WEDDING CARD SALES SYSTEM

23IT454 - MINI PROJECT

A REPORT

Submitted by

R.LAKSHANA(9517202306050)

K.B.SHINMAYA (9517202306091)

M.VIKASHINI DHAKSHA(9517202306118)

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(AUTONOMOUS)

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this project report titled Wedding Card Sales System the bonafide work of **R.LAKSHANA(9517202306091),K.B.SHINMAYA (9517202306091), M.VIKASHINI DHAKSHA(9517202306118)** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

DR.N.MALATHY,B.E, M.E, PhD
Internal guide
Associate Professor,
Department of Information Technology,

Mepco Schlenk Engineering College,
Sivakasi.

Dr.T.REVATHI, M.E., Ph.D.,
Head of the Department
Senior Professor,
Department of Information Technology,
Mepco Schlenk Engineering College,
Sivakasi.

Submitted for Viva-Voce Examination held at **MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI (AUTONOMOUS)** on **……**

Internal Examiner I **Internal Examiner II**

ABSTRACT

ABSTRACT

"**Geetha Wedding Cards**" is a user-friendly desktop application designed to modernize the process of selecting and purchasing wedding and invitation cards. Developed using Java Swing for the frontend and MySQL as the backend database, this application aims to provide a smooth and digital experience for both customers and administrators.The main objective of this project is to simplify the traditional wedding card selection process, which usually involves time-consuming visits to shops and limited choices. "Geetha Wedding Cards" brings this experience into a digital platform, making it easier for users to explore a wide variety of cards and for shopkeepers to manage their business efficiently. The application also applies core concepts of object-oriented programming and database management, ensuring both functionality and data security.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES:

CHAPTER 1

INTRODUCTION

1.1 AIM

To develop a user-friendly Java Swing application named **Geetha Wedding Cards** that allows users to view, customize, and purchase wedding invitation cards, and enables administrators to manage card data and monitor sales.

1.2 OBJECTIVE

i. To provide an interactive platform for users to register, login, and explore various wedding card designs.

ii. To categorize cards based on religion and style for easier browsing.

iii. To implement secure purchase functionality with address and payment details.

iv. To allow administrators to add, update, delete cards, and view monthly sales.

v. To store and retrieve all data using a database.

1.3 PROBLEM DEFINITION

The conventional method of buying wedding cards involves visiting stores, browsing limited physical samples, and manually placing orders. This process is time-consuming and lacks personalization. Additionally, managing card inventory and tracking purchases is tiresome for shop owners. This project gives a better solution by using a computer application to make everything easier, faster, and better organized.

1.4 OBJECT ORIENTED PROGRAMMING CONCEPT

This project follows the core principles of **Object-Oriented Programming (OOP)** using **Java** in the **NetBeans IDE**, which provides a modular and reusable structure for building the application. The application is developed using **Java Swing** for GUI and applies OOP concepts as follows:

- **Encapsulation**

Each screen (like LoginFrame, PurchasePage, AdminFrame) is implemented as a separate class that encapsulates its own components, event listeners, and logic.

The database utility class conn encapsulates connection setup and access methods, keeping the rest of the code clean.

- **Inheritance**

All GUI classes such as AdminFrame, LoginFrame, etc., extend JFrame, which provides window behavior and GUI capabilities from the Swing library.

This allows code reuse and ensures a consistent structure across different screens.

- **Polymorphism**

Button actions and UI interactions use polymorphism through overridden actionPerformed() methods. Each button can have its own behavior despite using the same method name.

Listeners adapt dynamically depending on the button or event source clicked.

- **Abstraction**

The application separates GUI design from core logic. For example, methods like loadSalesByMonthYear() and saveOrderToDatabase() hide the database complexity from the user interface.

The conn class abstracts the internal details of JDBC setup, so other classes just call conn.getConnection() to work with the database.

1.5 DATABASE MANAGEMENT CONCEPT

The backend of the application uses MySQL. That includes:

i. Use of primary and foreign keys to define relationships.

Primary Keys uniquely identify each record in a table.

Foreign Keys establish relationships between tables, ensuring data consistency.

These relationships enforce referential integrity and reduce data redundancy.

ii. Stored procedures for fetching cards and cart data efficiently.

The system uses stored procedures in MySQL for optimized and reusable data access.

iii. Structured queries to perform create,read,update,delete operations on tables like users, cards, cart, and purchases.

iv. **JDBC integration to connect Java code with the MySQL database**.

JDBC handles:

- Connection management: Opening and closing connections using DriverManager.
- Query execution: Using PreparedStatement or CallableStatement for parameterized queries.
- Result processing: Fetching data using ResultSet.

JAR FILES NEEDED IN THIS PROJECT:

- mysql-connector-java.jar - JDBC driver to connect Java with MySQL database
- itextpdf-5.x.x.jar - Used to generate PDF bills/invoices from Java

CHAPTER 2

2.1 ER DIAGRAM

Entity:

An entity is any object or concept that can be stored in a database and has a distinct existence.

Attribute:

An attribute is a property or characteristic of an entity. These become the columns in a table.

ER Diagram (Entity-Relationship Diagram)

An ER Diagram is a visual representation of the data model. It shows:

Entities (usually in rectangles)

Attributes (ovals connected to entities)

Relationships (diamonds connecting entities)

ATTRIBUTES AND ENTITIES:

cards: card_id, name, type, price, image_path, descriptions, added_on
cart: cart_id, userid, card_id, name, price, quantity, total
orders: order_id, user_id, card_name, quantity, amount, country, state, address, payment_method, order_date, total_price

TABLE:

User,Cart	Put	User id
User,order	Buys	User id
Cart,Card	Adds	Card id

ER DIAGRAM:

