VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AN ENGINEERING

A blue and white logo

Description automatically generated

**PRINCIPLES OF DATABASE MANAGEMENT IT079IU**

Instructor: Assoc. Prof. Nguyen Thi Thuy Loan

**PROPOSAL REPORT**

**Invoice Billing Database**

**By Group 03 – Member List**

1. Lê Phan Hồng Huế ITDSIU23032 (Team Leader)
2. Nguyễn Thị Diễm Quỳnh ITDSIU23021 (Team Member)
3. Lê Thị Ngọc Tuyền ITDSIU23028 (Team Member)
4. Đặng Nhân Hoà ITITIU21205 (Team Member)
5. Võ Phúc Khang ITCSIU23012 (Team Member)

**Table Of Contents**

[Chapter 1: INTRODUCTION 5](#_Toc184511537)

[1. ABSTRACT 5](#_Toc184511538)

[2. SYSTEM OVERVIEW 5](#_Toc184511539)

[3. OBJECTIVES 6](#_Toc184511540)

[4. TECHIQUE AND TOOL USED 6](#_Toc184511541)

[Chapter 2: TASK TIMELINE AND DIVISION 7](#_Toc184511542)

[1. CONTRIBUTION 7](#_Toc184511543)

[2. PROJECT TIMELINE & TASK DIVISION 7](#_Toc184511544)

[Chapter 3: PROJECT ANALYSIS 8](#_Toc184511545)

[1. REQUIREMENT 8](#_Toc184511546)

[2. APPROACH ANALYSIS 10](#_Toc184511547)

[Chapter 4: DATABASE DESIGN 10](#_Toc184511548)

[1. ENTITY AND RELATIONSHIP 10](#_Toc184511549)

[2. ENTITY RELATIONSHIP DIAGRAM 11](#_Toc184511550)

[3. SYSTEM ANALYSIS 11](#_Toc184511551)

[**3.1.** **Database Design** 11](#_Toc184511552)

[**3.2** **Database and Table Creation** 12](#_Toc184511553)

[**3.2.1** **Database Creation** 12](#_Toc184511554)

[**3.2.2** **Tables Creation** 12](#_Toc184511555)

[**3.2** **Database Data Insertion** 16](#_Toc184511556)

[**3.3** **Database Normalization** 19](#_Toc184511557)

[**3.4** **Database Queries** 22](#_Toc184511558)

[Chapter 5: APPLICATION PROJECT STRUCTURE 27](#_Toc184511559)

[1. PROJECT STRUCTUR 27](#_Toc184511560)

[2. SQL CONNECTION 29](#_Toc184511561)

[3. PROJECT DEMO 31](#_Toc184511562)

[Chapter 6: CONCLUSION 39](#_Toc184511563)

[1. ACHIEVE GOALS 39](#_Toc184511564)

[2. FUTURE WORK 39](#_Toc184511565)

[3. SYSTEM ANALYSIS 40](#_Toc184511566)

[4. REFERENCES 40](#_Toc184511567)

**LIST OF FIGURES**

[**Figure 1: Entity Relationship Diagram** 11](#_Toc184502492)

[**Figure 2: Entity Relationship Diagram using SQL Server** 11](#_Toc184502493)

[**Figure 3 Project’s server structure** 28](#_Toc184502494)

[**Figure 4: SQL structure** 29](#_Toc184502495)

[**Figure 5 SQL Connection** 30](#_Toc184502496)

[**Figure 6: Home Page** 31](#_Toc184502497)

[**Figure 7: Add New Customer** 32](#_Toc184502498)

[**Figure 8: Successfully add new customer** 32](#_Toc184502499)

[**Figure 9: Update customer** 33](#_Toc184502500)

[**Figure 10: Update customer successfully** 33](#_Toc184502501)

[**Figure 11: Delete customer** 34](#_Toc184502502)

[**Figure 12: Print out the detail** 34](#_Toc184502503)

[**Figure 13: Add product to the bill** 35](#_Toc184502504)

[**Figure 14: The bill format** 36](#_Toc184502505)

[**Figure 15: Search product’s information** 37](#_Toc184502506)

[**Figure 16: Show detail product** 38](#_Toc184502507)

[**Figure 17: Add product** 39](#_Toc184502508)

**LIST OF TABLES**

[**Table 1: Individual responsibility and contribution** 7](#_Toc183864571)

[**Table 2: Sprint planning and task division for individuals and teams** 7](#_Toc183864572)

[**Table 3: Entities and Attributes** 12](#_Toc183864573)

[**Table 4: Normal form achievement** 19](#_Toc183864574)

[**Table 5: SQL keywords** 22](#_Toc183864575)

[**Table 6: SQL Queries** 24](#_Toc183864576)

# **Chapter 1: INTRODUCTION**

## **ABSTRACT**

The Invoice Billing Database project focuses on developing a sophisticated system tailored for shopping complexes and retail environments, including generating, tracking, and managing invoices for products sold or services rendered. This database effectively streamlines billing processes by maintaining an organized price list and implementing a keyword recognition feature that enhances user interaction. It also maintains detailed information about contacts, products, and invoice items to facilitate accurate billing and customer communication.

It typically includes several key features:

• Product Information

• Customer Data

• Invoice Records

• Price List

• Keyword Recognition

• Payment Information

By automating and simplifying invoicing procedures, this project aims to improve operational efficiency, reduce errors, and elevate customer satisfaction and will cover the key features of a typical Invoice Billing Database using MySQL.Thus create a manageable shopping platform for customers and staff.

## **SYSTEM OVERVIEW**

The Invoice Billing Database is a core component in financial management systems, designed to streamline billing and payment processes for businesses. It efficiently handles tasks such as invoice generation, tracking payments, and managing client accounts. Modern systems incorporate automation to reduce manual errors, improve accuracy, and save time.

Beyond basic billing functions, advanced databases offer features like real-time updates, detailed reporting, and analytics to support financial decision-making. Integration with accounting tools and payment gateways ensures seamless transactions and efficient reconciliation of financial records.

A focus on user accessibility is integral to the system, ensuring an inclusive interface that caters to diverse user needs. Adopting secure and scalable designs, the database supports businesses of varying sizes, ensuring data integrity and compliance with financial regulations. Additionally, predictive analytics can be utilized to forecast cash flows, detect overdue payments, and enhance overall financial health.

## **OBJECTIVES**

**Design the database to suit the requirements of Normalization BC Normal Form**

● Connect the front-end interface (application) to the back-end database

● Develop the functions such as login, account creation,profile edition, content distribution, interactive materials using complex queries.

● Take security measures in the database to ensure robust privacy for the users' data .

