Vietnam National University - Ho Chi Minh City

International University

School of Computer Science and Engineering

Principles of Database Management

Project Report

Restaurant Management System

Submitted by

Name: Nguyen Van Lac Thien - ITCSIU22245

Name: Thai Quang Tinh - ITCSIU22222

Name: Nguyen Duc Dai - ITCSIU22180

Date: 05/14/2024

Table of Contents

I. INTRODUCTION

A. Abstract

The Restaurant Management System (RMS) is a comprehensive, Java-based application designed to streamline the operations of a restaurant. The RMS provides an efficient way of managing various aspects of a restaurant including handling bills, managing products and categories, and managing users. The system aims to improve the overall efficiency of restaurant operations and enhance the user experience.

B. System Overview

The Restaurant Management System (RMS) covers a wide range of restaurant management needs, making it easy for managers and staff to handle orders, print and view bills, manage users, and update categories and products. The system is neatly divided into packages, each focusing on different tasks. The "DataAssessObject" (DAO) package serves as the middleman for all database interactions, ensuring data is smoothly retrieved and stored. The "Model" package sets up the core data structures, detailing bills, products, categories, and users. The "UI" package handles everything related to the user interface, letting users add products, modify user details, manage categories, and place orders.

C. Goal

The primary goal of this Restaurant Management System (RMS) is to provide a user-friendly interface for managing part of a restaurant's operations. By automating tasks like order placement, product management, and bill generation, it aims to reduce manual effort and errors, leading to smoother operations.

D. Technique & Tools

The Restaurant Management System is developed using the Java programming language, which offers robustness, security, and platform independence. The system utilizes Data Access Objects (DAOs) for database interactions, providing a consistent interface for database operations. The GUI is implemented using Java's built-in libraries, offering a visually appealing

and intuitive user interface. Furthermore, the RMS incorporates mySQL for efficient database management.

II. PROJECT PLANNING

A. Timeline

Date	Description of project milestone
Tuesday - 04/09/2024	**Assign tasks to members in group**
Wednesday - 04/10/2024	**Assemble resources for project**
Saturday- 04/27/2024	**Created github project** **Following link: Click here**
Saturday - 04/27/2024 **to Wednesday - 05/08/2024**	**Complete software completion and fixing bugs.**
Thursday - 05/09/2024 to Monday - 05/13/2024	**Complete and submit the final report**

B. Roles and Responsibilities

Name	Student ID	Responsibilities
Nguyen Van Lac Thien	**ITCSIU22245**	**Code support** **Report Writing**
Thai Quang Tinh	**ITCSIU22222**	**Primary coding** **Report writing**
Nguyen Duc Dai	**ITCSIU22180**	**Preparing presentation PowerPoint** **Report writing**

C. Resources

The primary resources utilized in this project encompass a variety of software development tools, libraries, and platforms:

- **Java Development Kit (JDK) version 21**: Serves as the foundational platform for building the Java application.
- **Apache Netbeans Integrated Development Environment (IDE)**: Provides a productive environment for writing, debugging, and testing code.
- **Absolute Layout (AbsoluteLayout.jar)**: Assists in the design of the user interface, allowing for precise positioning of UI components.
- **MySQL Connector (mysql-connector-java-8.0.30.jar)**: A JDBC driver used to establish a connection between the Java application and the MySQL database.
- **iText (itextpdf-5.5.9.jar)**: Used for creating and manipulating PDF documents, enabling the application to generate PDF reports or invoices.
- **Github**: Utilized for version control, collaboration, and code hosting, allowing the team to work together efficiently, track changes, and maintain a history of project development.

D. Risk Assessment and Mitigation Strategies

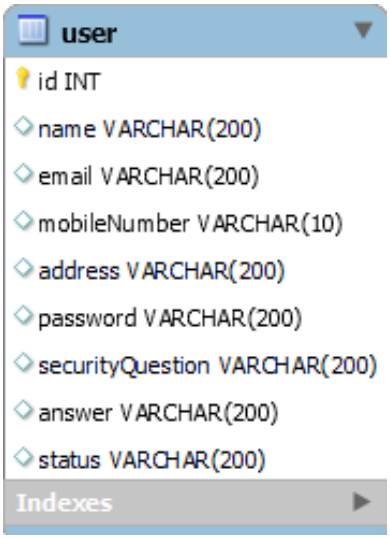
- **Difficulties in Data Retrieval**:
    - Solution: We implemented a comprehensive suite of tests for all data retrieval operations. Exception handling was used to manage unexpected situations during data extraction. Regular reviews and optimizations of database queries were conducted to ensure efficient data access.
- **Malfunctions in User Interface Elements**:
    - Solution: We conducted thorough testing of all user interface components, including buttons and data input fields. Validation checks were added to data input fields to prevent incorrect data entry. A version control platform like GitHub was used to track changes and identify potential sources of malfunctions.
- **Errors in Database Creation Queries**:
    - Solution: We reviewed all database creation queries to ensure they were correctly formulated, paying special attention to foreign keys and table connections.
- **Connection Interruptions with MySQL**:

- Solution: We ensured the MySQL server was properly configured and regularly maintained. Error handling and retry logic were integrated into the application to manage connection disruptions. The system was also monitored to quickly identify and resolve connection issues.

III. LIST OF TABLES

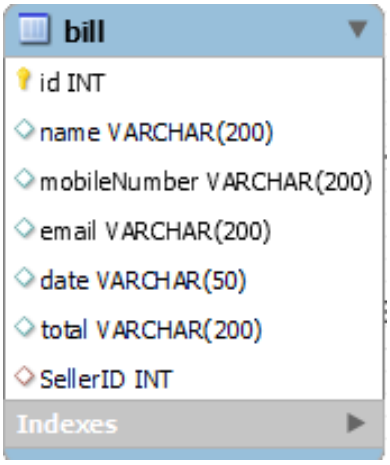
A. User Table:

The "User" Table serves as a repository for critical information pertaining to the administrative personnel and staff members of the restaurant. It encompasses the following fields: ID (Unique), Full name, Email, Address, Mobile Number, Password, Security Question, Answer, and Status.



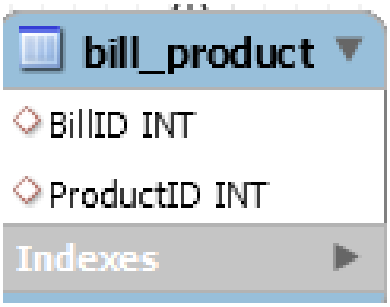
B. Bill Table

The "Bill" Table is designed to capture essential data regarding bills generated within the restaurant, along with the basic customer details. It encompasses fields such as: ID (Unique), Customer Name, Mobile Number, Email, Transaction Date, Total, and the Seller ID (FK) referencing the "User" Table.



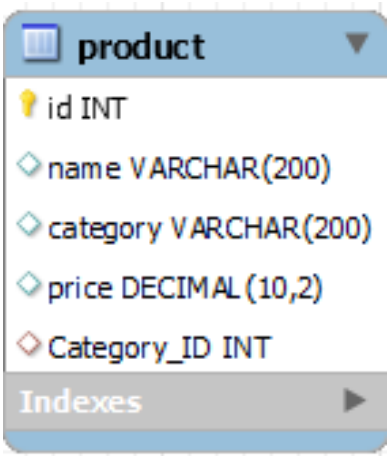
C. Bill_Product Table

The "Bill_Product" Table acts as an intermediary entity to establish a many-to-many relationship between "Bill" and "Product". It comprises Bill ID and Product ID referencing the "Bill" and "Product" Tables.



