2uovunnjfqd)

[2.4.3 Menu management 7](#_mqrtg3n6h3vx)

[**3. System Requirements 7**](#_a5rf4mp9m8p7)

[3.1 Hardware Requirements 7](#_afnpw8awenhs)

[3.2 Software Requirements 7](#_iucvmwm7508z)

[**4. Interfaces 7**](#_9phxpeyphfp3)

[4.1 Standalone Program 8](#_7luf17bunslc)

[4.2 Use of Windows DLLs 8](#_bafna7dy2q2d)

[**5. Performance 8**](#_auv0ndu24o0c)

[5.1 Application 8](#_bqcstxth9d9g)

[5.2 UI 8](#_ph8ynbw6pq7)

[**6. Delivery 8**](#_5kea0qtec9t2)

[**7. Schedule 8**](#_6ungr3ng61x1)

[**8. Miscellaneous 9**](#_igh8s3dp54fj)

**Revision History**

| **Date** | **Version** | **Description** | **Author(s)** |
| --- | --- | --- | --- |
| 9/25/23 | 1.0 | SRS first version | Ariat, Alex, Garrett, Malik |
| 10/23/2023 | 2.0 | An updated version of the first SRS document | Ariat, Alex, Garret, Malik |

# **1. Introduction**

# **1.1 Purpose**

The purpose of this document is to specify the software requirements for the functional capabilities of the Pizza topping app being designed for users to place an order and get the total amount for said order. The requirements contained in this document will serve as the basis for the creation of Test plans to ensure that each requirement is satisfied.

## **1.2 Scope**

The scope of this Software Requirements Specification (SRS) document is thorough and includes various required features. It covers functional requirements, consisting of detailed descriptions of the pizza topping software's features and functionalities. These comprises user interactions, system behavior for ordering, customization, payment processes, and user account management, including registration and authentication.

Non-functional requirements are defined, covering performance criteria like response times and system capacity, security measures for user data and payment processing, usability and user experience guidelines, and compatibility considerations for web browsers.

Moreover, the SRS document includes use cases and user stories to illustrate real-world scenarios and system interactions.

Lastly, it addresses business rules, specifying the intricate logic and regulations that must be implemented within the software, such as order validation criteria.

## **1.3 Definitions, Acronyms, and Abbreviations**

| **Acronym** | **Meaning** |
| --- | --- |
| Admin | individual or user role responsible for managing and overseeing certain aspects of a system, software application, website, network, or organization |
| DLL | Dynamic Link Libraries |
| SDD | Software Design Document |
| SDP | Software Development Plan |
| SRS | Software Requirements Specification |
| Stakeholder | Any person who has interaction with the system who is not a developer |
| STP | Software Test Plan |
| UI | User Interface |

## **1.4 Document References**

There is no doc references

## **1.5 System Overview**

Our goal is to create an application that caters to customers placing a pizza order, providing them with a delightful pizza-topping experience. This user-friendly app, designed for any desktop (mainly windows), will engage users in a world of pizza customization. Users will be able to look at a wide array of pizza toppings, select their topping, and seamlessly complete their orders. Elevating the pizza selection process, the app will offer intuitive search and menu selection, making it a breeze to discover the perfect toppings. Our app is crafted to accommodate the diverse tastes of pizza customers, ensuring it's accessible on desktop.

# **2. Functional Description**

This section includes the requirements that define all the fundamental actions of the software system. The functional requirements for this software application process are shaped to reflect the user's description of the application’s function, including both basic requirements and the right features. Requirement descriptions, specified by the use of “shall”, make the essential functionality that the system must deliver, ensuring traceability and furthering testing.

For example, the system shall allow users to customize their pizza with various toppings and sizes.

In addition, “desirement” statements, indicated by “should,” focus on the desired features that increase the customer satisfaction. For instance, the system should recommend popular toppings based on the customers preferences.

These requirements all define the functionality and attributes of the pizza toppings application, providing a thorough direction for its development and testing, while the non-requirement statements give condition and clarity to help better understand it.

## **2.1 Main Screen Requirements**

The following requirements describe the main screen functions of the Pizza tops application.

### **2.1.1 Main Screen readability**

The main screen of the application should be accessible and readable. The main screen should provide quick access to general transactions, more specifically orders, payments and receipts. The main screen layout should have easily readable buttons that will allow you to switch between transactions, inventory information, customer information, employee information ect. When it comes to readability the system will be able to display messages in an organized way in case of errors (problems with transactions). Buttons will be formatted in a proper way (large enough to see) and messages describing what buttons are, will be large enough to read and easily clickable.

### **2.1.2 Main Screen content**

The main screen of the system shall have a menu, featured toppings, and cart to add ordered items.

### **2.1.3 Main Screen content**

The main screen of the application shall show the following features

* Menu
* Cart and checkout
* Featured toppings based on customer preferences

### **2.1.4 Return to main page**

The system screen shall contain a return link to go back to the main screen/ the menu.

## **2.2 Menu**

## **2.2.1 Main Menu Pizza Option**

The system shall allow users to browse through a menu of pizza options, including pre-configured pizzas and build their own pizza option.