## **TECHIQUE AND TOOL USED**

* ERDplus to build Entity-Relationship Diagrams.
* GitHub to easily collaborate on the project.
* Java Swing
* Visual Studio Code
* NetBean, Git is used to streamline the styling process in app development, providing a utility-first approach that allows for rapid prototyping, responsive design, and consistent styling across projects..

# **Chapter 2: TASK TIMELINE AND DIVISION**

## **CONTRIBUTION**

**Table 1: Individual responsibility and contribution**

|  |  |  |
| --- | --- | --- |
| **Full Name** | **Role** | **Contribution** |
| Lê Phan Hồng Huế | Leader | 20% |
| Nguyễn Thị Diễm Quỳnh | Member | 20% |
| Lê Thị Ngọc Tuyền | Member | 20% |
| Đặng Nhân Hòa | Member | 20% |
| Võ Phúc Khang | Member | 20% |

## **PROJECT TIMELINE & TASK DIVISION**

**Table 2: Sprint planning and task division for individuals and teams**

|  |  |  |  |
| --- | --- | --- | --- |
| **WEEK** | **STAGE** | **TASK** | **ASSIGNED TO** |
| 1 + 2 | **PLANNING**  **AND PREPARATION** | Researching about the project topic, analyzing the project requirements | All members |
| Determining research scopes and the objective of the project | Huế |
| Constructing a timeline for the project | All members |
| Determining the DBMS and tools for the project | All members |
| **Write Project Proposal** | All members |
| 3 + 4 | **DESIGNING ER DIAGRAM AND SCHEMA DIAGRAM** | Analyzing the database requirements | All members |
| Design ERD, relational model, schema diagram, and class diagram | Quỳnh, Hòa |
| 5 | **CREATING**  **THE DATABASE** | Creating the database, setting tables and relationships | Huế, Tuyền |
| Applying queries in SQL | All members |
| Connecting to database SQL server via Java | All members |
| **Finishing the Midterm Report** | Huế, Tuyền, Quỳnh |
| 6 | **CREATE APPLICATION INTERFACE** | Developing application interface | Huế, Khang |
| Developing Home Page,Custiomrt,Product Interface | Khang |
| Developing Billing Interface | Hòa |
| 7+9 | **TESTING**  **AND COMPLETING THE REPORT** | Fixing bugs on Java | Khang |
| Testing functions of application and optimizing the application performance | Huế, Quỳnh, Tuyền |
| **Writing Final Report** | Tuyền,Huế |
| Making slides for the presentation | Huế, Tuyền Quỳnh |
| Rehearsing the presentation and reviewing the overall project | All memners |
| 11 | **PRESENTATION** |  |  |

# **Chapter 3: PROJECT ANALYSIS**

## **REQUIREMENT**

**● Product Information Management:**

Store detailed product information, including names, product codes, and prices.

Integrate a quick search mechanism using keyword recognition to help staff retrieve product information efficiently.

**● Customer Data Management:**

Maintain customer details such as names, addresses, emails, and phone numbers for communication and support purposes.

**● Invoice Management System:**

Create, track, and manage invoices for sold products or services rendered.

Allow editing and querying of invoice histories to assist in financial management.

**● Price List:**

Provide an updated price list, including discounts and promotional offers.

Automatically apply the correct pricing table when generating invoices.

**● Keyword Recognition Feature:**

Develop a keyword recognition system to expedite information retrieval and enhance the user experience.

**● Payment Information:**

Store detailed payment transaction data, including payment methods

Support diverse payment methods (cash, card, or bank transfer).

**● Efficient User Interaction:**

Design a user-friendly interface to ensure easy navigation for staff and customers.

Provide intuitive navigation to minimize transaction processing time.

**● Database Design:**

Comply with Boyce-Codd Normal Form (BCNF) to ensure data integrity and eliminate redundancy.

Clearly define entities like Products, Customers, Invoices, and Payments, as well as the relationships between them.

**● Query and Performance Optimization:**

Optimize SQL queries for fast data retrieval.

Ensure system stability while handling large datasets.

**● Notification System:**

Send notifications to customers via email or SMS for new invoices or successful payments.

**● Personalized User Experience:**

Incorporate login functions for individual staff or customers with specific permissions.

Personalize the interface and content based on user roles (staff or customers).

**● System Objectives:**

Automate the invoicing process to reduce errors and enhance operational efficiency.

Create a convenient platform for shopping and billing, benefiting both customers and staff.

## **APPROACH ANALYSIS**

* **Netbean**

**URL:** [**https://netbeans.apache.org/tutorial/main/kb/**](https://netbeans.apache.org/tutorial/main/kb/)

* **ERDPlus**

**URL:** [**https://erdplus.com/**](https://erdplus.com/)

* **MySQL**

**URL:** [**https://www.mysql.com/**](https://www.mysql.com/)

# **Chapter 4: DATABASE DESIGN**

## **ENTITY AND RELATIONSHIP**

**Customer and Invoice:**

* 1 customer can have multiple invoices.
* One invoice can only belong to one customer.
* Invoice is **mandatory** because the invoice must be linked to a customer. However, a customer can exist in the database without having any invoices associated with them so customer is **optional**.

=> The relationship between customer and invoice is a **one-to-many relationship.**

**Product and Price\_List**

* 1 product can have many prices.
* 1 price in the price list applies to 1 specific product.
* Price\_List is a **weak entity** that cannot exist without a corresponding Product. It stores pricing details for specific products.
* Price\_List is **optional** because a product can exist without a corresponding price, but each price must be linked to a specific product.

=> The relationship between product and price list is a **one-to-many relationship.**

**Product and Invoice** :

* 1 invoice can contain many products  and at least 1 product.
* 1 product can belong to many invoices
* Invoice is mandatory because an invoice can not exist without any products.

=> The relationship between product and invoice is a **many-to-many relationship**