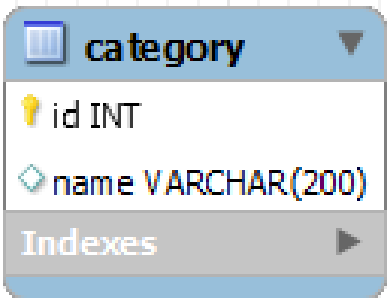
D. Product Table

The "Product" Table serves as a repository for comprehensive information regarding the restaurant's offerings. It includes fields such as: ID (Unique), Product Name, Category, Unit Price, and Category ID (FK) referencing the "Category" Table.



E. Category Table

The "Category" Table maintains a record of all product available within the restaurant's inventory. It encompasses fields such as ID (Unique), and Category Name.



F. Schema

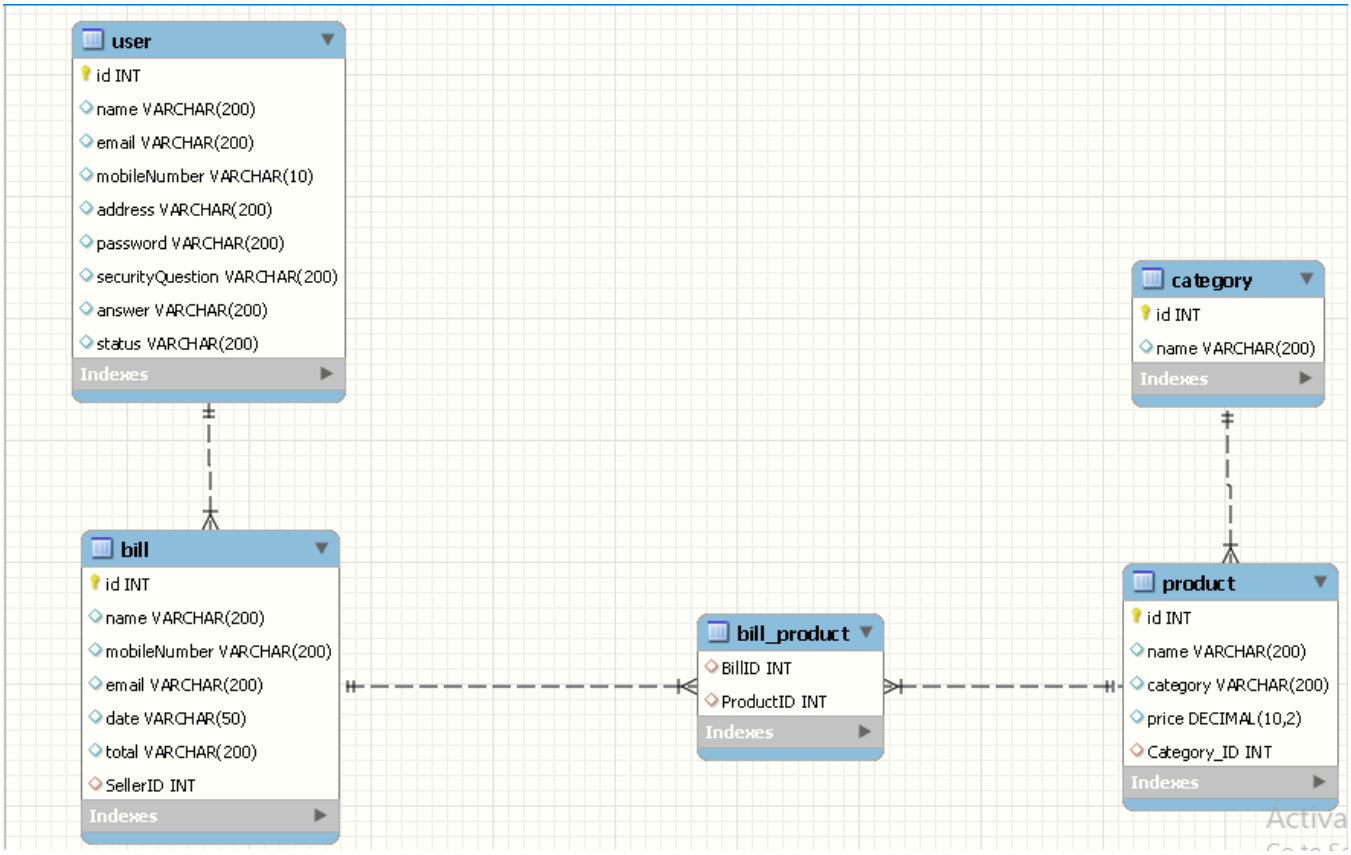


Figure 1. Database schema

IV. LIST OF FIGURES

A. Entity Relationship Diagram

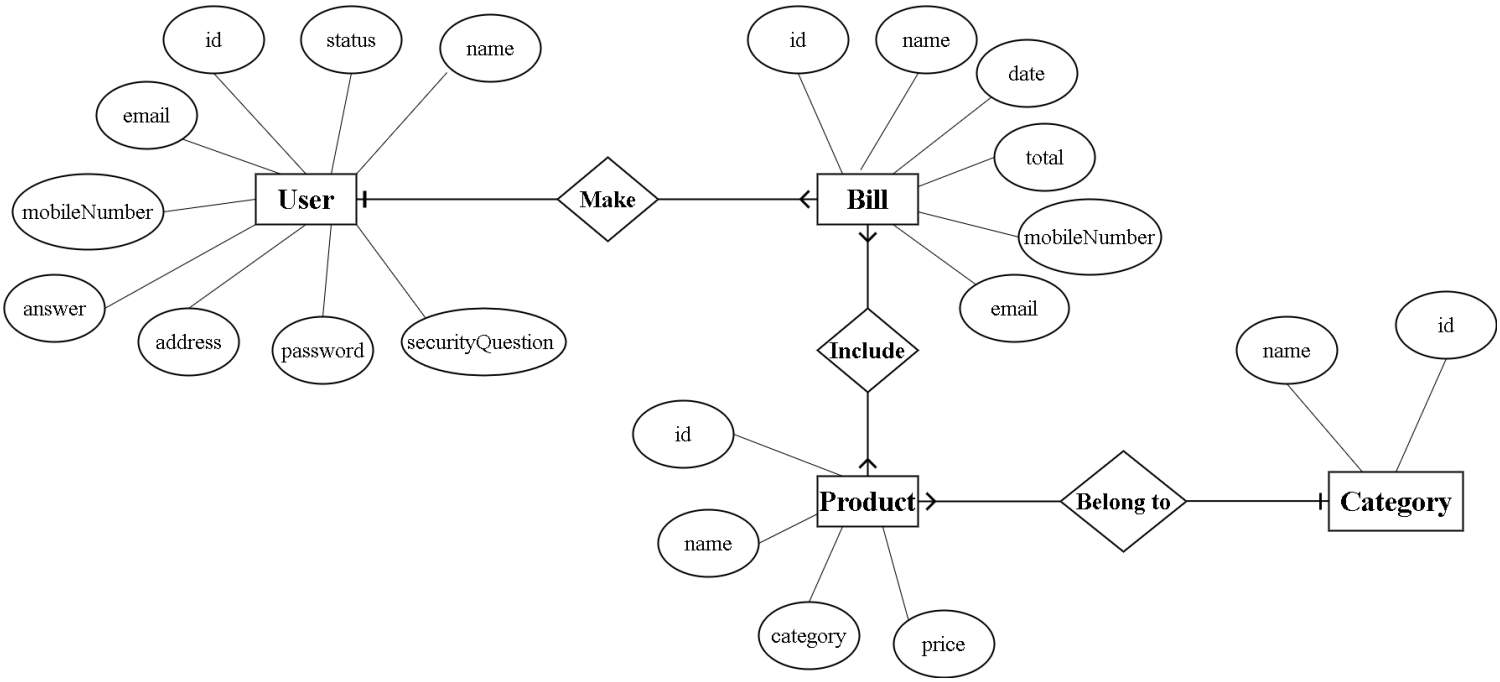


Figure 2. Entity Relationship Diagram

B. Relational Schema

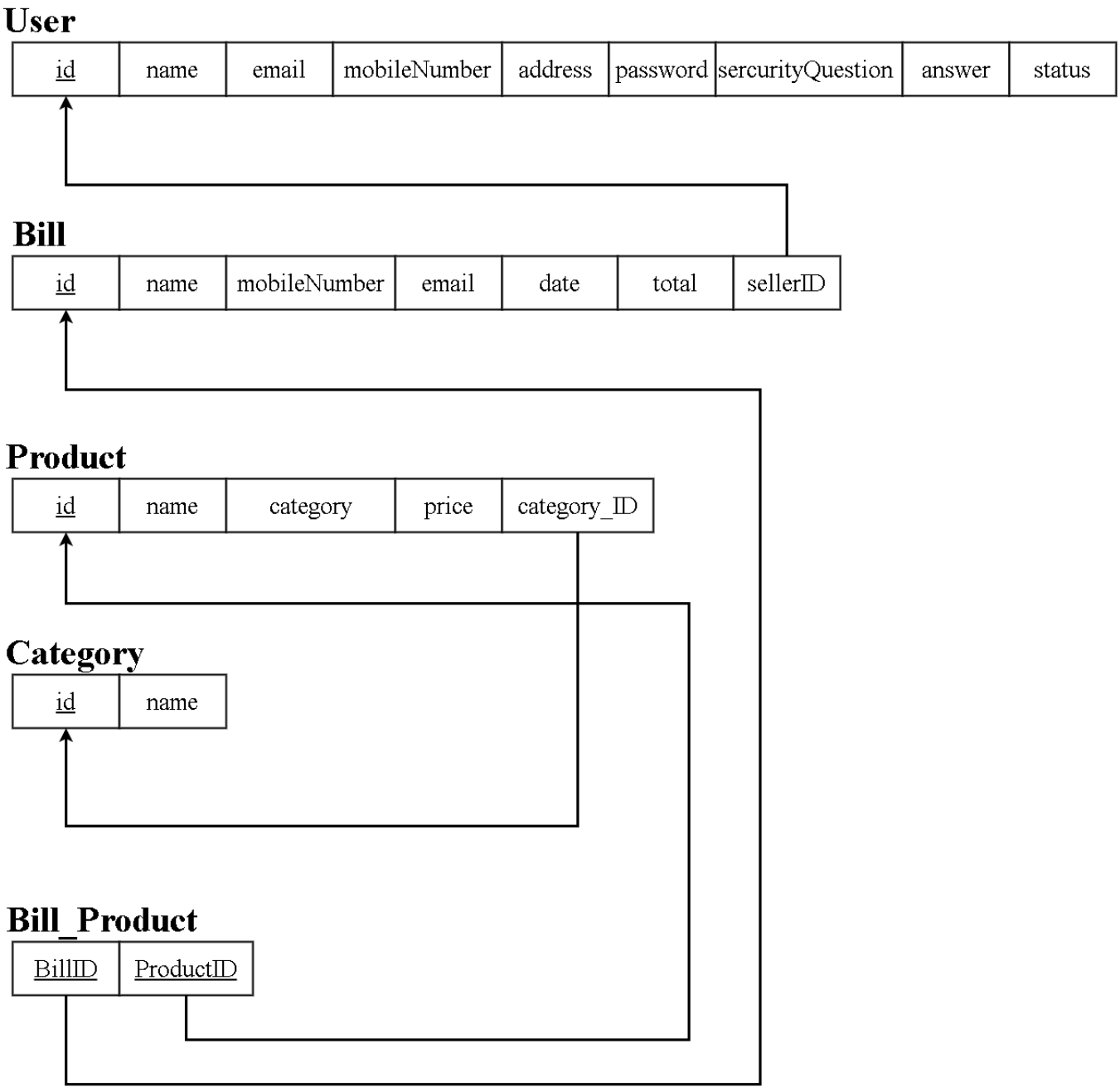


Figure 3. Relational schema

V. PROJECT ANALYSIS

A. Requirement Analysis

1. Requirements:

For this project, our task is to develop a robust management system tailored to the users' needs. We'll be using any language to build the software application, which will be closely integrated with a database management system like MySQL or Microsoft SQL Server. This system should effectively handle user interactions and oversee various operations.

There are some key requirement in this project:

- **Objective and Methodology**: We'll start by clearly defining our project goals, such as making dining experiences smoother, orders faster, and customer service better. We'll choose a development approach based on what suits our project's needs best.
- **Database Design and Implementation**: We'll create database structures tailored to our specific domain, like organizing tables or orders, menus, staff, and customer information. Using SQL, we'll build these structures to ensure they're well-organized and easy to work with.
- **Application Development**: Our focus will be on creating user-friendly interfaces, making it simple for staff to handle orders and customers to navigate menus and reservations in a Restaurant Management System.
- **Database Connectivity**: Ensuring smooth interaction between our application and the database is vital. For instance, in our Restaurant Management System, we'll need to retrieve menu items, creating new categories and adding new products.
- **Functionality and Control**: We'll implement core functionalities such as processing, bill generation, and staff management.
- **Testing and Optimization**: Our system will undergo through testing to ensure reliability and effectiveness. We'll focus on optimizing performance, particularly in areas like order processing speed and database query efficiency.
- **Documentation**: Comprehensive documentation will cover database schema, application architecture, and user guides. This documentation will serve as a valuable resource for restaurant staff and administrators, facilitating efficient system usage and maintenance.

