**Pizza Ordering System**

**Software Requirements Specification**

**Revised**

**Team Number: Group 6**

**Project Manager: Brian Gill**

**Team Members: Arthur L Holmes, Alexander Bass, Joe Dacci, Sheng ‘Tony” Sun, Elham Jahandideh, Kodee Ethridge, William Segree**

**Table of Contents**

1. Introduction
   1. Project Objectives
   2. Project Scope
   3. Project Overview
2. Project Description
   1. Project Features / Functions
   2. User Stories
   3. Use Cases
   4. Project Assumptions and Dependencies
3. Project Collaboration and Documentation
4. Project Management
5. Requirements Specification
   1. Business Requirements
   2. User Requirements
   3. Functional Requirements
   4. Non-Functional Requirements
   5. Implementation (Performance) Requirements
6. High-level Design
   1. Security
   2. Hardware
   3. User Experience
   4. Architecture
   5. Database
   6. Top-level Classes
   7. Data Flow and States
7. **Introduction**
   1. Project Objectives

The primary goal of the Pizza Ordering System is to provide an efficient, user-friendly platform for customers to order pizza online. The system aims to streamline the process of user login, pizza customization, and secure payment processing.

* 1. Project Scope

This project includes the development of a web-based application that allows users to log in with accounts, customize pizzas, view available options, place orders, and make payments online.

* 1. Project Overview

The goal of this project is to deliver an intuitive program that directly connects to clients restaurant so that the customer can place orders and make secure credit/debit payments with the application. To achieve this goal, the development team will plan out, code, and thoroughly test the application over the course of several sprints so that all requirements are met.

1. **Project Description**
   1. Project Features / Functions
2. **User Account Creation and Login –** Create a user Loginfor the customer (user).
3. **Pizza Customization** – Users can select pizza size, crust type, and toppings.
4. **Shopping Cart** - Users can review their order within the cart before placing the order.
5. **Secure Online Payment** – The user will have an option to directly pay within the app, otherwise cash payment can be accepted at the restaurant.
   1. User Stories
      1. As a customer, I want to create a user account for the restaurants ordering website.
      2. As a customer, I want to customize my pizza so that I can choose my favorite toppings.
      3. As a customer, I want to review my order before paying so I don’t order the wrong thing.
      4. As a customer, I want to securely pay for my order using the application.
   2. Use Case
      1. Use Case 1: Pizza Customization
         * Actors: Customer
         * Precondition: The customer is logged into the system.
         * Description: The customer selects a pizza, chooses a size, crust type, and toppings, and adds it to the cart.
         * Postcondition: The customized pizza is added to the shopping cart.
      2. Use Case 2: Shopping Cart
         * Actors: Customer
         * Precondition: The customer has a pizza and topping selected and is ready to pay.
         * Description: The customer clicks a “Review Cart” button.
         * Postcondition: The cart correctly displays the items the customer has selected.
      3. Use Case 4: Payment Processing
         * Actors: Customer
         * Precondition: The customer has finalized the order and is ready to pay.
         * Description: The customer selects a payment method and completes the transaction.
         * Postcondition: The order is confirmed, and payment is processed securely.
   3. Project Assumptions and Dependencies
      * + The restaurant's kitchen must have a system in place to receive and manage orders.
        + Internet access is required for customers to use the application.
        + The application depends on third-party payment gateways for secure transactions.
6. **Project Collaboration and Documentation**

The team will use GitHub for version control and Microsoft Teams for communication. Documentation will be maintained with the GitHub repository, and tasks will be discussed and distributed in Teams.

1. **Project Management**

The project will follow an Agile development process, using Scrum methodology. Sprints will be 2 weeks long, with weekly stand-up meetings, sprint planning, and retrospectives.

1. **Requirements Specification**
   1. **Business Requirements**

| Requirement ID | Requirement Description | MOSCOW |
| --- | --- | --- |
| BR1 | The system must provide a way for customers to place orders online. | M |
| BR2 | The system must offer pizza customization options for customers. | M |
| BR3 | The system should provide detailed order analytics for the restaurant management. | S |

* 1. **User Requirements**

| Requirement ID | Requirement Description | MOSCOW |
| --- | --- | --- |
| UR1 | Users must be able to customize their pizza and view the price dynamically. | M |
| UR2 | Users should be able to review their order before paying. | S |
| UR3 | Users must be able to securely pay within the application. | M |
| UR4 | Users should be able to input their email and look at previous orders. | S |
| UR5 | Users should be able to add a new email to the system to store orders. | S |