**Invoice and Payment :**

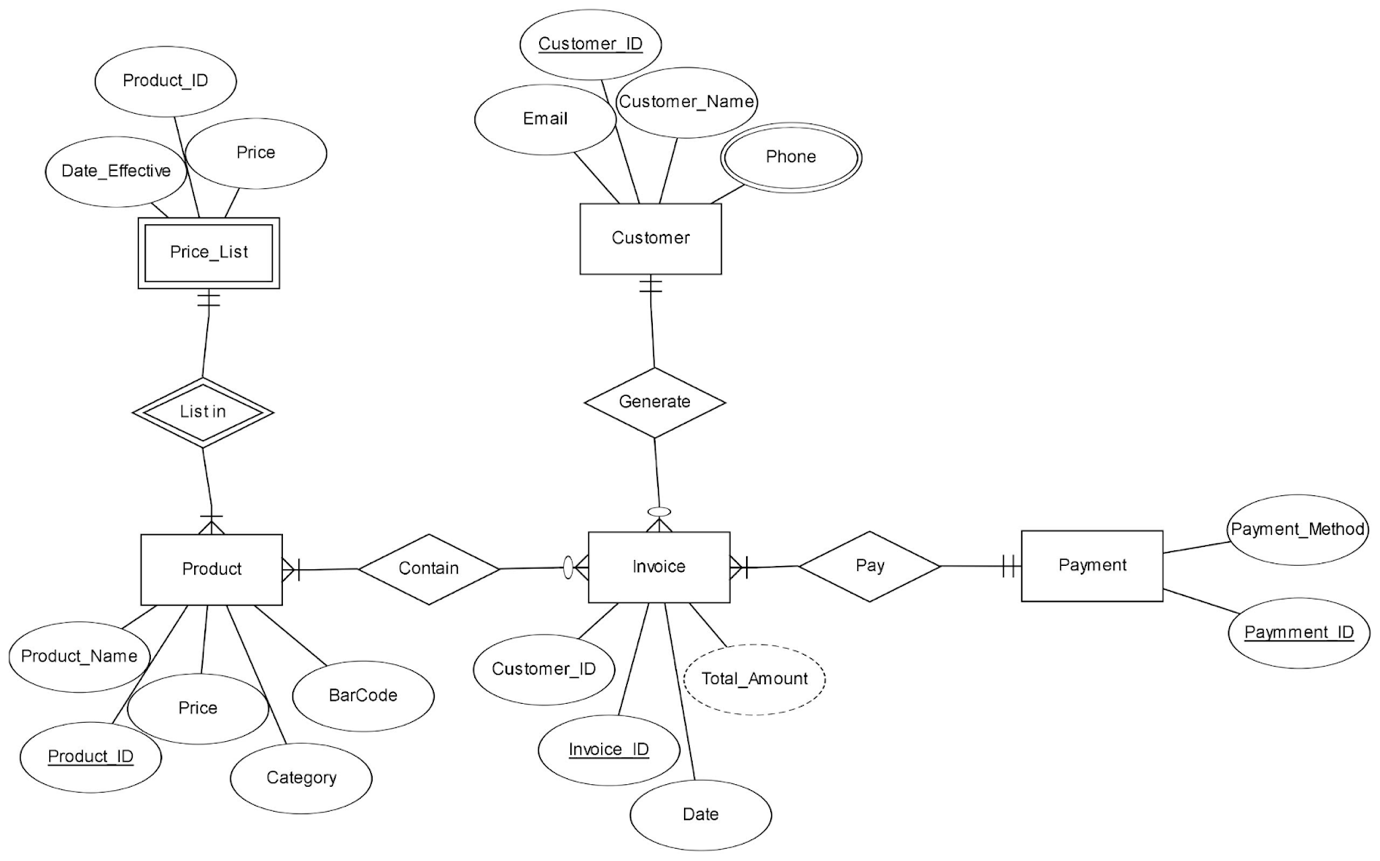
* 1 Invoice can have 1 Payment associated with it
* 1 Payment applies to 1 specific Invoice**.**
* Payment is mandatory because every invoice must have an associated payment record.

=> The relationship between Payment and Invoice  is a **one-to-many relationship.**

## 

## **ENTITY RELATIONSHIP DIAGRAM**

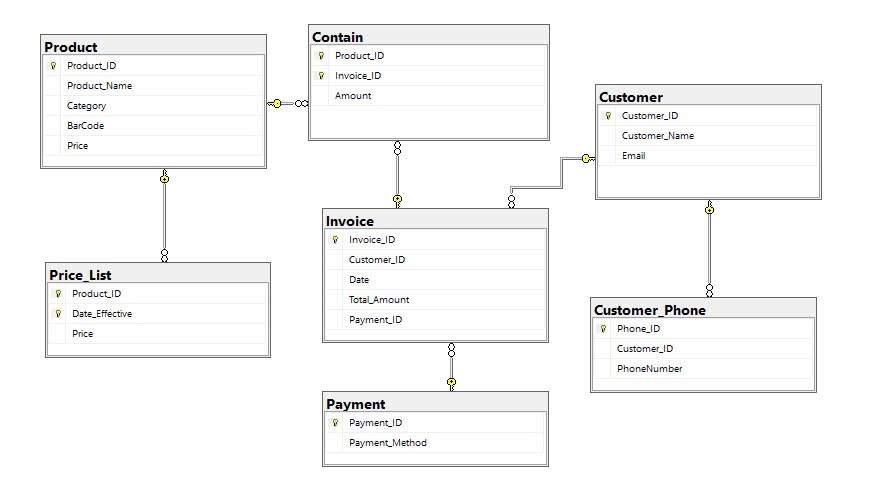
**Figure 1: Entity Relationship Diagram**

****

## **SYSTEM ANALYSIS**

### **Database Design**

**Figure 2: Entity Relationship Diagram using SQL Server**

****

* 1. **Database and Table Creation** 
     1. **Database Creation**

**A close-up of a white background

Description automatically generated**

* + 1. **Tables Creation**

The current version of the Entity- Relationship Diagram (**Figure 1**) has the following entities:

**Table 3: Entities and Attributes**

|  |  |
| --- | --- |
| **Table** | **Attribute** |
| **Product** | Product\_ID INT AUTO\_INCREMENT PRIMARY KEY,  Product\_Name VARCHAR(255) NOT NULL,  Category VARCHAR(100),  BarCode VARCHAR(100),  Price DECIMAL(10, 2) NOT NULL |
| **Customer** | Customer\_ID INT AUTO\_INCREMENT PRIMARY KEY,   Customer\_Name VARCHAR(255) NOT NULL,   Email VARCHAR(255) NOT NULL |
| **Customer\_Phone** | Phone\_ID INT AUTO\_INCREMENT PRIMARY KEY,  Customer\_ID INT,  PhoneNumber VARCHAR(20),  FOREIGN KEY (Customer\_ID) REFERENCES Customer(Customer\_ID) ON DELETE CASCADE |
| **Contain** | Product\_ID INT,  Invoice\_ID INT,  Amount INT,  PRIMARY KEY (Product\_ID, Invoice\_ID),  FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID),  FOREIGN KEY (Invoice\_ID) REFERENCES Invoice(Invoice\_ID) |
| **Price\_List** | Product\_ID INT NOT NULL,  Date\_Effective DATE NOT NULL,  Price DECIMAL(10, 2) NOT NULL,  PRIMARY KEY (Product\_ID, Date\_Effective),  FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID) |
| **Invoice** | Invoice\_ID INT AUTO\_INCREMENT PRIMARY KEY,  Customer\_ID INT,  Date DATE NOT NULL,  Total\_Amount DECIMAL(10, 2) NOT NULL,  Payment\_ID INT,  FOREIGN KEY (Customer\_ID) REFERENCES Customer(Customer\_ID) ON DELETE CASCADE,  FOREIGN KEY (Payment\_ID) REFERENCES Payment(Payment\_ID)  ALTER TABLE Invoice AUTO\_INCREMENT = 1; |
| **Payment** | Payment\_ID INT PRIMARY KEY,  Payment\_Method VARCHAR(50) |