This project will demonstrate our ability to design and implement a functional management system for chosen domains, integrating database management principles to ensure effective data handling and system control.

2. **Objective:**

The aim of this project is to develop a comprehensive Restaurant Management System, covering administrative functionalities, and inventory management. This system will streamline operations, enhance customer service and optimize resource utilization within the restaurant environment.

- **User Management**:
    - Admins: Admins can add, modify and deactivate staff accounts for effective workforce management. Access control ensures staff member' access is restricted based on roles, enhancing data security.
- **Billing**:
    - Staff are responsible for placing orders, generating bills and processing payments promptly, ensuring a smooth dining experience.
- **Inventory Management**:
    - The admin has authority over product catalog updates. Including adding new items, and updating process, to maintain an up-to-date inventory.

These user stories and functional specifications will guide the development process, ensuring that the Restaurant Management System meets the needs of both customers and staff while facilitating efficient management of inventory and resources.

3. **Changes:**

Throughout the project, there were no changes or updates to the requirements initially outlined. The project proceeded smoothly with the established objectives and functional specifications remaining consistent.

B. Approach Analysis

In preparation for meeting project requirements effectively, we've conducted a thorough review of relevant in-class lectures. This focused on topics like Entity-Relationship diagrams,

relational schemas, SQL, and normalization techniques. By assimilating insights from these materials, we aim to construct a database adhering to the principles of the Third Normal Form (3NF), laying a solid foundation for our software application development process.

In terms of research analysis, our approach encompasses three key aspects:

Firstly, we're gathering insights on designing the software application's user interface (UI) by exploring tutorials and projects available on platforms like YouTube and public GitHub repositories. This will provide us with practical examples and best practices to guide our UI design decisions.

Secondly, we're delving into online articles and resources to understand user needs and expectations regarding restaurant management systems. Analyzing customer feedback, industry trends, and expert opinions helps us identify essential features and functionalities our software should incorporate to effectively meet user requirements.