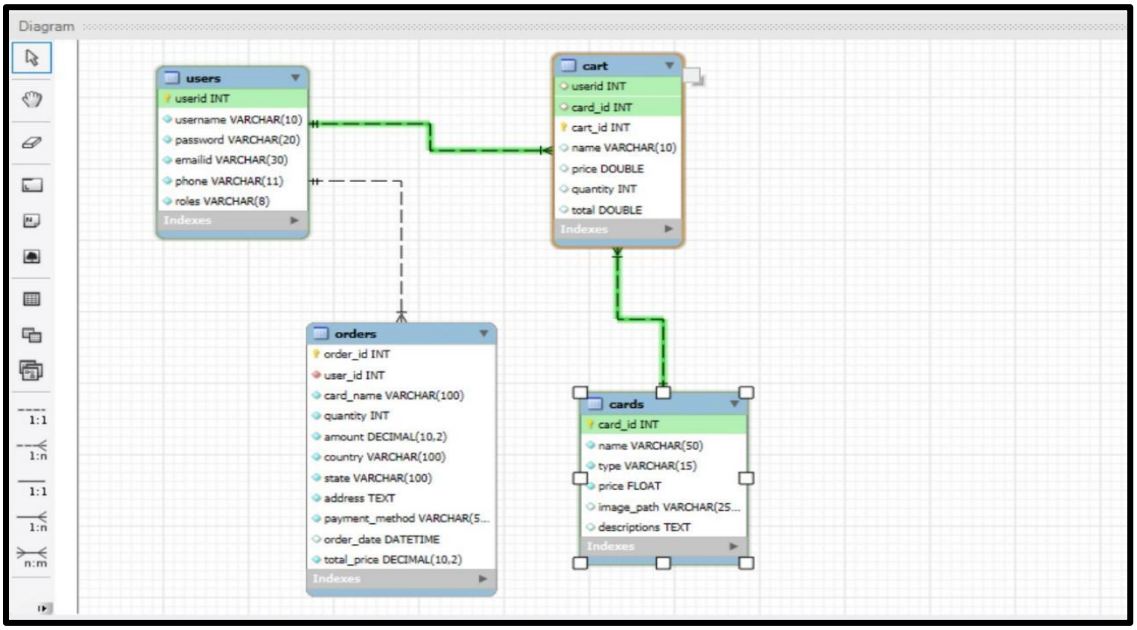


Fig 2.1.1

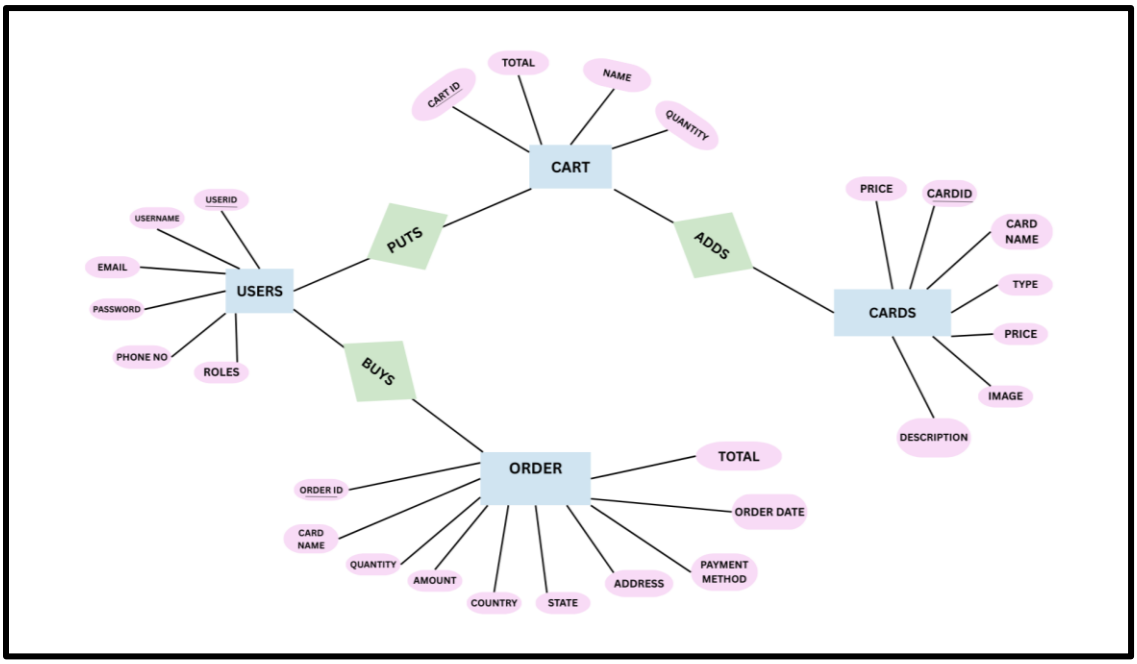
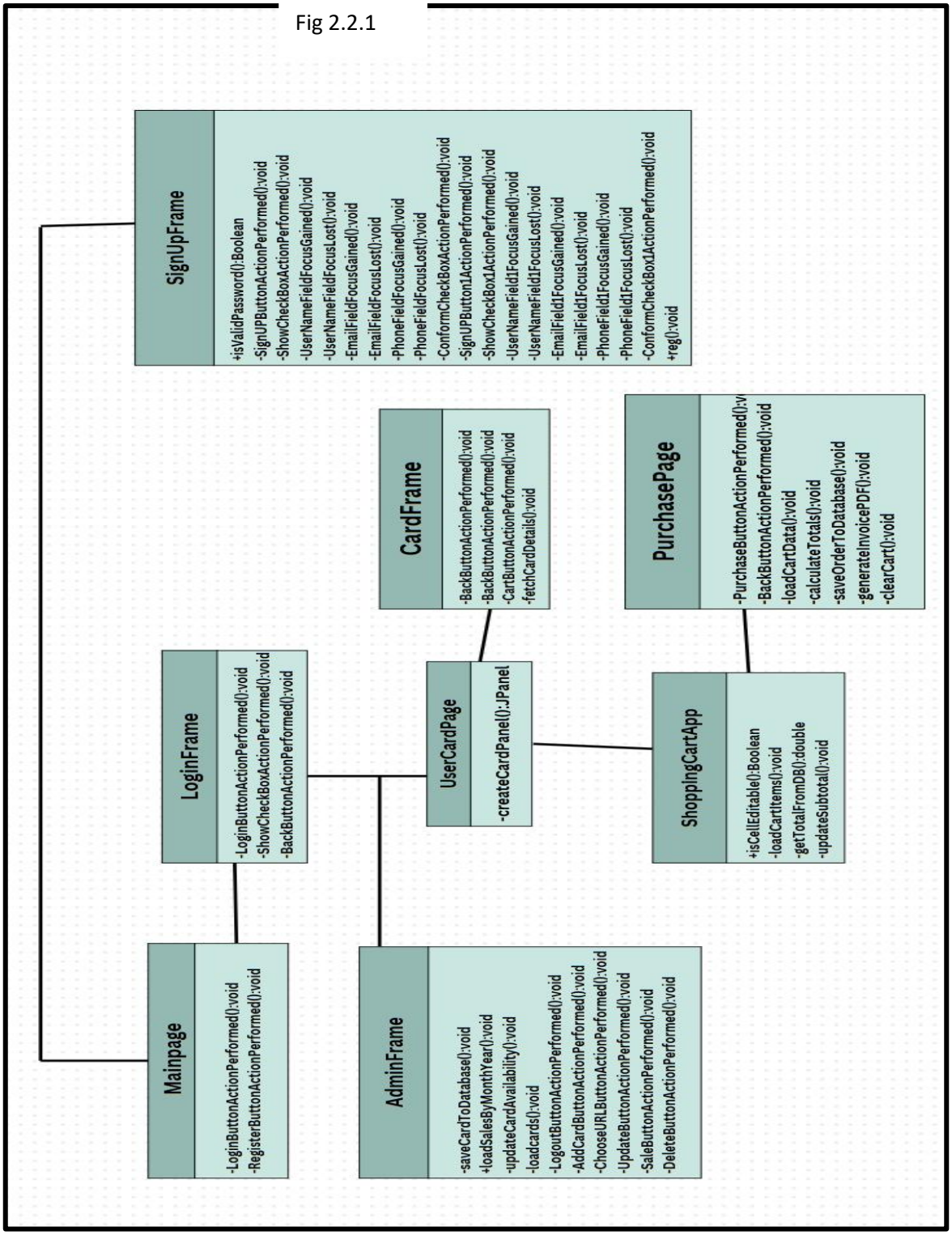


Fig 2.1.2

2.2 CLASS DIAGRAM

Fig 2.2.1



CHAPTER 3

IMPLEMENTATION METHODOLOGY

3.1 MODULES IDENTIFIED

- **Database Connection Module-**Connects Java application to MySQL database using JDBC.
- **Main Frame Module -** Entry point showing options for Login and Sign Up.
- **Login Module -** Allows existing users to log in securely.
- **SignUp Module -**New users can register by providing email, username, phone number, and password.
- **User Module** - Displays available wedding cards categorized by religion (All, Hindu, Muslim, Christian) Allows adding cards to the shopping cart.
- **Shopping Cart Module -** Shows selected cards with options to update quantity or proceed to purchase.
- **Purchase Module -** Collects delivery address, payment method, and completes the order.
- **Admin Module -**Admin can Add, Update, Delete cards.Admin can view monthly sales reports.
- **Image module**
- **Try catch exception**

3.2 MODULES DESCRIPTION

1 .GUI Components:

Component	Purpose in our App
JFrame	Main window container for each screen (e.g., Login, Cart, Admin, PurchasePage).
JPanel	Sub-panel inside JFrame to group elements and manage layouts.