**Create Tables:**

**Table Customer**

**A close-up of a computer code

Description automatically generated**

**Table Product**

**A screen shot of a computer code

Description automatically generated**

**Table Customer\_Phone**

**A screenshot of a computer code

Description automatically generated**

**Table Contain**

**A screenshot of a computer code

Description automatically generated**

**Table Price\_List**

**A computer screen shot of a code

Description automatically generated**

**Table Invoice**

**A screen shot of a computer code

Description automatically generated**

**Table Payment**

**A close-up of a computer code

Description automatically generated**

* 1. **Database Data Insertion**

**Table Customer Data**

**A screenshot of a computer

Description automatically generated**

**Table Product Data**

**A screenshot of a computer code

Description automatically generated**

**Table Invoice Data**

**A screenshot of a computer screen

Description automatically generated**

**Table Price\_List Data A screenshot of a computer program

Description automatically generated**

**Table Contain Data**

**A screenshot of a computer

Description automatically generated**

**Table Customer\_Phone Data**

**A screenshot of a computer

Description automatically generated**

**Table Payment Data**

**A screen shot of a computer

Description automatically generated**

* 1. **Database Normalization**

**Table 4: Normal form achievement**

|  |  |
| --- | --- |
| **Normal Form** | **Description** |
| **1 N.F** | **No duplicated row or column**  **Each column has only atomic values** |
| **2 N.F** | **Has no partial dependency, all non-key attributes are fully dependent on a primary key.** |
| **3 N.F** | **Have no transitive partial dependency** |
| **BC N.F** | **Every non-trivial functional dependency in the database depends on a candidate key.** |

**PRODUCT Table**

**1NF: Satisfied because:**

* **All attributes contain atomic values.**
* **There are no repeating groups.**

**2NF: Satisfied because:**

* **The table has a single-column primary key (Product\_ID).**
* **All non-key attributes (Product\_Name, Category, BarCode, Price) are fully functionally dependent on Product\_ID.**

**3NF: Satisfied because:**

* **There are no transitive dependencies. Every non-key attribute is directly dependent on the primary key.**

**BCNF: Satisfied because:**

* **The primary key Product\_ID is the only determinant for all other attributes.**

**4NF: Satisfied because:**

* **There are no multivalued dependencies (e.g., a single product cannot have multiple barcodes or categories in this schema).**

**CUSTOMER Table**

**1NF: Satisfied because:**

* **All attributes contain atomic values.**
* **There are no repeating groups or multivalued attributes.**

**2NF: Satisfied because:**

* **The primary key is Customer\_ID.**
* **All non-key attributes (Customer\_Name, Email) are fully functionally dependent on Customer\_ID.**

**3NF: Satisfied because:**

* **There are no transitive dependencies.**

**BCNF: Satisfied because:**

* **The primary key Customer\_ID is the only determinant for all other attributes.**

**4NF: Satisfied because:**

* **There are no multivalued dependencies (e.g., a single customer cannot have multiple emails in this schema).**

**CUSTOMER\_PHONE Table**

**1NF: Satisfied because:**

* **All attributes contain atomic values.**
* **There are no repeating groups or multivalued attributes.**

**2NF: Satisfied because:**

* **The primary key is Phone\_ID.**
* **All non-key attributes (Customer\_ID, PhoneNumber) are fully functionally dependent on Phone\_ID.**

**3NF: Satisfied because:**

* **There are no transitive dependencies.**

**BCNF: Satisfied because:**

* **The primary key Phone\_ID is the only determinant for Customer\_ID and PhoneNumber.**

**4NF: Satisfied because:**

* **There are no multivalued dependencies (e.g., a single phone record is linked to one customer and one phone number).**

**CONTAIN Table**

**1NF: Satisfied because:**

* **All attributes contain atomic values.**
* **The composite primary key ensures no repeating groups.**

**2NF: Satisfied because:**

* **All non-key attributes (if any) would depend on the entire composite primary key (Product\_ID, Invoice\_ID).**

**3NF: Satisfied because:**

* **There are no transitive dependencies in this relationship table.**

**BCNF: Satisfied because:**

* **The composite primary key is the only determinant.**

**4NF: Satisfied because:**

* **There are no multivalued dependencies in this table. Each Product\_ID and Invoice\_ID combination is unique.**

**PRICE\_LIST Table**

**1NF: Satisfied because:**

* **All attributes contain atomic values.**
* **There are no repeating groups.**

**2NF: Satisfied because:**

* **The primary key PriceList\_ID fully determines all other attributes.**

**3NF: Satisfied because:**

* **There are no transitive dependencies (e.g., Price and Date\_Effective are directly determined by PriceList\_ID).**

**BCNF: Satisfied because:**

* **The primary key PriceList\_ID is the only determinant.**

**4NF: Satisfied because:**

* **There are no multivalued dependencies (e.g., each product has a single price for a specific date).**

**INVOICE Table**

**1NF: Satisfied because:**

* + **All attributes contain atomic values.**
  + **There are no repeating groups.**

**2NF: Satisfied because:**

* + **The primary key Invoice\_ID fully determines all other attributes.**

**3NF: Satisfied because:**

* + **There are no transitive dependencies.**

**BCNF: Satisfied because:**

* + **The primary key Invoice\_ID is the only determinant.**

**4NF: Satisfied because:**

* + **There are no multivalued dependencies (e.g., an invoice is linked to a single customer and has one total amount).**

**PAYMENT Table**

**1NF: Satisfied because:**

* + **All attributes contain atomic values.**
  + **There are no repeating groups.**

**2NF: Satisfied because:**

* + **The primary key Payment\_ID fully determines all other attributes.**

**3NF: Satisfied because:**

* + **There are no transitive dependencies.**

**BCNF: Satisfied because:**

* + **The primary key Payment\_ID is the only determinant.**

**4NF: Satisfied because:**

* + **There are no multivalued dependencies (e.g., a payment cannot have multiple amounts or methods).**
  1. **Database Queries** 
     1. **Queries analysis**