Lastly, we're conducting a comprehensive assessment to determine the primary and secondary functions of our restaurant management software. This involves outlining core functionalities such as table management, menu customization, order processing, inventory management, staff scheduling, and reporting. Additionally, we're identifying supplementary features that enhance user experience and operational efficiency, such as online reservation systems, customer feedback management, and integration with payment gateways.

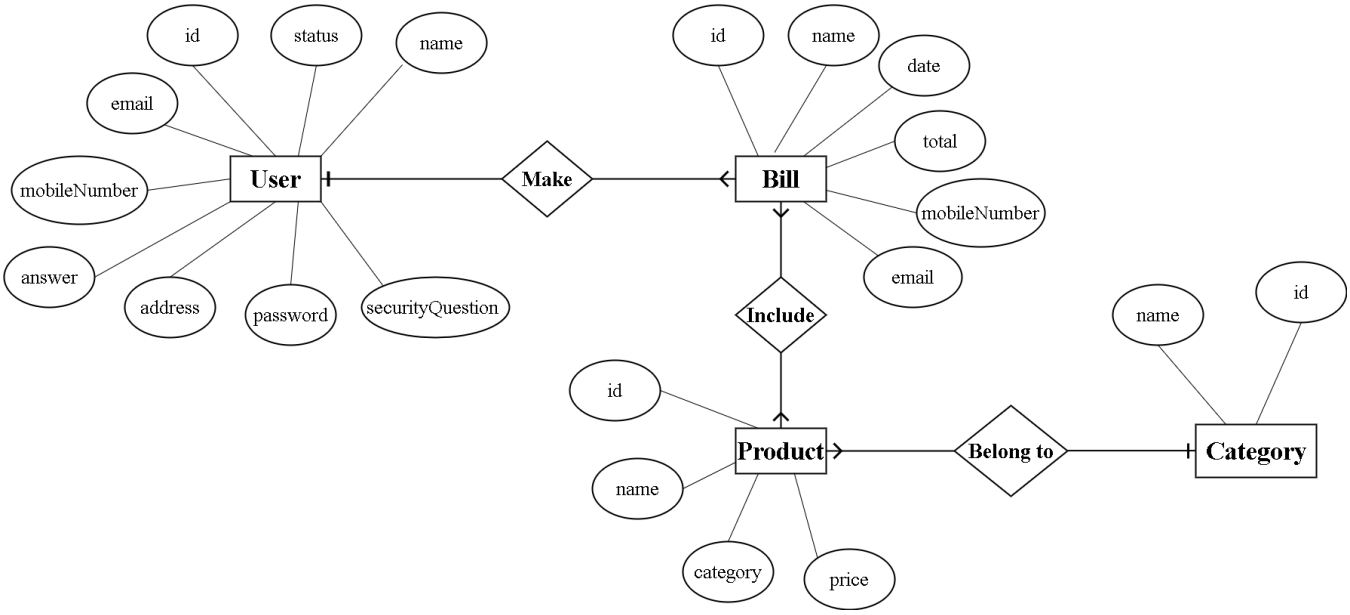
Through this thorough research process, we're gathering valuable insights to inform the design and development of our restaurant management software, ensuring it aligns with user needs and industry standards.

C. System Analysis

1. Database design.

Background: Initially, the design of the restaurant system database established the basic relationships between entities such as User, Bill, Product, and Category. To ensure the database's efficiency and reliability, it was crucial to evaluate and refine the schema using normalization principles up to the third normal form.

From the ERD:



- We can define the relationship between four entities:
    + One-to-many: between “User” and “Bill” as well as between “Product” and “Category”.
    + Many-to-many: between “Bill” and “Product”
- Applying the rules to normalize the database, we will create an associative entity to manage the relationship many-to-many between “Bill” and “Product”. Besides, we also add foreign keys to the “many” side of one-to-many relationships.
- The normalization implementation resulted in the following changes to the ERD:
    + Creation of a new table, "Bill_Product": This table ensures that each pairing of bill and product is unique and properly referenced.
    + Alteration of the "Product" and "Bill" tables with the addition of new foreign keys: "Category_Id" referencing Category (ID) and "SellerID" referencing User (ID), respectively.

2. Database and table creation.

- **Database creation:**

```
create schema rms;

use rms;
```

- **Tables creation:**

+ User table:

```
CREATE TABLE `user` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(200) DEFAULT NULL,
  `email` varchar(200) DEFAULT NULL,
  `mobileNumber` varchar(10) DEFAULT NULL,
  `address` varchar(200) DEFAULT NULL,
  `password` varchar(200) DEFAULT NULL,
  `securityQuestion` varchar(200) DEFAULT NULL,
  `answer` varchar(200) DEFAULT NULL,
  `status` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`),
  UNIQUE KEY `email` (`email`)
)
```

+ Bill table:

```
CREATE TABLE `bill` (
  `id` int NOT NULL,
  `name` varchar(200) DEFAULT NULL,
  `mobileNumber` varchar(200) DEFAULT NULL,
  `email` varchar(200) DEFAULT NULL,
  `date` varchar(50) DEFAULT NULL,
  `total` varchar(200) DEFAULT NULL,
  `SellerID` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `SellerID` (`SellerID`),
  CONSTRAINT `bill_ibfk_1` FOREIGN KEY (`SellerID`) REFERENCES `user` (`id`)
);
```

+ Product table:

```
CREATE TABLE `product` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(200) DEFAULT NULL,
  `category` varchar(200) DEFAULT NULL,
  `price` decimal(10,2) DEFAULT NULL,
  `Category_ID` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`,`category`),
  KEY `Category_ID` (`Category_ID`),
  CONSTRAINT `product_ibfk_1` FOREIGN KEY (`Category_ID`) REFERENCES `category` (`id`)
);
```

+ Category table:

```
CREATE TABLE `category` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ;
```

+ Bill_Product table:

```
CREATE TABLE `bill_product` (
  `BillID` int DEFAULT NULL,
  `ProductID` int DEFAULT NULL,
  KEY `BillID` (`BillID`),
  KEY `ProductID` (`ProductID`),
  CONSTRAINT `bill_product_ibfk_1` FOREIGN KEY (`BillID`) REFERENCES `bill` (`id`),
  CONSTRAINT `bill_product_ibfk_2` FOREIGN KEY (`ProductID`) REFERENCES `product` (`id`)
);
```

3. Database data insertion.

- Insertion of data was performed manually through the system. Datas were verified through testing and validation.
- **Some example code:**
  + Insert a bill into bill table:

```
public static void save(Bill bill){
    String query = "INSERT INTO bill (id, name, mobileNumber, email, date, total, SellerID) VALUES ('" + bill.getId() + "', '"
                    + bill.getName() + "', '"
                    + bill.getMobileNumber() + "', '"
                    + bill.getEmail() + "', '"
                    + bill.getDate() + "', '"
                    + bill.getTotal() + "', '"
                    + bill.getSellerID() + "')";
    DbOperation.setDataOrDelete(query, msg:"Bill Detail Added Successfully");
}
```

+ Insert a category into category table:

```
public static void save(Category category) {
    String query = "insert into category (name) values ('"+category.getName()+"')";
    DbOperation.setDataOrDelete(query, msg:"Category Added Successfully");

}
```

4. Database queries:

- SQL queries were implemented inside the code and developed to retrieve, update, delete, and insert data into the database. Queries were optimized for efficiency and performance
- Some code examples:
    + Delete a category:

```
public static void delete(String id) {
    String query = "delete from category where id = '"+id+"' ";
    DbOperation.setDataOrDelete(query, msg:"Category Deleted Successfully");
}
```

+ Retrieve a category into an object by category name:

```
public static Category getCategoryByname (String name){
    Category category = new Category();
    try{
        ResultSet rs = DbOperation.getData("select id from category where name = '"+name+"'");
        while (rs.next()){
            category.setId(id: rs.getInt(i: 1));
            category.setName(name);
        }
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(parentComponent:null, message:e);
    }
    return category;
}
```

+ Update a product detail:

```
public static void update(Product product) {
    String query = "update product set name = '"+product.getName()
            +"', category = '"+product.getCategory()+"', price = '"
            +product.getPrice()+"', Category_ID = '"
            +product.getCategory_ID()+"' where id = '"
            +product.getId()+"' ";
    DbOperation.setDataOrDelete(query, msg:"Product Update Successfully");
}
```

D. Application Structure

1. Project structure:

- Our project include 4 main folders:
  + Source package: use for containing all functional classes of our project (DataAccessObject, UI, common, main, images, model).
  + Test Package: use for testing the application.
  + Libraries: contain all libraries that are used in the application.
  + Test libraries: use for testing libraries before using.