Component	Purpose in our App
JLabel	Displays text like labels, headings, or instructions. Also used to display images.
JTextField	Input field for users to type text (e.g., address, card name, price).
JPasswordField	Special version of JTextField used for entering passwords securely.
JButton	Interactive button for actions like “Login”, “Add to Cart”, “Purchase”.
JComboBox	Drop-down list to select options (e.g., state, country, payment method, card type).
JTable	Used in Shopping Cart and Admin panel to show card list, orders, and editable quantities.
DefaultTableModel	Table model that allows dynamic updates to the JTable’s data.
JScrollPane	Enables scrolling when content (e.g. tables, text) exceeds visible area.
ImageIcon	Displays card images (preview or thumbnail).
JFileChooser	Lets admin select an image file to upload for a card.
JOptionPane	Shows pop-up dialog boxes for messages, confirmations, or errors.
FileNameExtensionFilter	Restricts file selection in JFileChooser to specific formats like .jpg, .png.
BorderLayout / GridLayout	Used to arrange GUI components in structured layouts (top, bottom, grid, etc).
JTabbedPane	Used in UserCardPage to switch between categories (e.g., Hindu, Muslim, Christian).
JTextArea	Multi-line text box, used to show invoice in showBillOnScreen() before PDF download.

2. **Database Connection Module-**Connects Java application to MySQL database using JDBC.

Connection	To create and manage database connections
DriverManager	To get a Connection to database
PreparedStatement	To execute SQL queries safely
ResultSet	To store and read query results
SQLException	To handle database errors

Fig 3.2.1

SYNTAX:

```
public void connectToDatabase() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
            Connection  conn = DriverManager.getConnection(url, user, password);
             JOptionPane.showMessageDialog(null, "Database connected successfully!");
    }
```

3. **Main Frame (Homepage)**

Description:

The first page the user sees when the app starts. It offers **Login** and **Sign Up** options.

Explanation:

Clicking Login → moves to Login Frame.

Clicking Sign Up → moves to SignUp Frame.

4. Login Frame

Description:

Existing users can log into the application using their username and password.

Explanation:

If credentials are correct → moves to UserCardPage.

If wrong → shows error message. Back button available to return to Main Frame.

5. SignUp Frame

Description:

New users can create an account by entering their email, username, phone number, and password.

Explanation:

Password must have at least 3 digits, 1 uppercase, and 1 lowercase letter.

After successful registration → user is redirected to Login Frame.

Validation :

The password should be atleast 6 characters.

The email field gets accepted only if it conatins a uppercase , lowercase ,a digits and a special character.

All the fields are required.

6.UserCardPage (User Home)Description:

Users can browse wedding cards .

Explanation:

Clicking a card → opens that card's details in CardFrame.

CART button → view shopping cart (ShoppingCartApp).

LOGOUT button → logout to Main Frame.

Each card = 1 button.

Buttons show on the page . Each button can be clicked to see card details.

Why we used HTML?

JButton cardButton = new JButton("<html> " + cardName + "
Price: " + cardPrice + " </html>");

In Java Swing , normal buttons (JButton) can only show simple, plain text.

For Multiline text and Formatted text (like bold, center, color)

<html> ... </html>	Tells Swing to use HTML formatting inside the button.
** ... **	Centers the text horizontally inside the button.
** **	Inserts a line break (moves to next line).

How Buttons are alligned automatically?

cardPanel.add(cardButton);

JPanel cardPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 20));

FlowLayout.LEFT → aligns all buttons **starting from the left**.

20, 20 → gap of 20 pixels horizontally and vertically between buttons.

If no space,buttons start new row.

Procedure Used:

CallableStatement cs = conn.prepareCall("{CALL GetCardsByType(?)}");

cs.setString(1, typeFilter);

GetCardsByType(type) – Fetches cards based on type (All, Hindu, Muslim, Christian).

FLOWCHART:

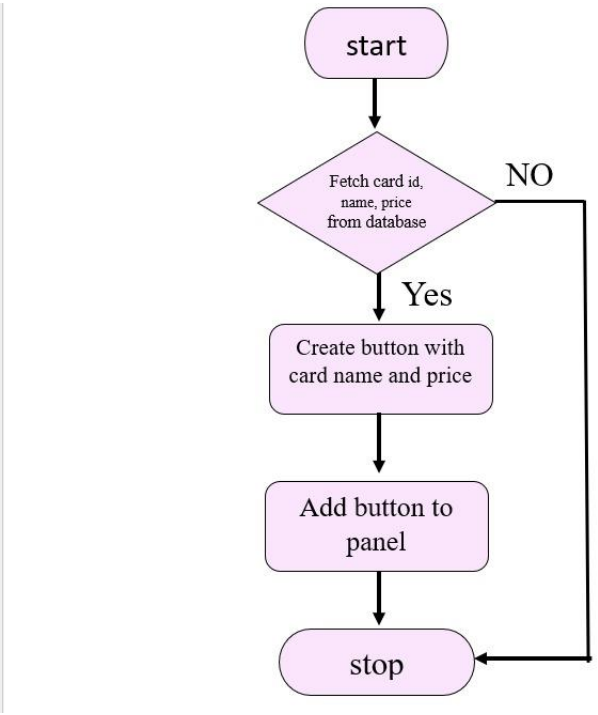


Fig 3.2.2

7. CardFrame (Card Details Page)

Description:
Displays full details of a selected card (name, price, description, image).

Explanation:

CART button → adds card to Shopping Cart.

BACK button → goes back to UserCardPage.

8.ShoppingCartApp (Cart Page)

Description:

Shows the list of cards added by the user with their quantities and total prices. Users can edit quantity directly in the cart.

PURCHASE button → proceeds to PurchasePage.

BACK button → goes back to UserCardPage.

ALGORITHM :

Load user's cart items from database.

Show items in table.

Allow quantity changes.

Show subtotal.

CODE EXPLANATION:

- This listens for any change made to the cart table (JTable).
- Especially if user edits a cell ,like changing the quantity.
  - e.getType() == UPDATE: -Checks only if something was edited.
  - e.getColumn() == 3: -Checks only if Quantity column was changed (Column 3).
  - Finds which row the user edited
  - Reads the new quantity entered by the user from the table. Converts it from String → Integer.
  - Reads the price of the card from the same row, Column 2 (Price column).Converts it from String → Double.
  - Sets the new total into Column 4 (Total column) for that row.
  - Recalculates the overall subtotal (sum of all rows' totals).
  - Updates the Subtotal label at the bottom.

Procedure Used Here:

CallableStatement cs = conn.prepareCall("{call **GetCartItemsByUserId**(?)}");

cs.setInt(1, userId);

The procedure we used here to get the cart items by the given user id.

FLOWCHART:

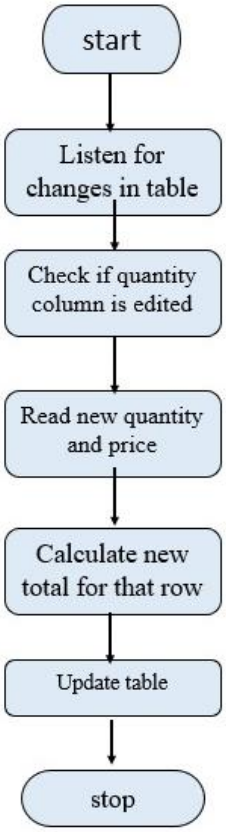


Fig 3.2.3

removeSelectedRow()

Purpose: Removes the selected item from the cart (both from the table and the database).

How it works:

- **Check Selection**

Gets the selected row index (table.getSelectedRow()).

If no row is selected, does nothing.

- **Database Deletion**:

Extracts the card name from the selected row.

Executes a DELETE **query** (DELETE FROM cart WHERE userId = ? AND name = ?).

- **Update UI**:

Removes the row from the table (model.removeRow(selected)).