**Table 5: SQL keywords**

| **Keyword** | **Function** |
| --- | --- |
| **SELECT** | **Retrieves** data from one or more **tables** in a  database. |
| **JOIN** | **Combines r**ows from two or more **tables** based  on a related **column** between them. |
| **DELETE** | **Removes** one or more **rows** from a **table.** |
| **CONVERT** | **Converts** a value from one **data type** to another. |
| **DATEDIFF** | **Calculates** the difference between two **dates** or  **times.** |
| **INSERT INTO VALUES** | **Adds** one or more **records** to a **table.** |
| **AND** | **Combines** multiple **conditions** in a **WHERE**  clause, ensuring that all conditions must be true. |
| **DECLARE** | **Defines** a **variable** in SQL. |
| **UPDATE SET** | **Modifies data** in a **table** by **updating existing**  **records.** |
| **SELECT DISTINCT** | **Retrieves unique values** for a specified **column**  or **columns.** |
| **FROM** | **Specifies** the source **table** or **tables** for a **query.** |
| **WHERE** | **Filters the result set based on a specified**  **condition.** |
| **INNER JOIN** | **Returns** only the **rows** that have **matching values** in **both tables.** |
| **ORDER BY** | **Sorts** the **result** set in **ascending** or **descending**  **order** based on one or more **columns.** |
| **AS** | **Renames** a **column** or **table** using an **alias.** |

* + 1. **All Queries used**

**Table 6: SQL Queries**

|  |  |  |
| --- | --- | --- |
| **MODEL** | **ACTIONS** | **SQL QUERIES** |
| **Customer** | Get all customer details | **SELECT \***  **FROM Customer;** |
| Get customer details by ID | **SELECT \***  **FROM Customer**  **WHERE Customer\_ID = @varID;** |
| Get phone numbers for a specific customer | **SELECT PhoneNumber FROM Phone**  **WHERE Customer\_ID = @varID;** |
| Create a new customer | **INSERT INTO Customer (Customer\_Name, Email) VALUES (@Customer\_Name, @Email);** |
| Update customer email | **UPDATE Customer**  **SET Email = @Email WHERE Customer\_ID = @varID;** |
| Delete a customer by ID | **DELETE FROM Customer**  **WHERE Customer\_ID = @varID;** |
| **Product** | Get all product details | **SELECT \***  **FROM Product;** |
| Get product details by ID | **SELECT \***  **FROM Product**  **WHERE Product\_ID = @varID;** |
| Get all products in a specific category | **SELECT Product\_Name FROM Product**  **WHERE Category = @varCategory;** |
| Create a new product | **INSERT INTO Product (Product\_Name, Category, BarCode, Price) VALUES (@Product\_Name, @Category, @BarCode, @Price);** |
| Update product price | **UPDATE Product**  **SET Price = @Price WHERE Product\_ID = @varID;** |
| Delete a product by ID | **DELETE FROM Product WHERE Product\_ID = @varID;** |
| Get products included in invoices | **SELECT DISTINCT Product\_Name**  **FROM Product JOIN Contain ON Product.Product\_ID = Contain.Product\_ID;** |
| **Customer\_Phone** | Get All  Customer\_Phone | **SELECT \***  **FROM**  **Customer\_Phone;** |
| Get Customer Phone by Customer ID | **SELECT \***  **FROM Customer\_Phone**  **WHERE Phone\_ID = @varPhoneID;** |
| Get All Phone Numbers by Customer ID | **SELECT Phone\_ID, PhoneNumber**  **FROM Customer\_Phone**  **WHERE Customer\_ID = @varCustomerID;** |
| Insert a New Phone Number | **INSERT INTO Customer\_Phone (Customer\_ID, PhoneNumber)**  **VALUES (@CustomerID, @PhoneNumber);** |
| Update a Phone Number | **UPDATE Customer\_Phone**  **SET PhoneNumber = @PhoneNumber**  **WHERE Phone\_ID = @varPhoneID;** |
| Delete a Phone Number by Phone ID |  |
| **Payment** | ​​ Get All Payments | **SELECT \***  **FROM Payment;** |
| Get Payment by Payment\_ID | **SELECT \***  **FROM Payment**  **WHERE Payment\_ID = @PaymentID;** |
| Create Payment | **INSERT INTO Payment (Amount, Invoice\_ID, Payment\_Method)**  **VALUES (@Amount, @InvoiceID, @PaymentMethod);** |
| Update Payment | **UPDATE Payment**  **SET Amount = @Amount,**  **Invoice\_ID = @InvoiceID,**  **Payment\_Method = @PaymentMethod**  **WHERE Payment\_ID = @PaymentID;** |
| Delete Payment | **DELETE FROM Payment**  **WHERE**  **Payment\_ID=@Payment;** |
| **Price\_List** | Get All Price\_List | **SELECT \***  **FROM Price\_List;** |
| Get Price\_List by ProductID | **SELECT \* FROM Price\_List**  **WHERE Product\_ID = @ProductID;** |
| Get All Prices by Date | **SELECT \* FROM Price\_List**  **WHERE Date\_Effective = @DateEffective;** |
| Create Price | **INSERT INTO Price\_List (Product\_ID, Price, Date\_Effective)**  **VALUES (@ProductID, @Price, @DateEffective);** |
| Update Price | **UPDATE Price\_List**  **SET Price = @Price, Date\_Effective = @DateEffective**  **WHERE Product\_ID = @ProductID;** |
| Delete Price | **DELETE FROM Price\_List**  **WHERE Product\_ID = @ProductID;** |
| **Invoice** | Get All Invoices | **SELECT \* FROM Invoice;** |
| Get Invoice by ID | **SELECT \* FROM Invoice**  **WHERE Invoice\_ID = @InvoiceID;** |
| Get All Invoices by Customer ID | **SELECT \* FROM Invoice**  **WHERE Customer\_ID = @CustomerID;** |
| Create Invoice | **INSERT INTO Invoice (Customer\_ID, Date, Total\_Amount, Payment\_ID)**  **VALUES (@CustomerID, @Date, @TotalAmount, @PaymentID);** |
| Update Invoice | **UPDATE Invoice**  **SET Customer\_ID = @CustomerID, Date = @Date, Total\_Amount = @TotalAmount, Payment\_ID = @PaymentID**  **WHERE Invoice\_ID = @InvoiceID;** |
| Delete Invoice | **DELETE FROM Invoice**  **WHERE Invoice\_ID = @InvoiceID;** |
| **Contain** | Get All Contain | **SELECT \***  **FROM Contain;** |
| Get Contain by Product ID and Invoice ID | **SELECT \***  **FROM Contain**  **WHERE Product\_ID = @ProductID AND Invoice\_ID = @InvoiceID;** |
| Get All Contain Records by Invoice ID | **SELECT \***  **FROM Contain**  **WHERE Invoice\_ID = @InvoiceID;** |
| Create Contain | **INSERT INTO**  **Contain (Product\_ID, Invoice\_ID)**  **VALUES (@ProductID, @InvoiceID);** |
| Delete Contain | **DELETE FROM Contain**  **WHERE Product\_ID = @ProductID AND Invoice\_ID = @InvoiceID;** |