2. Class structure:

- All of our application classes is contained in the “Source package”:
  + DataAccessObject file: contain all classes that are used for accessing to MySQL server, create tables, execute queries as well as all queries.
  + User Interface(UI) file: this file contains all classes that function the UI.

+ Common file: include the class that is responsible for printing out the bill in pdf form.
+ Main file: the main class for executing the application.
+ Images file: contains all images used in the application.
+ Model file: contain all object classes of the tables in the database.

3. Connection implementation:

- Implemented in the "ConnectionProvider.java":
  + Initialize some connection structure.

```
package DataAccessObject;
import java.sql.*;
//Change both password in getcon and getCon_initial
public class ConnectionProvider {
    static String user = "root";
    static String url = "jdbc:mysql://localhost:3306/rms";
    static String url2 = "jdbc:mysql://localhost:3306";
    static String password = "VLNVLTT112702t18@";
    //static String password = "01102000";
```

  + Get connection after creating schema "rms"

```
    public static Connection getCon() {
        try {
            Connection con = DriverManager.getConnection(url, user, password);
            return con;
        }
        catch (Exception e) {
            return null;
        }
    }
```

  + Get connection before create schema "rms"

```
public static Connection getCon_initial() {
    try {
        Connection con = DriverManager.getConnection(url:url2, user, password);
        return con;
    }
    catch (Exception e) {
        return null;
    }
}
```

4. **GUI design - frames implementation.**
    - All of the GUI is implemented using Java Swing library.
5. **Buttons implementation.**
    - All of the buttons in GUI function through the implementation in Action Listeners methods.