## **2.2.2 Main Menu Build Your Own Pizza Option**

The “build own pizza” feature shall allow the employees to select preferred crust type, size, and toppings. Users can remove or add toppings as desired.

System shall show updates to pizza’s price as the users customize it.

## **2.2.3 Cart and check out**

Users shall add pizzas to their cart, view cart contents, and edit the items.

Users shall proceed to check

The system shall calculate and display the total order amounts.

## **2.2.4 Admin Functions**

Admin users shall use the ID login to access to management console to oversee the menu, including editing pizzas and topping and the menu

Admin shall view and manage customer orders, mark them as processed, and view order details.

**2.2.5 Log in and Log Out**

Each employee shall have their own unique four digit pin which they use to log in and log out. If the user is not able to log in, the system should have an automatic lock mechanism where only the manager is allowed to override.

If a user misenters their pin then an error message will pop up.

If a user would like to change their pin, they must select that option to do so, re-enter their original pin which then an option of changing users pin will pop up.

## **2.2.6 Compatibility**

The application shall run as expected on windows, ensuring accessibility for stakeholders.

**2.2.7 Taxing, Display Cost Information**

The system shall have a location setting to make sure the tax amount applies to its state tax. There will also be an option to override tax in case the customer is tax exempt. The system shall calculate the total of the order and include the tax. The system shall also remember what the tax is unless it is changed.

1. Total Before Tax: Once order is complete, the system displays the total price before tax
2. Total Tax: Calculate the tax from the total
3. Final Total: Add the tax amount and pre-tax-total. This will pop up on the screen for the employee to tell the customer

The employee shall be allowed to override prices before completing the order. This will only change the price for that specific item for that order. To change an item's price indefinitely is only allowed when no transaction is being done and in the inventory maintenance section.

## **2.3 Use case**

### **2.3.1 Placing order**

This use case shall allow the users to create and submit pizza orders.

Actors - users.

Precondition - customers should first select items from the menu

The flow:

users selects items from the menu

users customizes the items (different toppings, and size)

users add items to cart

users review the order and submit it

Other flows:

If the order is canceled, it is removed from cart

Postcondition - the order is confirmed by the admin

### **2.3.2 Process payment**

This use case shall allow both users and admin to process payments

Actors - user and employees

Precondition - customer should first submit an order

The flow:

Order is submitted

Employee calculates the total amount

user submit payment

System processes the payment

Other flow:

If payment declined, notify and retry

Postcondition - payment authorized and update the order status

### **2.3.3 Manage menu items**

This use case shall allow admin to add, edit, or delete menu items.

Actors - employees

Precondition - users logged in as an admin.

The flow:

Admin login to access the menu management system

Admin adds, edits, or deletes menu items

Postcondition - the admin confirms orders, and can update the menu with new items.

## **2.4 User stories**

### **2.4.1 Placed order**

As a customer, I should be able to customize my pizza order with toppings and special instructions, so my pizza is created to my liking.

Criteria:

I shall select from a list of available toppings.

I shall be able to specify special preferences for my order.

My orders shall reflect my customized preferences.

### **2.4.2 Payments**

As an employee, I should be able to calculate the total order amount and process any payments to ensure users check out.

Criteria:

I shall input the order items and calculate the total amount accurately.

System shall process any payments.

### **2.4.3 Menu management**

As a user, I should be able to login and manage the menu.

Criteria:

I shall login and adjust the menu items with names, description, and prices.

# **3. System Requirements**

The minimum hardware requirement for this application shall be any computer device that includes a keyboard, monitor, mouse

## **3.1 Hardware Requirements**

This application shall require any personal computer device or laptop that is compatible with windows, a monitor, keyboard, and mouse.

## **3.2 Software Requirements**

This application shall require windows 10/11 to run.

# **4. Interfaces**

The pizza topping application covers the standalone program and the use of windows DLLs.

## **4.1 Standalone Program**

The program shall run standalone, not interacting with a network in any manner.

## **4.2 Use of Windows DLLs**

The program shall be self-sufficient, or will use only standard Windows DLLs. This will ensure that no matter what type of computer the customer will be using, the program will be running properly

# **5. Performance**

## **5.1 Application**

The application shall open and load within 5-10 seconds.

## **5.2 UI**

The user interface shall respond to user inputs via mouse and keyboard within 0.5 seconds

# **6. Delivery**

The program shall be delivered via email in a zipped file folder to the customer. This file will not have any bugs and will download properly on the users computer.

# **7. Schedule**

| **Artifact** | **Delivery Date** |
| --- | --- |
| Software Development Plan | 9/13/23 |
| Software Requirements Specification | 9/25/23 |
| Traceability Matrix | 10/2/23 |
| Software Design Document | 10/11/23 |
| Updated SDP | 10/16/23 |
| Updated SRS | 10/23/23 |
| Updated Traceability Matrix | 10/25/23 |
| Updated SDD | 11/1/23 |
| Software Test Plan | 11/8/23 |
| Updated STP | 11/20/23 |
| Source Code due to be delivered | 12/5/23 |

# **8. Miscellaneous**

There are no miscellaneous items at this time.