Calls calculateTotals() to refresh the subtotal

Functions used here:

Calls calculateTotals() - calcaltes the total and returns it.

9. PurchasePage

Description: After the user is done adding cards into the cart,
they go to the Purchase Page.

The PurchasePage is the final step of the card ordering process where the user:Reviews cart item,Enters delivery address and payment method,Places the order,Views the bill,and sends it in gmail.

loadCartData()

Calls stored procedure **GetCartItemsByUserId(?) t**o load cart items for the current user

Here we used procedure to get the cart items by user

Reads: name of the card , price and quantity**,** image_path to show icons

Calculates total for each item (price × quantity)

Adds all rows to the JTable

calculateTotals()

Sums up all item totals to get **subtotal**

Adds **10% tax** and **₹50 shipping**

Displays values in subtotal, tax, and total labels

Invoice Display and Email

- Builds a receipt using data from the CardTable.
- Shows it in a new JFrame with a JTextArea.
- Prompts the user: "Send invoice to email?"
- If yes, gets email from DB and calls sendInvoiceEmail().

sendInvoiceEmail( )

- Sends the invoice using JavaMail API.
- Authenticates using Gmail with an App Password.
- Sends a plain text email with the invoice.
- Session.getInstance(...): Creates a mail session using properties (props) and an authenticator.
- props contains SMTP settings (host, port, TLS, etc.).
- Authenticator: Provides Gmail email and app password (fromEmail, password) when needed.
- This ensures the SMTP server (Gmail) knows it's a legitimate sender.

clearCart()

- Deletes all entries in the cart for that user after successful purchase
- Refreshes the cart table and total labels

10.Admin Page It has different tabs for the Admin to:

- Add new cards.
- Update existing cards.
- Delete cards.
- View monthly sales

1. Add new cards:

The admin can upload new cards in the add panel

2. Update New Cards:

Why we used switch statement for update?

In your GUI, the **ComboBox shows user-friendly names** like:

"Card Name" "Card Price", "Card Type", e;tc.

But in the database, the column names arename, price, type, etc.

The switch statement is used to map from UI label → database column.

3. **Delete Card:**

The values are viewed in a table.We can delete the cards using card id.

4.View sales

The sales can be viewed in a JTable.

it helps the admin check: How many cards were sold,The quantity,Total revenue,And the date of each order.

11.IMAGE MODULE:

Gets the image file path from the database result set

ImageIcon– object from the image file at the specified

Image -object from the icon

Scales it to 180×120 pixels using smooth scaling algorithm (Image.SCALE_SMOOTH)

This ensures consistent display size regardless of original image dimensions

The ImageIcon constructor can accept:

File paths (as in this case)

URLs

Byte arrays

Image objects

Scaling options (Image.SCALE_* constants)

- SCALE_SMOOTH: Highest quality (slower)
- SCALE_FAST: Lower quality (faster)
- SCALE_DEFAULT: Platform-dependent default

12.try-catch block

Why we need try-catch?

· Use try-catch to **protect** your app from crashes.

· Place risky code (like DB access or file writing) inside try.

· Use catch to show error messages or log details.

Two types of exception:

1. Checked Exceptions - Handled during compile time. Java forces you to catch or declare them
2. Unchecked Exceptions - Occur during runtime. Not checked by compiler, but can crash the program.

Code Area	**Likely Exception Type**	**Type**
DriverManager.getConnection(...)	SQLException	Checked
Class.forName(...)	ClassNotFoundException	Checked
Integer.parseInt(...)	NumberFormatException	Unchecked
image.getScaledInstance(...)	NullPointerException (if image is null)	Unchecked

13 . IMPORT STATEMENTS

Import Statement	Purpose / Description
import java.awt.Image;	Handles image processing and scaling in the GUI.
import java.io.File;	Used to manage file paths and file selection.
import java.io.FileOutputStream;	Used to write data (PDF) into files.
import java.sql.Connection;	Establishes a connection with a SQL database.
import java.sql.DriverManager;	Loads JDBC driver and creates database connections.
import java.sql.PreparedStatement;	Executes parameterized SQL queries (safe from injection).
import java.sql.ResultSet;	Retrieves results from SQL SELECT queries.
import java.sql.SQLException;	Handles SQL-related errors and exceptions.
import java.sql.CallableStatement;	Calls stored procedures from Java.
import java.util.Date;	Represents and manages date/time values.
import java.util.logging.Level;	Defines logging levels (e.g. INFO, ERROR).
import java.util.logging.Logger;	Used for logging error and debug messages.
import java.util.*;	Imports all classes from java.util (e.g., ArrayList, Date).
import javax.swing.*;	Imports all Swing components (JFrame, JButton, etc.).

13 . IMPORT STATEMENTS

Import Statement	Purpose / Description
import javax.swing.ImageIcon;	Loads and handles image icons in the GUI.
import javax.swing.JLabel;	Displays static text or images in the GUI.
import javax.swing.JTable;	Displays tables in the GUI.
import javax.swing.filechooser.FileNameExtensionFilter;	Filters file types in JFileChooser (e.g. .jpg, .png).
import javax.swing.table.DefaultTableModel;	Used to manipulate data in JTable easily.
import javax.swing.event.TableModelEvent;	Detects changes (e.g. quantity update) in table cells.
import java.awt.BorderLayout;	Lays out components in top, bottom, center, etc.
import java.awt.Color;	Sets or changes color in GUI components.
import javax.swing.SwingConstants;	Aligns text or components (e.g. center).
import com.itextpdf.text.Chunk;	Adds blank lines or spacers in PDF layout.
import com.itextpdf.text.Document;	Creates and represents a PDF document.
import com.itextpdf.text.Font;	Sets font styles for PDF text.
import com.itextpdf.text.Paragraph;	Adds paragraph blocks in PDF.
import com.itextpdf.text.pdf.PdfPTable;	Creates tables inside PDF documents.
import com.itextpdf.text.pdf.PdfWriter;	Writes and saves the final PDF to a file.

CHAPTER 4

PROGRAM

4.1 DBMS:

CREATING A DATABASE :

```
create database project;

use project;
```

CREATING TABLE:

USERS:

```
create table users (
    userid int primary key auto_increment,
    username varchar(10) not null,
    password varchar(20) not null,
    emailid varchar(30) not null,
    phone varchar(11) not null ,
    roles varchar(8) not null );
```

INSERTING ADMIN DEATAILS INTO TABLE

```
insert into users (username,password,emailid,phone,roles) values('admin','admin','admin@gmail','6379520078','admin');
```

CREATING TABLE:

CARDS:

```
CREATE TABLE cards (
    card_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    type VARCHAR(15) NOT NULL,
    price FLOAT NOT NULL,
    image_path VARCHAR(255),
    descriptions VARCHAR(255)
);
ALTER TABLE cards MODIFY COLUMN name VARCHAR(100);
ALTER TABLE cards MODIFY COLUMN descriptions TEXT;
ALTER TABLE cards ADD COLUMN added_on TIMESTAMP DEFAULT CURRENT_TIMESTAMP;
```

CREATING TABLE CART:

```
CREATE TABLE cart (
    userid int ,
    card_id int,
    cart_id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    name VARCHAR(10),
    price DOUBLE,
    quantity INT,
    total double,
    foreign key (userid) references users(userid),
    foreign key (card_id) references cards(card_id)
```

CREATE TABLE ORDERS:

```
CREATE TABLE orders (
    order_id      INT PRIMARY KEY AUTO_INCREMENT,
    user_id       INT           NOT NULL,
    card_name     VARCHAR(100)   NOT NULL,
    quantity      INT           NOT NULL,
    amount   DECIMAL(10,2)  NOT NULL,
    country       VARCHAR(100)   NOT NULL,
    state         VARCHAR(100)   NOT NULL,
    address       TEXT          NOT NULL,
    payment_method VARCHAR(50)   NOT NULL,
    order_date    DATETIME       DEFAULT CURRENT_TIMESTAMP,
```

```
    total_price   DECIMAL(10,2)   NOT NULL,

        FOREIGN KEY (user_id) REFERENCES users(userid)

);
```