* 1. **Functional Requirements**

| Requirement ID | Requirement Description | MOSCOW |
| --- | --- | --- |
| FR1 | The system must allow users to customize pizzas by selecting various options. | M |
| FR2 | The system should display a shopping cart with the items the user has selected. | S |
| FR3 | The system must complete transactions after verifying payment methods. | M |

* 1. **Non-Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Description | MOSCOW |
| NFR1 | The system must have a user-friendly interface that allows users to order pizza easily. | M |
| NFR2 | The system could handle up to 500 concurrent users without performance degradation. | C |

1. **High-level Design**
   1. **Security**

**Requirement ID**: *SEC1*

**Description:** Authentication and Authorization

**Requirement**: Require customers to create accounts with secure passwords and optionally offer MFA(Multi-Factor Authentication) for added security, especially for repeat orders or saved payment information. Error Logging: Record errors or failures in the system like payments gateway issues, database connection errors to quickly resolve problems and maintain website reliability

**Requirement ID:** *SEC2*

**Description:** Logging and Monitoring

**Requirement:** Log all users activities, including login, order placements, payment

processing and profile updates, to track user interactions and detect anomalies. We will

use cryptographic hashing to ensure that logs remain tamper-proof, this allows us to verify

logs that have not been altered maliciously.

* 1. **Hardware**

**Requirement ID**:HW1

**Description**: Processor (CPU)

**Requirement**: A high-performance processor is needed to handle multiple orders simultaneously, process transactions, and manage interactions with the database.

* Specification 1: Minimum 4-core, 3.0 GHz processor
* Specification 2: Compatible with cloud computing

**Requirement ID**: HW2

**Description**: Storage (Disk Space)

**Requirement**: A reliable storage system to store customer data and transaction history securely and efficiently.

* Specification 1: Minimum 500GB SSD
* Specification 2: Sensitive information is encrypted
* Specification 3: Customer data and transaction history are efficiently stored in the database

**Requirement ID**: HW3

**Description**: Network

**Requirement**: Stable, high-speed infrastructure to ensure smooth communication between the server and client.

* Specification 1: Minimum 1 Gbps Ethernet connection
* Specification 2: Firewall support
* Specification 3: At least 99.9% uptime

* 1. **User Experience**

**Requirement ID: UX1**

**Description**: User Login

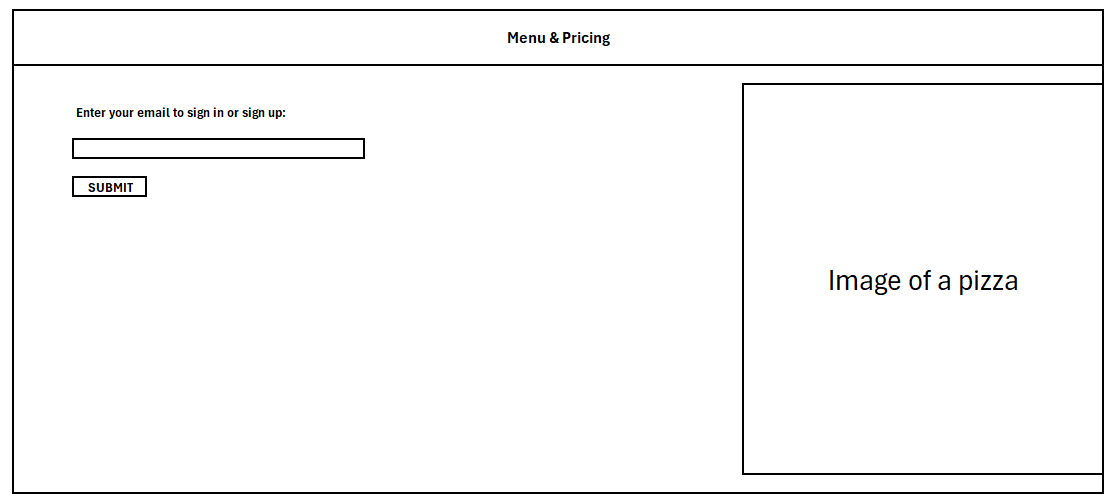
**Requirements**:

* Users will enter their email.
* If the email is already in the database, the user must be redirected to build a pizza.
* If the email is not already in the database, the email must be added to the database.
* All emails must be in a valid email format.

**Specifications**:

* An error message will appear if the user enters an invalid email.
* When a valid email is entered (existing or new), the user will be redirected to build a pizza.