# **Chapter 5: APPLICATION PROJECT STRUCTURE**

## **PROJECT STRUCTUR**

From the aforementioned detailed visualization of our application, the project structure is designed as bellow:

**A screenshot of a computer program

Description automatically generated**

**Figure 3 Project’s server structure**

All of the **backend** files are stored in **Sources Root java** and consist of the following

parts:

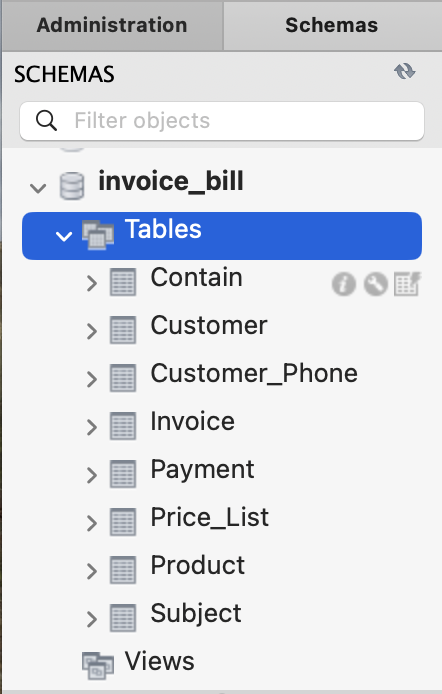
- File **pdmProject.java**: this class contains the main function, thus the program is set up and run from here.

- File **SQLConnection.java**: this class establishes the connection between the

developer’s local SQL Server Database and the code Java.

- Folder **engine**: contains all of the necessary entities managed in the website: Customer,

Invoice, Price, Price\_List, Product and Product\_List.

****

**Figure 4: SQL structure**

- **MySQL:** Serves as the relational database management system (RDBMS),

responsible for storing and managing structured data used by the application; it is

crucial for building scalable web applications**.**

## **SQL CONNECTION**

In order to connect MySQL database and Java, the JDBC Driver jar is installed and

added to the dependency of Maven. In our code, we specify the login name ‘sa’, password, and the url which consists of the developer’s Server name of MySQL and30 the database name: invoice\_bill. Then, via the static method connectDb(), the database is connected to Java.