PROCEDURES:

1)TO GET THE USERID FOR CART ITEMS:

```
CREATE PROCEDURE GetCartItemsByUserId(IN uid INT)

BEGIN

   SELECT * FROM cart WHERE userid = uid;

END //

DELIMITER ;
```

2.TO GET CARD TYPES:

```
CREATE PROCEDURE GetCardsByType(IN typeFilter VARCHAR(50))

BEGIN

   IF typeFilter = 'All' THEN

      SELECT * FROM cards ORDER BY added_on DESC;

   ELSE

      SELECT * FROM cards WHERE type = typeFilter ORDER BY added_on DESC;

   END IF;

END $$
```

FUNCTIONS:

TO CALCULATE THE TOTAL

```
CREATE FUNCTION calculate_total(price DOUBLE, quantity INT)

RETURNS DOUBLE

BEGIN

   RETURN price * quantity;

END //
```

4.2 JAVA:

1)MAINFRAME:

```
public class MainFrame extends javax.swing.JFrame {
   public MainFrame() {
      initComponents();  }
   private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
         LoginFrame l = new LoginFrame();
      l.show();
      dispose();
   }
    private void SignUpButtonActionPerformed(java.awt.event.ActionEvent evt) {
      SignUpFrame s = new SignUpFrame();
      s.show();
      dispose();
   }
   public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new MainFrame().setVisible(true);
         }
      });
}
```

2LOGIN FRAME

```
public class LoginFrame extends javax.swing.JFrame {
   LoginForm log = new LoginForm();
   public LoginFrame() {
      initComponents();
      log.connectToDatabase();
   }
   private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
       if(UsernameField.getText().equals("")){
             JOptionPane.showMessageDialog(null, "Please fill out username");
          }
          else if(PasswordField.getText().equals("")){
             JOptionPane.showMessageDialog(null,"Please fill out password");
          }
          else{
             log.Login();
             dispose();
          } }
   private void ShowCheckBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(ShowCheckBox.isSelected()){
         PasswordField.setEchoChar((char) 0);      }
      else{
         PasswordField.setEchoChar('*');
      }}
```

```
private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
    new MainFrame().setVisible(true);
}
public class LoginForm{
  //CONNECTION CODE
public void Login(){
    String username = UsernameField.getText();
    String password = new String(PasswordField.getPassword());
    try{
        String query = "select * from users where username = ? and password = ?";
        //prepareStatement pt=conn.prepareStatement(query);
        PreparedStatement pt = conn.prepareStatement(query);
        pt.setString(1,username);
        pt.setString(2,password);
        ResultSet rs = pt.executeQuery();
        if(rs.next()){
            int userid = rs.getInt("userid");
            Session.currentUserId = userid;
            JOptionPane.showMessageDialog(null, "login successfully!");
            if(username.equalsIgnoreCase("Admin")){
            new AdminFrame().setVisible(true);
            }
      else{
```

```
                new UserCardPage().setVisible(true);
            } }else{
            JOptionPane.showMessageDialog(null, "invalid");
        }           }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, "Login failed!\n" + e.getMessage());
        e.printStackTrace()
    }}}
  public static void main(String args[]) {
     java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new LoginFrame().setVisible(true);
        }   });}
public class Session {
   public static int currentUserId;
}
```

3. REGISTER FRAME

```
public class SignUpFrame extends javax.swing.JFrame {
  public SignUpFrame() {
      initComponents();
   }
public static boolean isValidPassword(String password) {
      if (password == null || password.length() < 6) {
          return false; }
```

```
        boolean hasUppercase = false;
        boolean hasLowercase = false;
        int digitCount = 0;
        for (char c : password.toCharArray()) {
            if (Character.isDigit(c)) {
                digitCount++;
            } else if (Character.isUpperCase(c)) {
                hasUppercase = true;
            } else if (Character.isLowerCase(c)) {
                hasLowercase = true;
            }}return digitCount >= 3 && hasUppercase && hasLowercase;
    }
    private void SignUPButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String username = UserNameField.getText();
        String email = EmailField.getText();
        String password;
        password = new String(PasswordField.getPassword());
        if (email.isEmpty() || username.isEmpty() || password.isEmpty()) {
            JOptionPane.showMessageDialog(this, "All fields are required!", "Error",
JOptionPane.ERROR_MESSAGE);
          }  else if (!email.matches("^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-z]{2,6}$")) {
            JOptionPane.showMessageDialog(this, "Invalid Email Address", "Error",
JOptionPane.ERROR_MESSAGE);
        }  else if (!isValidPassword(password)) {
            JOptionPane.showMessageDialog(this, "Requirement:Atleast 6 characters,1 Uppercase,1
Lowercase", "Error", JOptionPane.ERROR_MESSAGE);
```

```
            }    else{ s.reg();}

        JOptionPane.showMessageDialog(this, "Registration Successful!", "Success",
JOptionPane.INFORMATION_MESSAGE);

           MainFrame m = new MainFrame();

            m.show();

            dispose();

    }

    private void ShowCheckBoxActionPerformed(java.awt.event.ActionEvent evt) {

        ShowCheckBox.setBackground(SignUpPanel.getBackground());

        if(ShowCheckBox.isSelected()){

            PasswordField.setEchoChar((char) 0); //show password        }

        else{

            PasswordField.setEchoChar('*');//Hide Password}}

    private void UserNameFieldFocusGained(java.awt.event.FocusEvent evt) {

             if(UserNameField.getText().equals("Enter User Name")){

            UserNameField.setText("");

            UserNameField.setForeground(Color.BLACK);   }

    }

    private void UserNameFieldFocusLost(java.awt.event.FocusEvent evt) {

        if(UserNameField.getText().isEmpty()){

            UserNameField.setText("Enter User Name");

            UserNameField.setForeground(Color.GRAY);

        }}


    private void EmailFieldFocusGained(java.awt.event.FocusEvent evt) {                                :
```

```
    if(EmailField.getText().equals("name@gmail.com")){
        EmailField.setText("");
        EmailField.setForeground(Color.BLACK);
    }
}
private void EmailFieldFocusLost(java.awt.event.FocusEvent evt) {
        if(EmailField.getText().isEmpty()){
        EmailField.setText("name@gmail.com");
        EmailField.setForeground(Color.GRAY);
    }   }
private void PhoneFieldFocusGained(java.awt.event.FocusEvent evt) {
        if(PhoneField.getText().trim().equals("Enter Phone Number")){
        PhoneField.setText("");
        PhoneField.setForeground(Color.BLACK);
    }   }
private void PhoneFieldFocusLost(java.awt.event.FocusEvent evt) {
    if(PhoneField.getText().trim().isEmpty()){
        PhoneField.setText("Enter Phone Number");
        PhoneField.setForeground(Color.GRAY);
    }   }
private void ConformCheckBoxActionPerformed(java.awt.event.ActionEvent evt) {
    if(ConformCheckBox.isSelected()){
        ConformPasswordField.setEchoChar((char) 0); //show password
    }       else{
```

```
            ConformPasswordField.setEchoChar('*');//Hide Password

        } }

    private void SignUPButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        String username = UserNameField.getText();

        String email = EmailField.getText();

        String password;

        password = new String(PasswordField.getPassword());

        if (email.isEmpty() || username.isEmpty() || password.isEmpty()) {

            JOptionPane.showMessageDialog(this, "All fields are required!", "Error",
JOptionPane.ERROR_MESSAGE)

        }}

        else if (!email.matches("^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-z]{2,6}$")) {

            JOptionPane.showMessageDialog(this, "Invalid Email Address", "Error",
JOptionPane.ERROR_MESSAGE);

        }       else if (!isValidPassword(password)) {

            JOptionPane.showMessageDialog(this, "Requirement:Atleast 6 characters,1 Uppercase,1
Lowercase", "Error", JOptionPane.ERROR_MESSAGE);

            //return;

        }       else{

            s.reg();

     }}

    private void ShowCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {

            ShowCheckBox.setBackground(SignUpPanel.getBackground());

        if(ShowCheckBox.isSelected()){

            PasswordField.setEchoChar((char) 0); //show password
```

```
    }else{    PasswordField.setEchoChar('*');//Hide Password
   }    }

private void UserNameField1FocusGained(java.awt.event.FocusEvent evt) {
    if(UserNameField.getText().equals("Enter User Name")){
        UserNameField.setText("");
        UserNameField.setForeground(Color.BLACK);}  }
private void UserNameField1FocusLost(java.awt.event.FocusEvent evt) {
    if(UserNameField.getText().isEmpty()){
        UserNameField.setText("Enter User Name");
        UserNameField.setForeground(Color.GRAY);
    }
}
private void EmailField1FocusGained(java.awt.event.FocusEvent evt) {
    if(EmailField.getText().equals("name@gmail.com")){
        EmailField.setText("");
        EmailField.setForeground(Color.BLACK);
    }
}
private void EmailField1FocusLost(java.awt.event.FocusEvent evt) {
    if(EmailField.getText().isEmpty()){
        EmailField.setText("name@gmail.com");
        EmailField.setForeground(Color.GRAY);
    } }
```

```
    private void PhoneField1FocusGained(java.awt.event.FocusEvent evt) {
        // TODO add your handling code here:
        if(PhoneField.getText().trim().equals("Enter Phone Number")){
            PhoneField.setText("");
            PhoneField.setForeground(Color.BLACK);
    } }
    private void PhoneField1FocusLost(java.awt.event.FocusEvent evt) {
        if(PhoneField.getText().trim().isEmpty()){
            PhoneField.setText("Enter Phone Number");
            PhoneField.setForeground(Color.GRAY);
        }}
    private void ConformCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {
        if(ConformCheckBox.isSelected()){
            ConformPasswordField.setEchoChar((char) 0);   }
        else{
            ConformPasswordField.setEchoChar('*');
        }  }
        public class register{
        private Connection conn ;
        public register(){
             connectToDatabase();}
public void reg(){
        String username=UserNameField.getText();
        String gmail=EmailField.getText();
```

```
        String phnno=PhoneField.getText();
        String password=new String(PasswordField.getPassword());
          String role="customer";
        try{
            String query="insert into users(username,password,emailid,phone,roles)values(?,?,?,?,?)";
            PreparedStatement pt = conn.prepareStatement(query);
            pt.setString(1,username);
            pt.setString(2,password);
            pt.setString(3,gmail);
            pt.setString(4,phnno);
            pt.setString(5,role);
            pt.executeUpdate();
        }
        catch(Exception e){
            JOptionPane.showMessageDialog(null, "Registration failed!\n" + e.getMessage());
            e.printStackTrace();}
    }}
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new SignUpFrame().setVisible(true)          }});
    }
```