**Wireframe**:

**

**Requirement ID: UX2**

**Description**: Pizza Customization

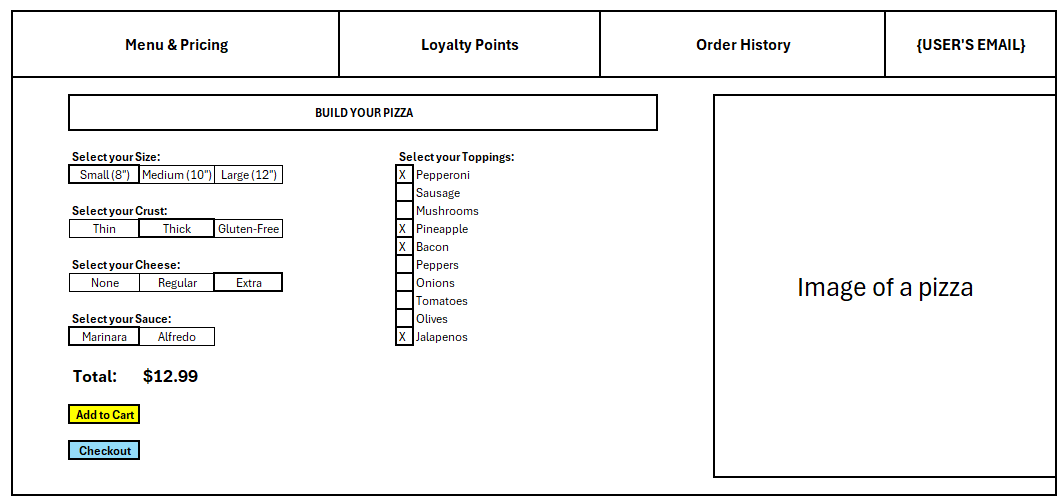
**Requirements**:

* Users must choose exactly 1 size option (small, medium, large)
* Users must choose exactly 1 crust option (thin, thick, gluten-free)
* Users must choose exactly 1 cheese option (no cheese, regular cheese, extra cheese)
* Users must choose exactly 1 sauce option
* Users MAY choose up to 10 different pizza topping options
* The pizza’s total cost must be displayed at the bottom of the page.
* The user can select “Add to Cart” to add the pizza to the cart.
* The user can select “Checkout” to proceed to finalizing their order.

**Specifications**:

* If a user tries to select “Add to Cart” without a size, crust, cheese, or sauce option: an error message will display and the pizza will not be added.
* The checkout option will only display if the user has at least one pizza in the cart.

**Wireframe**:

**

* 1. **Architecture**

**Requirement ID**: ARCH1

**Description**: Layered (N-Tier) Architecture

**Requirement**: The application will be structured into five layers, Presentation, Application, Database, Integration, and Security.

**Specifications**:

* + 1. **Presentation Layer**
       - Description: The Pizza Order System will be displayed as a web application which will include various web pages for browsing menu options, shopping cart, delivery tracking, and payment screens.
       - Tools & Languages: HTML, CSS, JavaScript, and a front-end framework (React)
    2. **Application Layer**
       - Description: The core logic that controls the processes of customizing a pizza, securing payment information, and tracking deliveries.
       - Tools & Languages: Python and a back-end framework (Django)
    3. **Database Layer**
       - Description: This is where all data is stored, including customer information, transaction history, etc.
       - Tools & Languages: SQL and a Database Management System
    4. **Integration Layer**
       - Description: The application will integrate external systems to process payments and track deliveries.
       - Tools & Languages: APIs and web services
    5. **Security Layer**
       - Description: This will include several security measures such as user authentication, secure payment processing, and encrypted web trafficking.
       - Tools & Languages: APIs and web services
  1. **Database**

**Requirement ID:** DB1

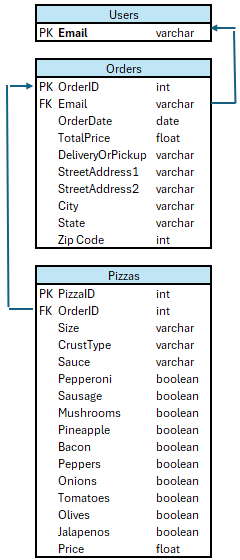
**Description**: Pizza Order and Customer Database

**Requirement**: The DBMS will store user information along with detailed order history.

**Specifications**: The database will have 3 entities:

* + - * The USERS table will store user profiles. This pizza application only obtains the “Email” which will act as the primary key and only attribute of the USERS table.
      * The ORDERS table stores orders. Each order will have a unique “OrderID” that acts as the primary key. Each order will also store the “Email” that acts as the foreign key to the users table. The table will also contain date, total price, delivery or pickup, and address information.
      * The PIZZAS table stores pizzas. Each pizza will have a unique “PizzaID” that acts as the primary key. Each pizza will also store the “OrderID” as a foreign key to the orders table. The pizza table also stories various information about topping selections.