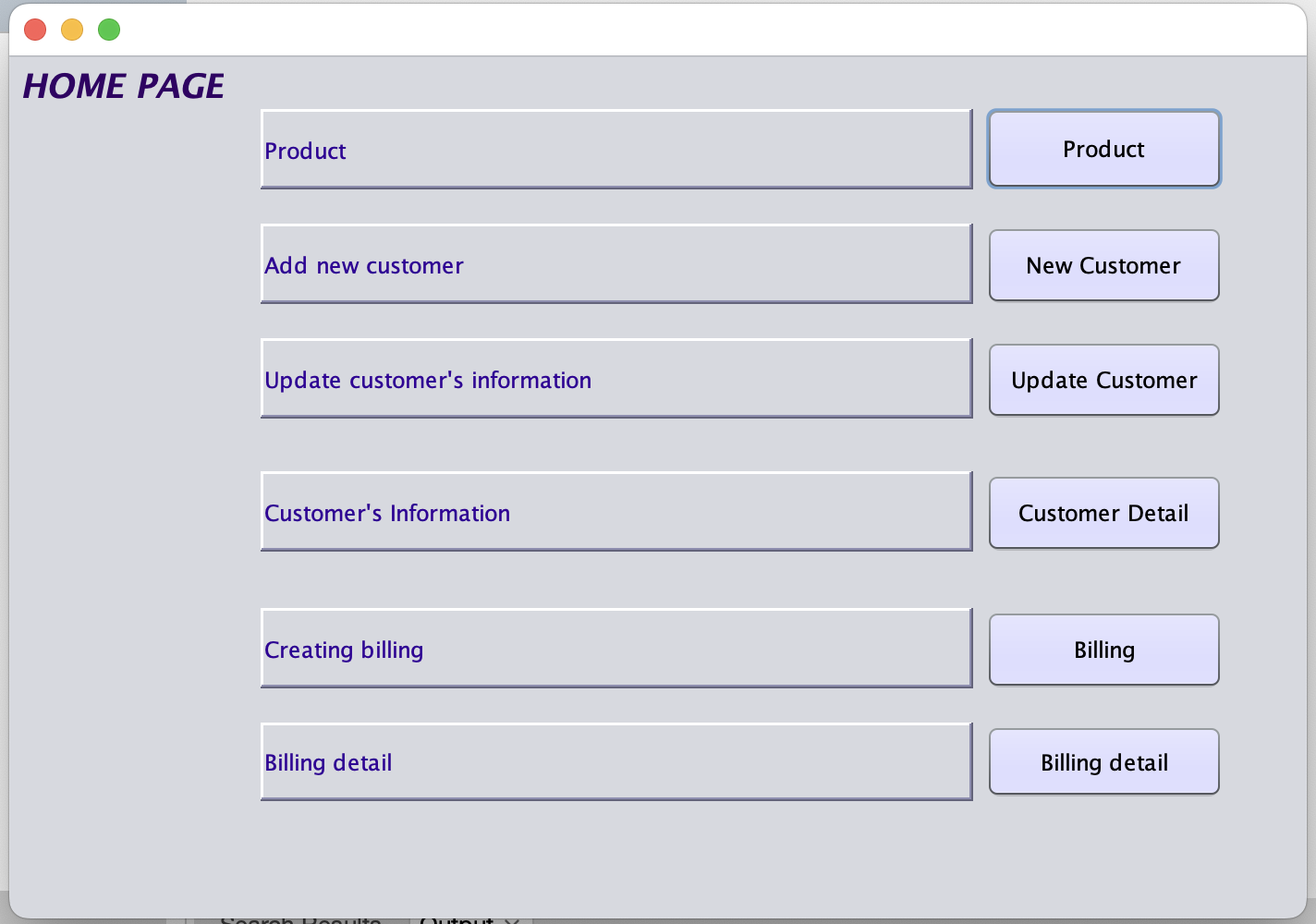
A screenshot of a computer

Description automatically generated

**Figure 5 SQL Connection**

## **PROJECT DEMO**

**Home page**

****

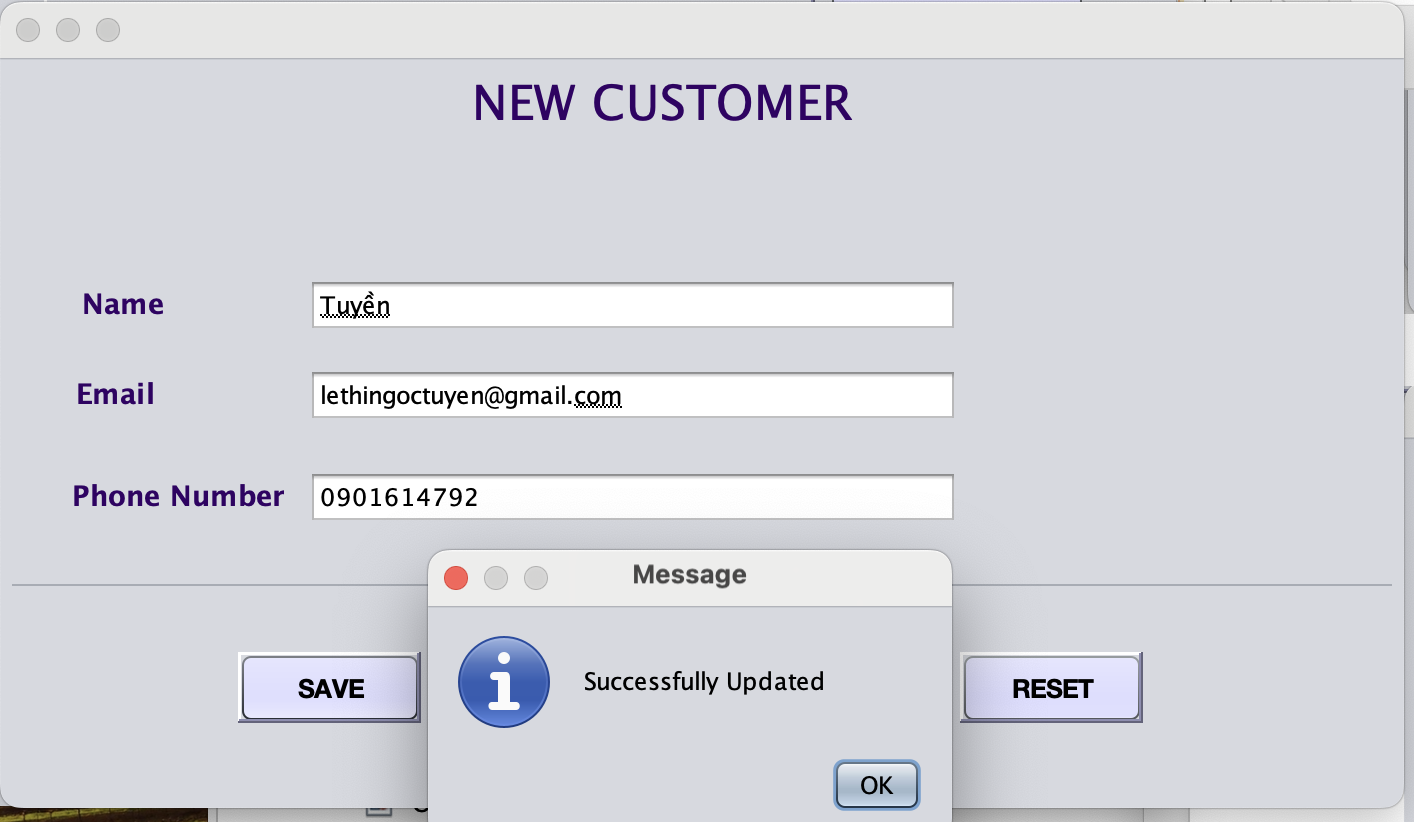
**Figure 6: Home Page**

**Add new customer**

**A screenshot of a computer

Description automatically generated**

**Figure 7: Add New Customer**

****

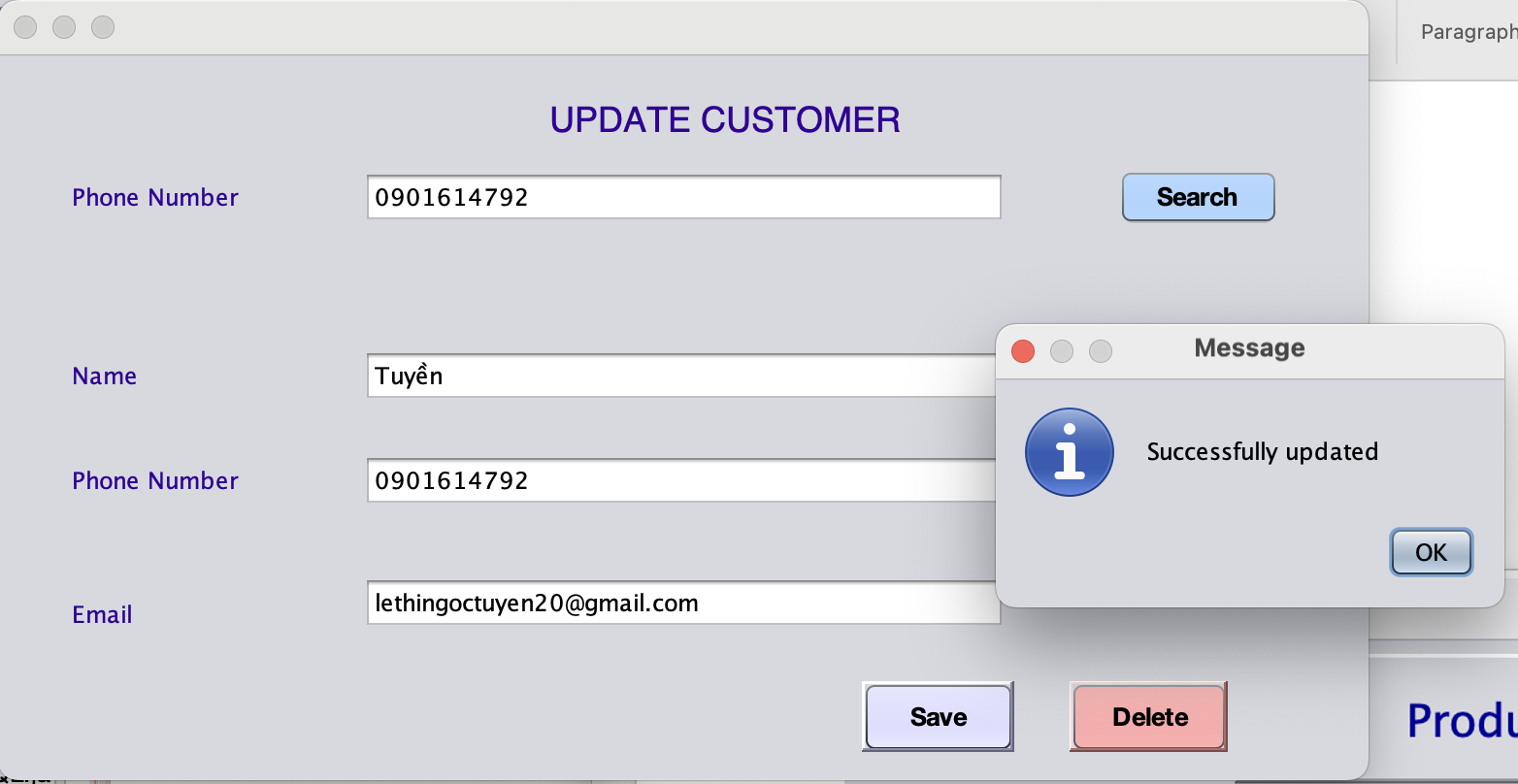
**Figure 8: Successfully add new customer**