4. DISPLAY CARDS:

```
import Website.LoginFrame.Session;
```

```
import javax.swing.*;
import java.awt.*;
import java.sql.*;
public class UserCardPage extends JFrame {
    private Connection conn;
    private JTabbedPane tabbedPane;
    public UserCardPage() {
        setSize(1280, 910);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
JPanel topPanel = new JPanel(new BorderLayout());
topPanel.setBackground(Color.WHITE);
topPanel.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
JLabel titleLabel = new JLabel("Geetha Cards");
titleLabel.setForeground(new Color(0, 0, 128));
titleLabel.setFont(new Font("Edwardian Script ITC", Font.BOLD, 48));
topPanel.add(titleLabel, BorderLayout.WEST);
JPanel rightButtonsWrapper = new JPanel();
rightButtonsWrapper.setLayout(new BoxLayout(rightButtonsWrapper, BoxLayout.Y_AXIS));
rightButtonsWrapper.setBackground(Color.WHITE);
rightButtonsWrapper.setAlignmentY(Component.CENTER_ALIGNMENT);
JPanel rightButtons = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10, 0));
rightButtons.setBackground(Color.WHITE);
JButton cartButton = new JButton("CART");
```

```
cartButton.setFont(new Font("Segoe UI Black", Font.PLAIN, 18));
cartButton.setBorder(null);
cartButton.addActionListener(e -> {
    dispose();
    new ShoppingCartApp(Session.currentUserId).setVisible(true);
});
rightButtons.add(cartButton);
JButton logoutButton = new JButton("LOGOUT");
logoutButton.setFont(new Font("Segoe UI Black", Font.PLAIN, 18));
logoutButton.setBorder(null);
logoutButton.addActionListener(e -> {
    dispose();
    new LoginFrame().setVisible(true);
});
rightButtons.add(logoutButton);
rightButtonsWrapper.add(Box.createVerticalGlue());
rightButtonsWrapper.add(rightButtons);
rightButtonsWrapper.add(Box.createVerticalGlue());
topPanel.add(rightButtonsWrapper, BorderLayout.EAST);
        tabbedPane = new JTabbedPane();
        tabbedPane.setBackground(Color.WHITE);
        connectDatabase();
        tabbedPane.addTab("All Cards", createCardPanel("All"));
        tabbedPane.addTab("Hindu Cards", createCardPanel("Hindu"));
```

```
        tabbedPane.addTab("Muslim Cards", createCardPanel("Muslim"));
        tabbedPane.addTab("Christian Cards", createCardPanel("Christian"));
        setLayout(new BorderLayout());
        add(topPanel, BorderLayout.NORTH);
        add(tabbedPane, BorderLayout.CENTER);
    }
    private JPanel createCardPanel(String typeFilter) {
        JPanel cardPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 20));
        cardPanel.setBackground(Color.WHITE);
        try {
            CallableStatement cs = conn.prepareCall("{CALL GetCardsByType(?)}");
            cs.setString(1, typeFilter);
            ResultSet rs = cs.executeQuery();
            while (rs.next()) {
                int cardId = rs.getInt("card_id");
                String cardName = rs.getString("name");
                float cardPrice = rs.getFloat("price");
                JButton cardButton = new JButton("<html><center>" + cardName + "<br>Price: " + cardPrice
+ "</center></html>");
                cardButton.setPreferredSize(new Dimension(200, 200));
                cardButton.setBackground(Color.LIGHT_GRAY);
                String imagePath = rs.getString("image_path");
                if (imagePath != null && !imagePath.isEmpty()) {
                    ImageIcon icon = new ImageIcon(imagePath);
                    Image img = icon.getImage().getScaledInstance(180, 120, Image.SCALE_SMOOTH);
```

```
                cardButton.setIcon(new ImageIcon(img));
                cardButton.setHorizontalTextPosition(SwingConstants.CENTER);
                cardButton.setVerticalTextPosition(SwingConstants.BOTTOM);
            }
            cardButton.addActionListener(e -> {
                dispose();
                new CardFrame(cardId).setVisible(true);
            });
            cardPanel.add(cardButton);
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error loading cards: " + e.getMessage());
    }

    return cardPanel;
  }
  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new UserCardPage().setVisible(true));
  }
}
```