VI. PROJECT DEMONSTRATION

A. Login

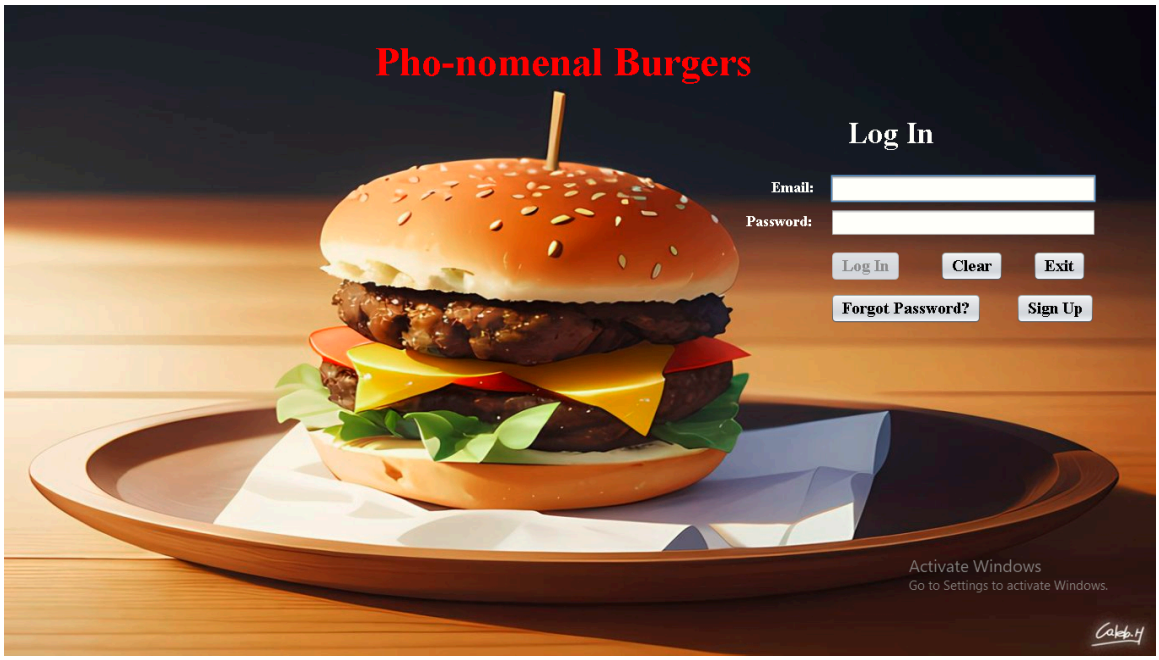


Figure 4. Login page

B. Sign Up

Figure 5. Sign-up page

C. Forgot Password

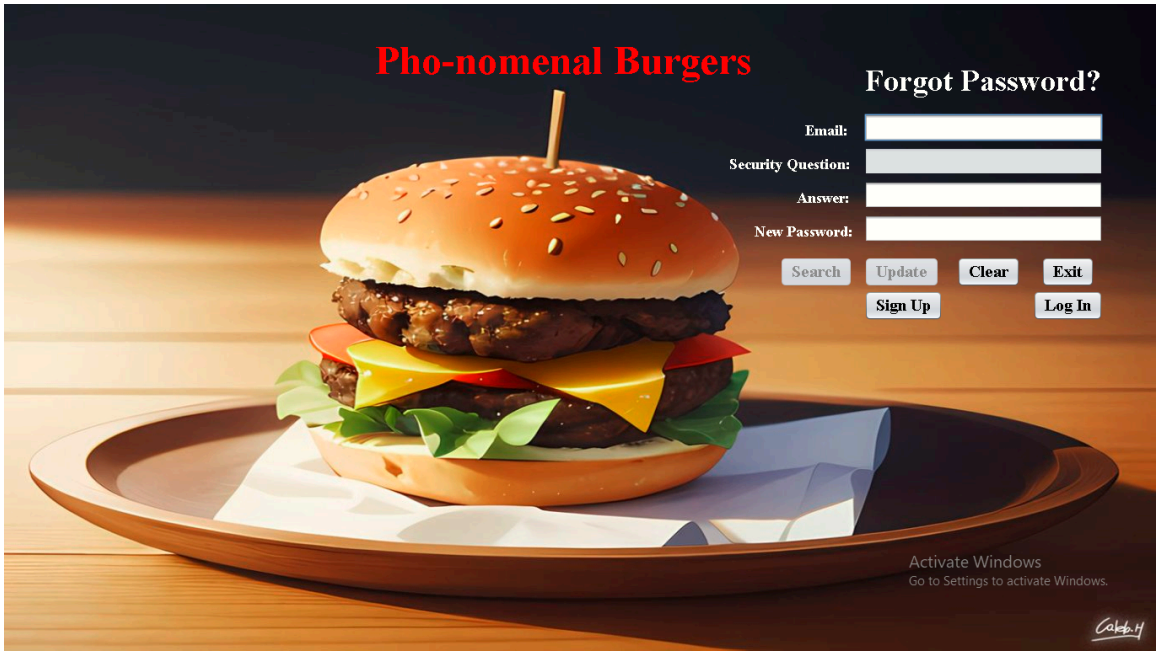


Figure 6. Forget Password page

D. Manage category

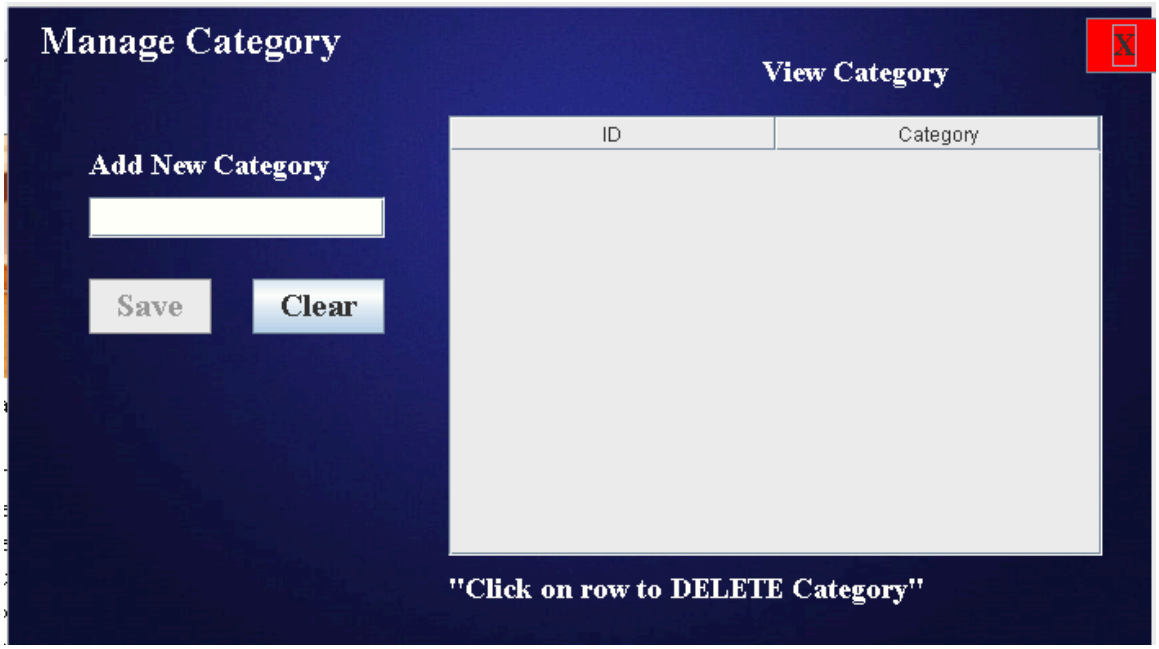


Figure 7. Manage Category page

E. Add product

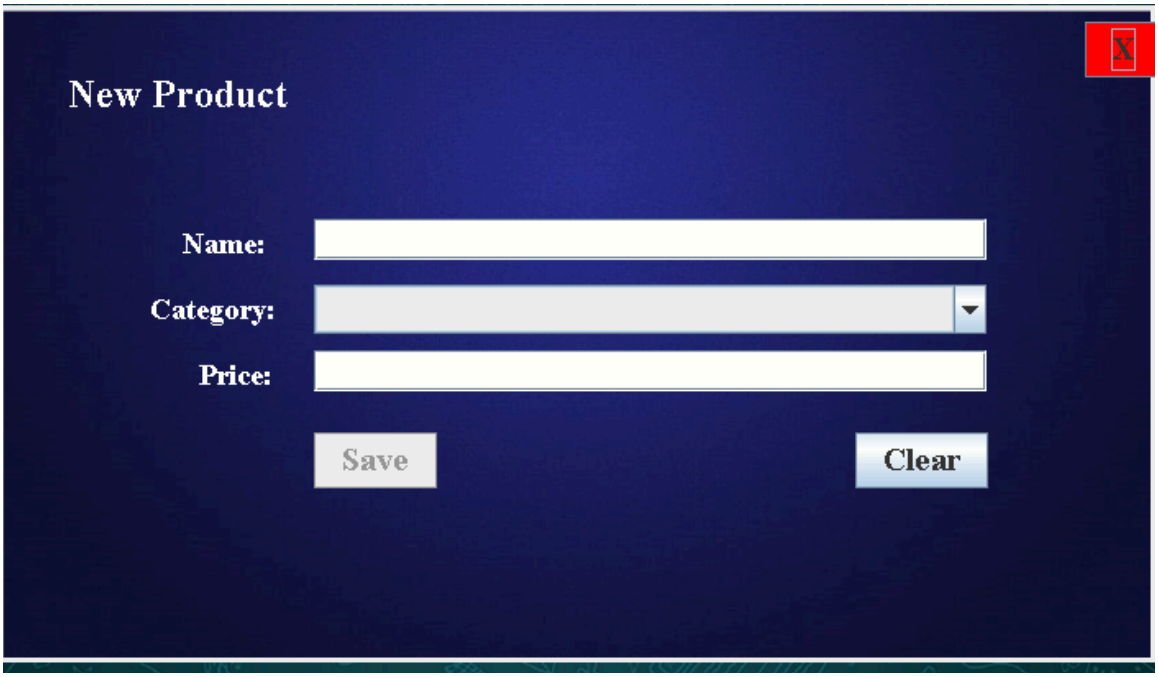


Figure 8. Add product page

F. Manage Product

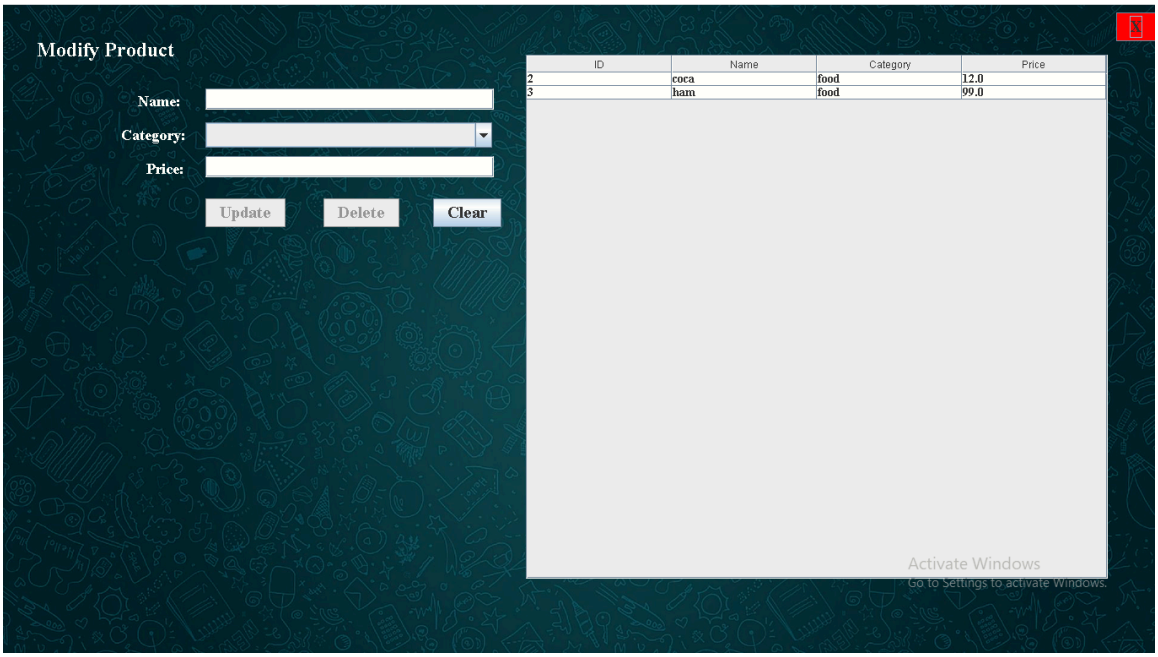


Figure 9. Manage product page

G. Verify user

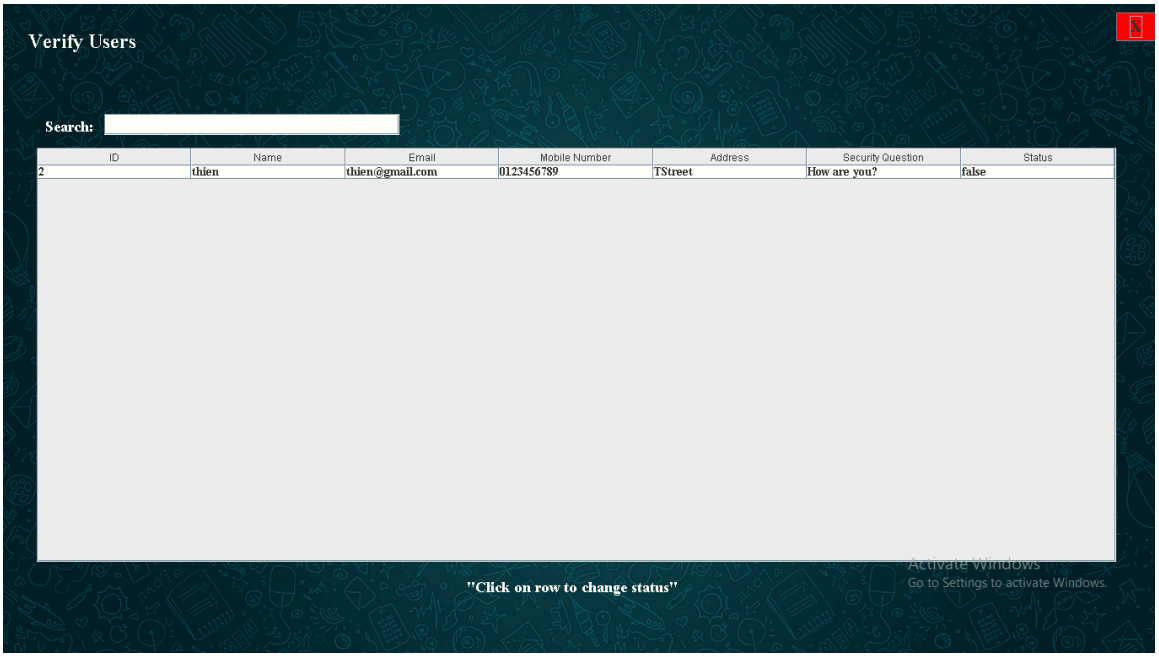


Figure 10. Verify user page

H. Place order

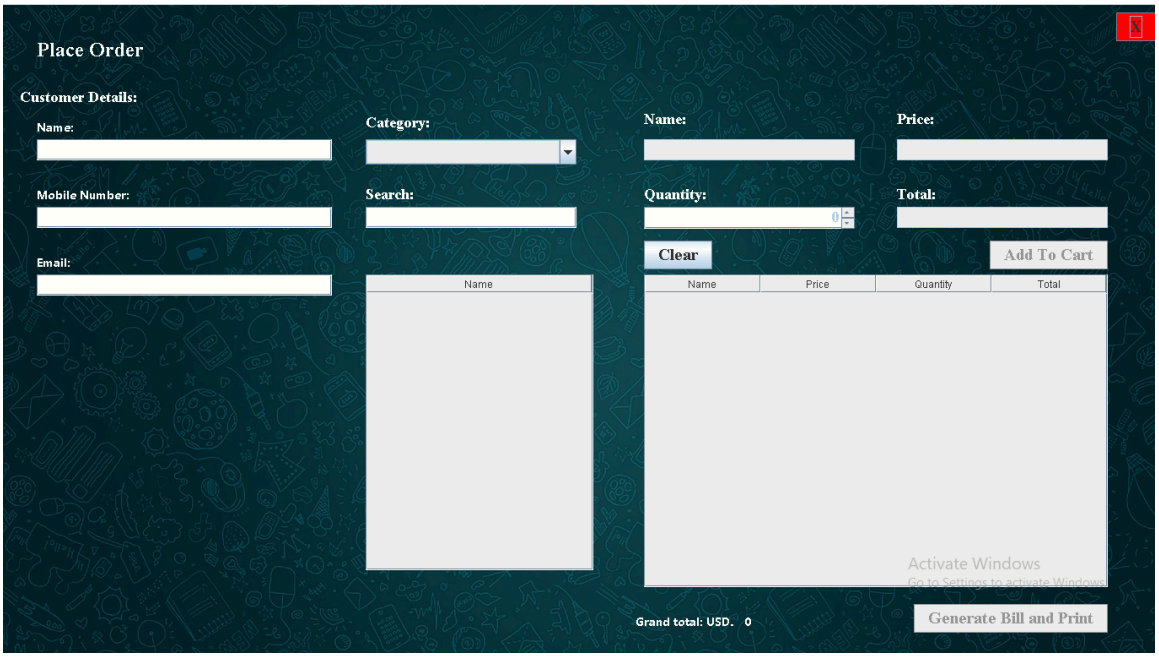


Figure 11. Place order page

I. View generated bill

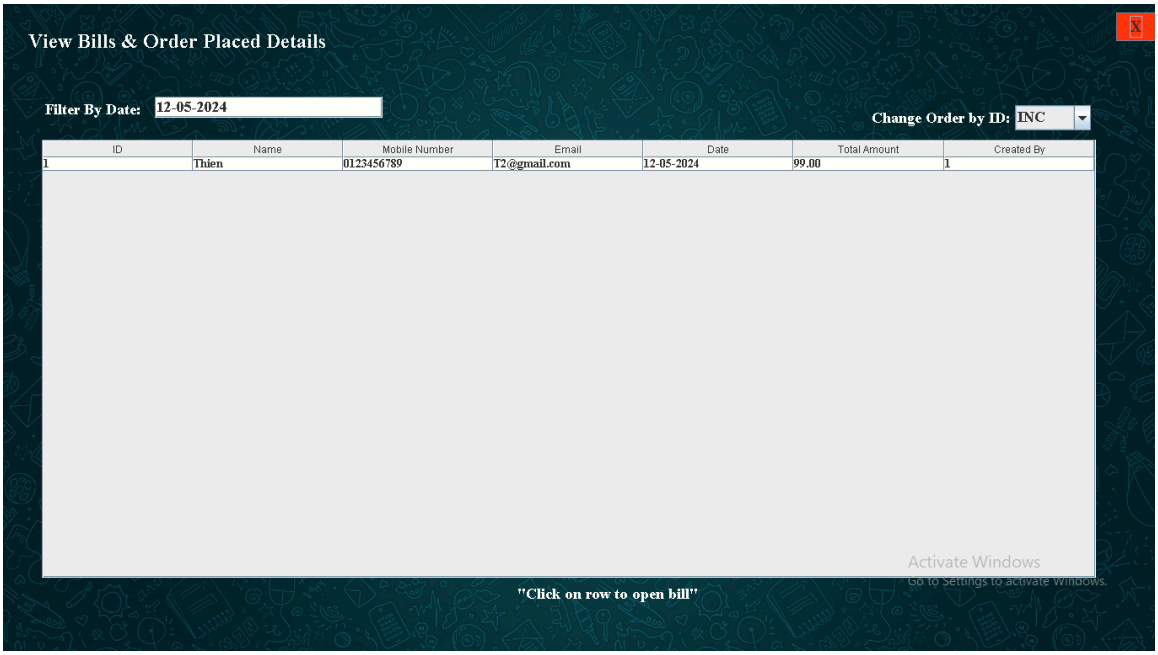


Figure 12. View bill page

J. Change password

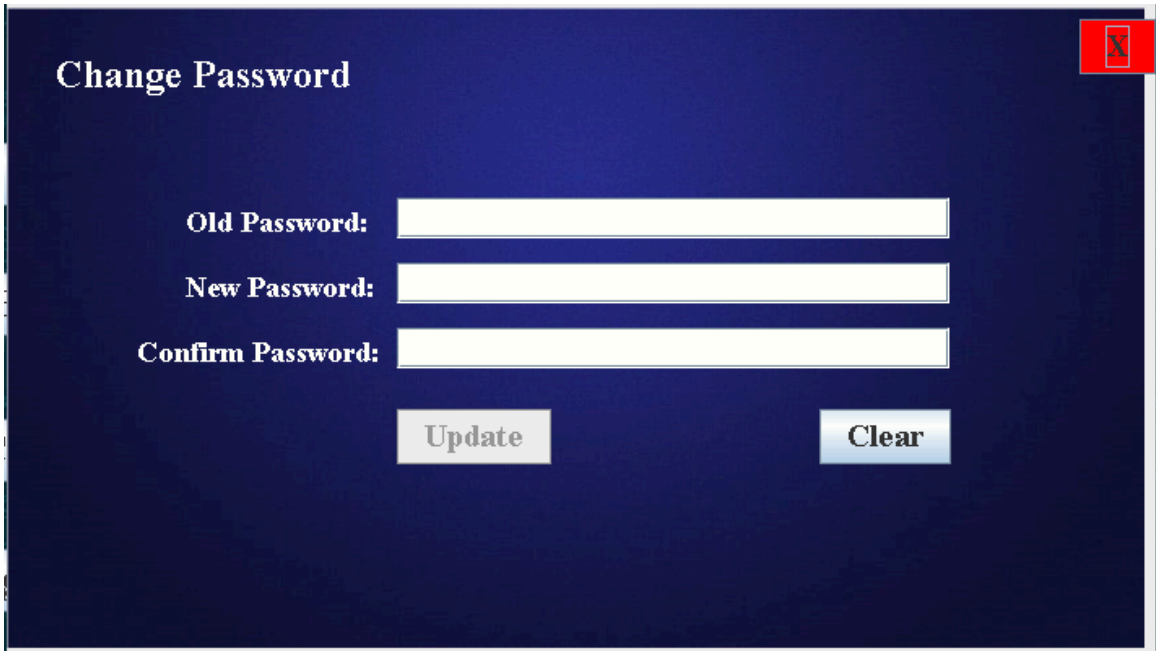


Figure 13. Change password page

K. Change security question

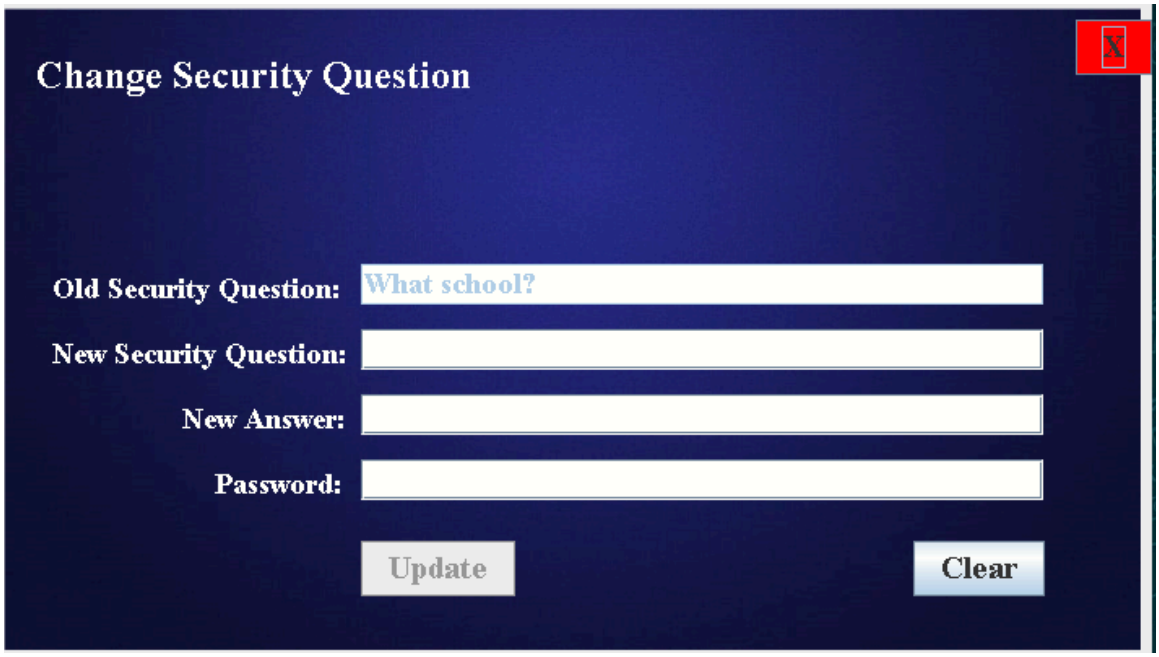


Figure 14. Change security question page

VI. CONCLUSION