**Update customer**

**A screenshot of a computer

Description automatically generated**

**Figure 9: Update customer**



**Figure 10: Update customer successfully**

A screenshot of a computer

Description automatically generated

**Figure 11: Delete customer**

**A screenshot of a computer

Description automatically generated**

**Figure 12: Print out the detail**

**Add product to the bill**

**A screenshot of a computer

Description automatically generated**

**Figure 13: Add product to the bill**

A screenshot of a computer

Description automatically generated

**Figure 14: The bill format**

A screenshot of a computer

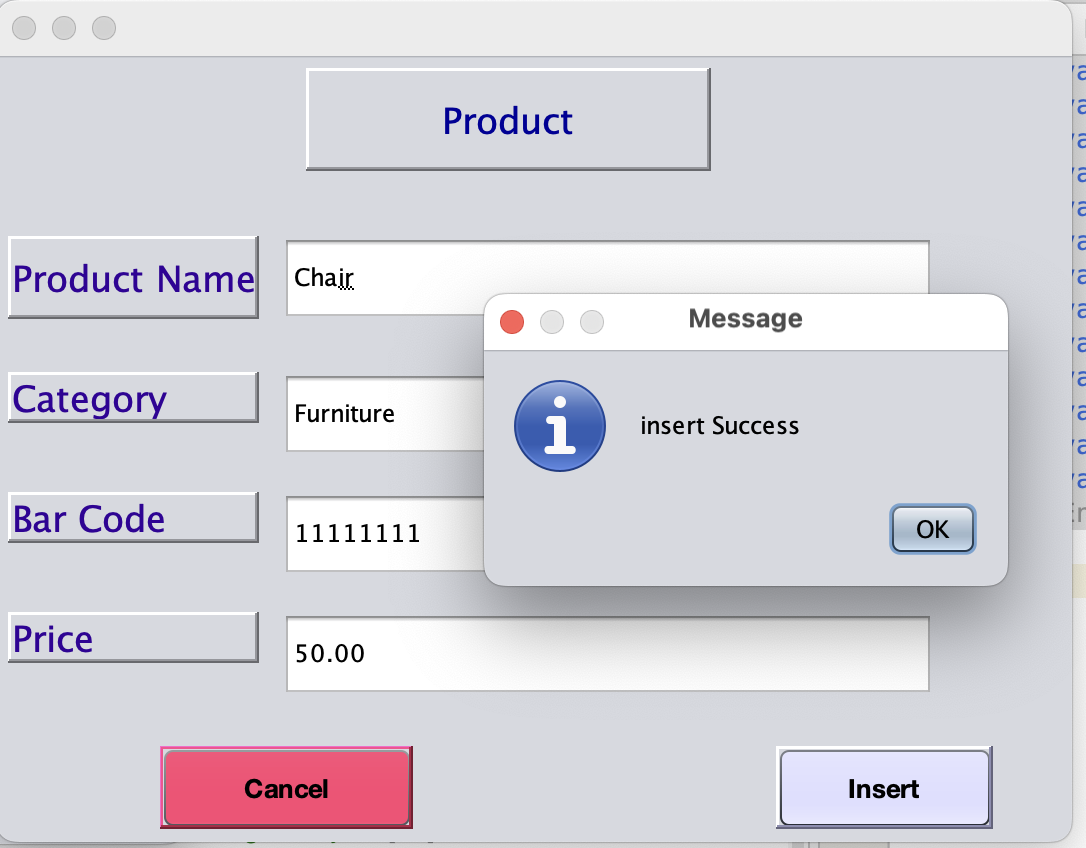
Description automatically generated

**Figure 15: Search product’s information**

A screenshot of a computer

Description automatically generated

**Figure 16: Show detail product**

****

**Figure 17: Add product**

# **Chapter 6: CONCLUSION**

## **ACHIEVE GOALS**

The Invoice Billing Database project successfully achieves its primary goal of automating and streamlining the billing process for shopping complexes and retail environments. By incorporating features such as detailed product information, comprehensive customer data management, invoice record tracking, and keyword recognition, the system enhances operational efficiency and reduces manual errors. Additionally, the inclusion of a well-organized price list and payment information ensures seamless customer transactions and improved satisfaction. Overall, the system establishes a reliable and user-friendly shopping platform that benefits both customers and staff.

## **FUTURE WORK**

* + Integration with Mobile Applications  
    Expanding the system to integrate with mobile platforms for customer convenience, allowing users to view invoices, make payments, and track orders in real-time.
  + Advanced Analytics and Reporting  
    Enhancing the database with advanced analytics to generate insightful reports on sales trends, customer behavior, and product performance.
* Multi-Language Support  
  Adding multi-language functionality to cater to diverse customer bases in retail environments worldwide.
* Cloud Storage and Scalability  
  Migrating the database to the cloud to improve accessibility, scalability, and disaster recovery capabilities.

## **SYSTEM ANALYSIS**

The Invoice Billing Database system addresses the critical needs of retail environments by focusing on efficiency, accuracy, and usability.

1. Functional Requirements
   * Product Information: Stores detailed data on product names, categories, prices, and availability.
   * Customer Data: Maintains accurate customer contact information for personalized service.
   * Invoice Records: Tracks all invoices with details of items purchased, dates, and payment methods.
   * Price List: Centralized management of pricing ensures consistency and transparency.
2. Non-Functional Requirements
   * Performance: Ensures fast response times for invoice generation and keyword recognition.
   * Scalability: Designed to handle increasing amounts of data as business operations grow.
   * Security: Implements access controls and encryption to protect sensitive customer and payment data.
3. System Challenges
   * Keyword Recognition: Ensuring high accuracy in recognizing keywords across diverse products.
   * Real-Time Updates: Maintaining synchronized data during high-volume transactions.
   * User Training: Simplifying the interface to reduce the learning curve for staff.

## **REFERENCES**

* SQL Server with NODE.JS: (851) NodeJS - SQL Server bài 1 - Cài đặt nodejs giới thiệu về npm tạo ứng dụng node đầu tiên - YouTube
* Geeksforgeeks.org : [**https://www.geeksforgeeks.org/**](https://www.geeksforgeeks.org/)
* Javatpoint:[**https://www.javatpoint.com/**](https://www.javatpoint.com/)