5. CARD DISPLAY

```
public class CardFrame extends javax.swing.JFrame {
  int id;
  public CardFrame(int id) {
```

```
        initComponents();
        this.id = id;
        fetchCardDetails(id);
    }
    private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
            this.dispose();
            new UserCardPage().setVisible(true);
    }
    private void CartButtonActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            // Database connection
            int userid = Session.currentUserId;
            String name = NameLabel.getText();
            String cardName = name.replace("Name: ", "").trim();
            System.out.println("Card Name: " + cardName);
            System.out.println("Card Name Length: " + cardName.length());
            String priceText = PriceDispLabel.getText().replaceAll("[^\\d.]", "");
            double price = Double.parseDouble(priceText);
            int quantity = 100;
            double total = price * quantity;
            String query = "INSERT INTO cart (userid, card_id, name, price, quantity, total) VALUES
(?, ?, ?, ?, ?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(query);
            pstmt.setInt(1, userid);
            pstmt.setInt(2, id);
```

```
        pstmt.setString(3, cardName);
        pstmt.setDouble(4, price);
        pstmt.setInt(5, quantity);
        pstmt.setDouble(6, total);
        pstmt.executeUpdate();
        JOptionPane.showMessageDialog(this, "Card added to cart successfully!");
        conn.close();
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage());
    } }
private void fetchCardDetails(int cardId) {
Connection conn = null;
PreparedStatement ps = null;
ResultSet rs = null;
try { conn = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/project?useSSL=false&allowPublicKeyRetrieval=true",
            "root", "Shin@maya");
    String sql = "SELECT * FROM cards WHERE card_id = ?";
    ps = conn.prepareStatement(sql);
    ps.setInt(1, cardId);
    rs = ps.executeQuery();
    if (rs.next()) {
        NameLabel.setText("Name: " + rs.getString("name"));
        NameLabel.setFont(new java.awt.Font("Segoe UI Black", 1, 22));
```

```
        PriceDispLabel.setText("₹ " + rs.getFloat("price"));
        PriceDispLabel.setFont(new java.awt.Font("Segoe UI", 0, 18));
        String desc = rs.getString("descriptions");
        if (desc != null && !desc.isEmpty()) {
           DescriptionAreaLabel.setText("<html><p style=\"width:470px;\">" + desc + "</p></html>");
           DescriptionAreaLabel.setFont(new java.awt.Font("Segoe UI", 0, 16));
        }
        String imagePath = rs.getString("image_path");
        if (imagePath != null) {if (!imagePath.isEmpty()) {
                ImageIcon icon = new ImageIcon(imagePath);
                Image img = icon.getImage().getScaledInstance(360, 360, Image.SCALE_SMOOTH);
                JLabel imageLabel = new JLabel(new ImageIcon(img));
                imageLabel.setHorizontalAlignment(SwingConstants.CENTER);

                CardPanel.removeAll();
                CardPanel.setLayout(new BorderLayout());
                CardPanel.add(imageLabel, BorderLayout.CENTER);
                CardPanel.revalidate();
                CardPanel.repaint();
           }        }} else {
        JOptionPane.showMessageDialog(this, "Card not found.");
      }
    } catch (SQLException e) {
      JOptionPane.showMessageDialog(this, "Database error: " + e.getMessage());
```

```
    } finally {
        try {
            if (rs != null) rs.close();
            if (ps != null) ps.close();
            if (conn != null) conn.close();
        } catch (SQLException ex) {
            Logger.getLogger(CardFrame.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {


            }
        });
```