Throughout this project, we've successfully met our objectives, enhancing the efficiency and productivity of our restaurant management procedures. By integrating robust administrative features and inventory management systems, we've streamlined our restaurant operations significantly. This improvement is evident in our enhanced customer service, where seamless ordering processes, quick bill processing, and detailed bill printing contribute to a smoother dining experience. Moreover, our system's intuitive interface simplifies restaurant product management, allowing for swift resource modification and providing users with prompt guidance. While our security measures may not be advanced, our adherence to stringent data security protocols ensures the integrity, confidentiality, and compliance of our systems, effectively safeguarding sensitive information and mitigating risks. Overall, the deployment of our comprehensive restaurant management system has led to tangible improvements in operational efficiency and customer service. I extend sincere appreciation to the project team, stakeholders, and partners for their invaluable contributions, as their dedication and collaboration have been instrumental in our success. Looking ahead, I'm eager to continue working together on future projects.

VII. REFERENCES

1. BoostMyTool. (2021, December 15). Create your First Java Project with Netbeans 12.6 | How to Create JFrame Forms using Window Builder [Video]. Retrieved from YouTube: https://www.youtube.com/watch?v=YykJGqvpFnU
2. Tyri0n. (n.d.). Online-Banking-App/Online Banking System/OBS/src/banking at master · tyri0n11/Online-Banking-App. Retrieved from GitHub: https://github.com/tyri0n11/Online-Banking-App/tree/master/Online%20Banking%20System/OBS/src/banking?fbclid=IwAR36uafzE3l3aY11eKr69YuHwJyUUuI7qU-UQRF-M-S8gZDZXdpI136kYIk
3. Java Database Connectivity with MySQL - javatpoint. (n.d.). Retrieved from www.javatpoint.com: https://www.javatpoint.com/example-to-connect-to-the-mysql-database?fbclid=IwAR2Ez_MfPpnNmmaxx8G9WyW0g5L9EHbmXcamJW3mVq3llvSy7LzZa_v3d74
4. 8 essential restaurant management system. (n.d.). Retrieved from Oracle: https://www.oracle.com/food-beverage/restaurant-pos-systems/restaurant-management-system-features/

VIII. ACKNOWLEDGEMENTS

Lecturer: Mr. Wojtkiewi Krystian - krystianwojtkiewicz@mp.hcmiu.edu.vn

Lab instructor: Thai Trung Tin - tintt22@mp.hcmiu.edu.vn