**Entity Relationship Diagram**:



* 1. **Top-level Classes**

**Requirement ID**: CLASS1

**Description**: User Class

**Requirement**: Will handle user creation and ADD users to the database.

* Specifications:
  + Attributes: id, email, position
  + Methods: VerifyEmail(), AddUser(user)

**Requirement ID**: CLASS2

**Description**: Customer Class (Child Class to User)

**Requirement**: Will handle the creation of order.

* Specifications:
  + Attributes: UserName
  + Methods: CreatePizza()

**Requirement ID**: CLASS3

**Description**: Worker Class (Child Class to User)

**Requirement**: Will assemble the Customer's order after reading their order.

* Specifications:
  + Methods: CompleteOrder()

**Requirement ID**: CLASS4

**Description**: Order Class

**Requirement**: Completes the task of gathering needed information from the customer about time, date, and place.

* Specifications:
  + Attributes: OrderID, Email, OrderDate, DeliceryOrPickUp, StreetAddress1, StreetAddress2, City, State, ZipCode
  + Methods: PrintOrderDetails(),SubmitOrder(), GetTotalPrice()

**Requirement ID**: CLASS5

**Description**: Pizza Class

**Requirement**: Will Gather and apply the Customers input to achieve the wanted outcome.

* Specifications:
  + Attributes: PizzaId, OrderId, Size, CrustType, Sauce, Toppings, Price
  + Methods: PrintPizzaDetails(), AddPizza(), GetPrice()

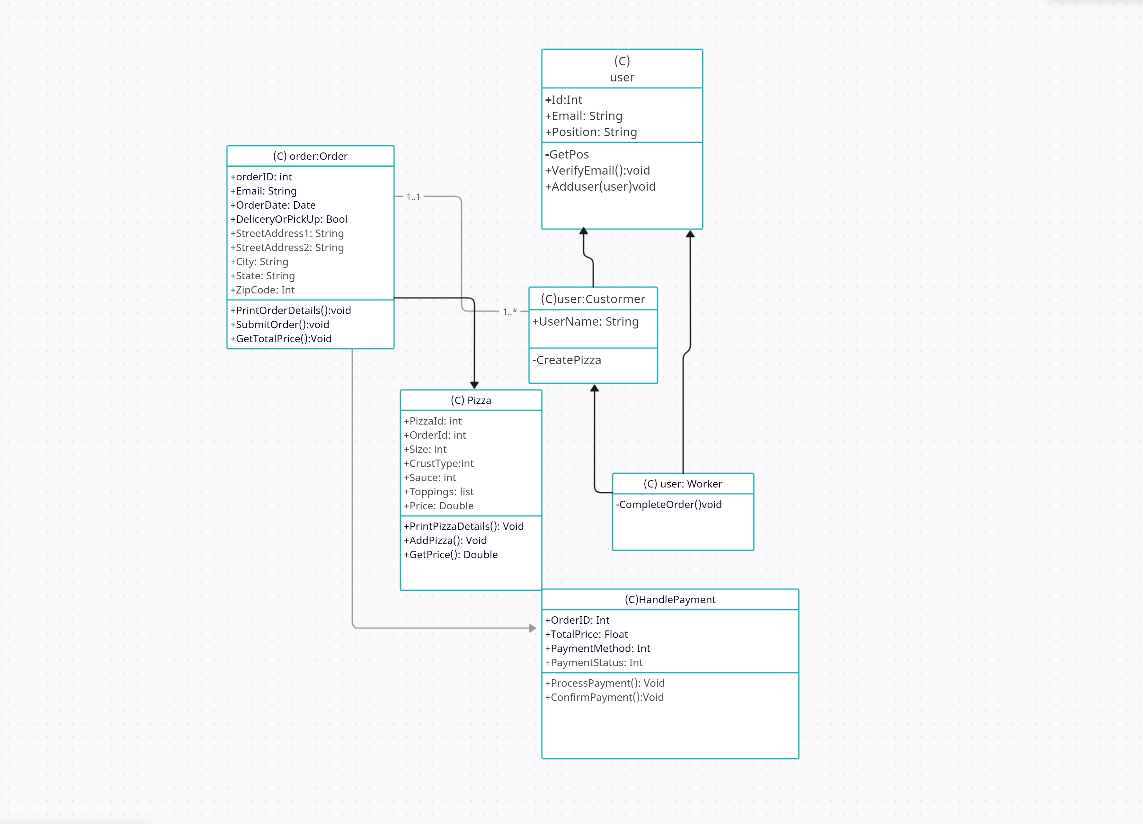
**Requirement ID**: CLASS6

**Description**: HandlePayment Class

**Requirement**: Will calculate the price of the Order and handle the Payment process.

* Specifications:
  + Attributes: OrderId, TotalPrice, PaymentMenthod, PaymentStatus
  + Methods:ProcessPayment(), ConfirmPayment()

Class Diagram

****

* 1. **Data Flow and States**

**Requirement ID:** DF1  
**Description:** Pizza Order Process  
**Requirement:** Design a data flow diagram (DFD) to illustrate the process of ordering a pizza, including states for each main system entity (user, pizza, order, and payment).

1. Data Flow Overview

**Entities:**

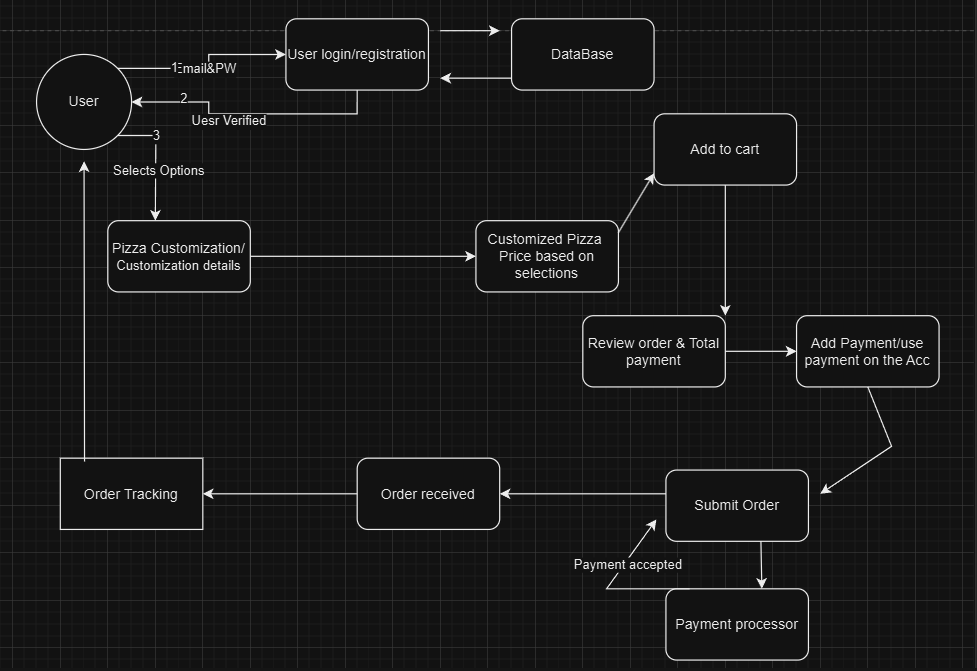
* *Customer (user):* Initiates the order and proceeds through pizza customization, order submission, and payment.
* *Pizza Order*: Represents the customized order details, including size, crust, toppings, and price.
* *Payment Processor*: Verifies and processes payments.

**Data Flow Processes:**

* *Customer Login/Registration*: Users provide or verify their email, with new emails being added to the system.
* *Pizza Customization*: Users create a custom pizza by selecting options for size, crust, toppings, and additional preferences. Customization details are sent to the cart.
* *Order review and Submission:* Users review the pizza(s) in their shopping cart and submit the order.
* *Payment Processing*: User selects a payment method; transaction data is sent to the payment processor.

1. Data Flow Diagram (DFD) - Overview of Processes and Entities DFD Outline

* **User Login/Registration (Start State)**
  + **Input**: User email (existing or new).
  + **Processes**:
    - Validate email format.
    - Check if email exists in the system.
* **Output**:
  + Redirect the user to the customization page (if registered).
  + Store new email in the database if not already registered.
* **Pizza Customization**
  + **Input**: User-selected pizza options (size, crust, toppings).
  + **Processes**:
    - Calculate pizza price based on selections.
  + **Output**:
    - Updated pizza price displayed.
    - Custom pizza added to the shopping cart.
* **Order Review and Submission**
  + **Input**: User reviews items in the shopping cart.
  + **Processes**:
    - Confirm order details and allow modifications.
    - User submits an order for processing.
  + **Output**:
    - Order details finalized and stored.
    - Order sent to payment processing.
* **Payment Processing**
  + **Input**: User payment details.
  + **Processes**:
    - Validate payment method and authenticate transaction.
    - Process payment with third-party gateway.
  + **Output**:
    - Payment confirmation.



**\*\*Order Tracking is now Order Confirmation. Order Confirmation can be seen by User and the Restaurant’s stakeholders.**