6. }**SHOPPING CART FRAME:**

```
private void PurchaseButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        saveOrderToDatabase();


    }
    private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        this.dispose();
```

```
        new ShoppingCartPage(UserSession.currentUserId).setVisible(true);
    }
    private void connectDatabase() {
    try {
        Connection con = conn.getConnection();
        if (con != null) {
            System.out.println(" Connected to DB from PurchasePage");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}



    private void loadCartData() {
        try {
            CallableStatement ps = conn.prepareCall("{CALL GetCartItemsByUserId(?)}");
            ps.setInt(1, userId);
            ResultSet rs = ps.executeQuery();
            DefaultTableModel model = (DefaultTableModel) CardTable.getModel();
            model.setRowCount(0);
            while (rs.next()) {
                String name = rs.getString("name");
                double price = rs.getDouble("price");
```

```
            int qty = rs.getInt("quantity");
            double total = price * qty;
            // Load image from the path
            String imagePath = rs.getString("image_path");
            ImageIcon icon = new ImageIcon(imagePath);
            Image scaledImage = icon.getImage().getScaledInstance(80, 80, Image.SCALE_SMOOTH);
            icon = new ImageIcon(scaledImage);
            model.addRow(new Object[]{icon, name, "₹" + price, qty, "₹" + total});
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
private void calculateTotals() {
    double subtotal = 0;
    DefaultTableModel model = (DefaultTableModel) CardTable.getModel();
    for (int i = 0; i < model.getRowCount(); i++) {
        String totalStr = model.getValueAt(i, 4).toString().replace("₹", "");
        subtotal += Double.parseDouble(totalStr);
    }
    double tax = subtotal * 0.10;
    double shipping = 50;
    double total = subtotal + tax + shipping;
    SubTotal.setText("₹" + subtotal);
```

```
        Tax.setText("₹" + tax);
        Shipping.setText("50.00");
        TotalAmount.setText("₹" + total);
    }
   private void saveOrderToDatabase() {
    try {
        Connection con = conn.getConnection();
        String country = (String) CountryCombo.getSelectedItem();
        String state = (String) StateCombo.getSelectedItem();
        String address = AddressField.getText();
        String paymentMethod = (String) PayCombo.getSelectedItem();
        Double total = Double.parseDouble(TotalAmount.getText().replace("₹", "").trim());
        if(address.isEmpty()){
           JOptionPane.showMessageDialog(this,"Please Fill Address Field");
        }
        else{
        PreparedStatement ps = con.prepareStatement(
           "INSERT INTO orders(user_id, card_name, quantity, amount, country, state, address,
payment_method, order_date, total_price) " +
           "SELECT ?, c.name, ct.quantity, (c.price * ct.quantity), ?, ?, ?, ?, NOW(), ? " +
           "FROM cart ct JOIN cards c ON ct.card_id = c.card_id WHERE ct.userid = ?"
        );
        ps.setInt(1, userId);         // user_id
        ps.setString(2, country);     // country
        ps.setString(3, state);       // state
```

```
        ps.setString(4, address);     // address
        ps.setString(5, paymentMethod); // payment method
        ps.setDouble(6, total);       // total_price
        ps.setInt(7, userId);         // WHERE ct.userid = ?
        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println(" Order inserted for user ID: " + userId);
            JOptionPane.showMessageDialog(this, " Payment Successful!");
            showBillOnScreen();
            clearCart();
        } else {
            System.out.println(" No rows inserted. Check cart for user ID: " + userId);
        } }
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Can't insert to Orders: " + e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}
private void showBillOnScreen() {
    JFrame billFrame = new JFrame("Invoice");
    billFrame.setSize(600, 600);
    billFrame.setLocationRelativeTo(null);
    JTextArea billArea = new JTextArea();
    billArea.setEditable(false);
```

```
StringBuilder billText = new StringBuilder();
billText.append("========= Geetha Cards - Invoice =========\n");
billText.append("Date: ").append(new Date()).append("\n\n");
billText.append("Delivery Details:\n");
billText.append("Country: ").append(CountryCombo.getSelectedItem()).append("\n");
billText.append("State: ").append(StateCombo.getSelectedItem()).append("\n");
billText.append("Address: ").append(AddressField.getText()).append("\n");
billText.append("Payment Method: ").append(PayCombo.getSelectedItem()).append("\n\n");
billText.append("Purchased Items:\n");
billText.append("-----------------------------------------\n");
billText.append(String.format("%-20s %-10s %-5s %-10s\n", "Card Name", "Price", "Qty", "Total"));
DefaultTableModel model = (DefaultTableModel) CardTable.getModel();
for (int i = 0; i < model.getRowCount(); i++) {
    String name = model.getValueAt(i, 1).toString();
    String price = model.getValueAt(i, 2).toString();
    String qty = model.getValueAt(i, 3).toString();
    String total = model.getValueAt(i, 4).toString();
    billText.append(String.format("%-20s %-10s %-5s %-10s\n", name, price, qty, total));
}
billText.append("-----------------------------------------\n");
billText.append("Subtotal: ").append(SubTotal.getText()).append("\n");
billText.append("Shipping: ₹50.00\n");
billText.append("Tax (10%): ").append(Tax.getText()).append("\n");
billText.append("Total Amount: ").append(TotalAmount.getText()).append("\n");
```

```
    billArea.setText(billText.toString());

    JScrollPane scrollPane = new JScrollPane(billArea);

    billFrame.add(scrollPane);

    billFrame.setVisible(true);

    // --- After showing the bill, ask if they want to send email

    int choice = JOptionPane.showConfirmDialog(null, "Do you want to send the invoice to the customer's
email?", "Send Invoice", JOptionPane.YES_NO_OPTION);

  if (choice == JOptionPane.YES_OPTION) {

    String email = getEmailFromDatabase(UserSession.currentUserId);

    if (email != null && !email.trim().isEmpty()) {

        sendInvoiceEmail(email, billText.toString());  // send without asking manually

    } else {

        JOptionPane.showMessageDialog(null, "Email not found for the user.");

    }}

}private String getEmailFromDatabase(int userId) {

    String email = null;

    try {

        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/your_database",
"your_username", "your_password");

        String query = "SELECT email FROM users WHERE userid = ?";

        PreparedStatement pst = con.prepareStatement(query);

        pst.setInt(1, userId);

        ResultSet rs = pst.executeQuery();

        if (rs.next()) {

            email = rs.getString("email");
```

```
            }
            rs.close();
            pst.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Failed to fetch email: " + e.getMessage());
        }
        return email;
    }
    private void sendInvoiceEmail(String recipientEmail, String invoiceContent) {
        String fromEmail = "geetharanioffsetprinters10@gmail.com";  // Your email
        String password = "fyap xvip iawk knlr";     // App password
        Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "587");
        Session session = Session.getInstance(props, new Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(fromEmail, password);
            }
        });
         try {
```

```
            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress(fromEmail));
            message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(recipientEmail));
            message.setSubject("Your Geetha Cards Invoice");
            message.setText(invoiceContent);  // sending the full invoice text
            Transport.send(message);
            JOptionPane.showMessageDialog(null, "Invoice sent successfully to customer!");
        } catch (MessagingException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Failed to send invoice: " + e.getMessage());
        }
    }
    private void clearCart() {
        try {
            PreparedStatement ps = conn.prepareStatement("DELETE FROM cart WHERE userid = ?");
            ps.setInt(1, userId);
            ps.executeUpdate();
            loadCartData();
            calculateTotals();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public static void main(String args[])
```

```
            java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                //new PurchasePage().setVisible(true);
            }
        });}
public class conn {
    private static Connection con;
    static {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver"); // or your JDBC driver
           con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/project?useSSL=false&allowPublicKeyRetri
eval=true", "root", "Shin@maya");
            System.out.println("Connected successfully!");
        } catch (Exception e) {
            System.out.println("Connection failed: " + e);
        }}
    public static Connection getConnection() {
        return con;



    public static CallableStatement prepareCall(String query) throws SQLException {
        return con.prepareCall(query);
    }
    public static PreparedStatement prepareStatement(String sql) throws SQLException {
```

```
        return con.prepareStatement(sql);          }


    }ADMIN PAGE:
public class AdminFrame extends javax.swing.JFrame {
    public AdminFrame() {
        initComponents();
        loadCards();
    }
    private void LogoutButtonActionPerformed(java.awt.event.ActionEvent evt)
{                                       \
        int confirmLogout = JOptionPane.showConfirmDialog(this,
            "Are you sure you want to logout?",
            "Logout Confirmation",
            JOptionPane.YES_NO_OPTION);
        if (confirmLogout == JOptionPane.YES_OPTION) {
           this.dispose();
            new MainFrame().setVisible(true);
        }  }
private File selectedFile;
    private void AddCardButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String cardName = CardNameText.getText();
    String cardType = (String) cardTypeCombo.getSelectedItem();
    String cardPrice = PriceField.getText();
    String descriptions = DescriptionField.getText();
    if (cardName.isEmpty() || cardPrice.isEmpty() || selectedFile == null || descriptions.isEmpty()) {
```

```
        JOptionPane.showMessageDialog(this, "Please fill all fields and select an image.", "Error",
JOptionPane.ERROR_MESSAGE);

        return;

    }

        try {

            saveCardToDatabase(cardName, cardType, cardPrice, selectedFile.getAbsolutePath(),
descriptions);

        } catch (ClassNotFoundException ex) {

            Logger.getLogger(AdminFrame.class.getName()).log(Level.SEVERE, null, ex);

        }

    //displayCard(cardName, cardType, cardPrice, selectedFile.getAbsolutePath());

   CardNameText.setText("");

    PriceField.setText("");

    PreviewLabel.setIcon(null);

    selectedFile = null;

    }

    private void ChooseURLButtonActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:

        JFileChooser fileChoose =  new JFileChooser();

        fileChoose.setFileFilter(new FileNameExtensionFilter("Image Files", "jpg", "jpeg", "png"));

        int result = fileChoose.showOpenDialog(this);

        if(result == JFileChooser.APPROVE_OPTION){

            selectedFile = fileChoose.getSelectedFile();

            String imgpath = selectedFile.getAbsolutePath();

        ImageIcon cardImage = new ImageIcon(selectedFile.getAbsolutePath());
```

```
            Image img = cardImage.getImage().getScaledInstance(200, 200, Image.SCALE_SMOOTH);
            PreviewLabel.setIcon(new ImageIcon(img));
            ImageField.setText(imgpath);
        }
    }
    private Connection conn ;
    private void saveCardToDatabase(String name, String type, String price, String imagePath, String
descriptions) throws ClassNotFoundException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            String url = "jdbc:mysql://localhost:3306/project?useSSL=false&allowPublicKeyRetrieval=true";
            String user = "root";
            String password = "Shin@maya";
            conn = DriverManager.getConnection(url, user, password);
            String query = "INSERT INTO cards (name, type, price, image_path,descriptions) VALUES
(?, ?, ?, ?, ?)"
            PreparedStatement pstmt = conn.prepareStatement(query);
            pstmt.setString(1, name);
            pstmt.setString(2, type);
            pstmt.setFloat(3, Float.parseFloat(price));
            pstmt.setString(4, imagePath);
            pstmt.setString(5, descriptions);
            pstmt.executeUpdate();
            JOptionPane.showMessageDialog(this, "Card added successfully!");
            conn.close();
```

```
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Database Error: " + e.getMessage());
    }}
private void UpdateButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String cardId = CardIdField.getText().trim();
    String selectedColumn = ColumnCombo.getSelectedItem().toString();
    String newValue = TextField.getText().trim();
    if (cardId.isEmpty() || newValue.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please enter Card ID and Data to change.");
        return;
    }
    String dbColumn = " ";
    switch (selectedColumn) {
        case "Card Name":
            dbColumn = "name";
            break;
        case "Card Price":
            dbColumn = "price";
            break;
        case "Card Image":
            dbColumn = "image_path";
            break;
        case "Card Type":
```

```
            dbColumn = "type";
            break;
        case "Description":
            dbColumn = "descriptions";
            break;
        default:
            JOptionPane.showMessageDialog(null, "Invalid column selected.");
            return;
    }
    // Update the database
    try {
        Connection conn = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/project?useSSL=false&allowPublicKeyRetrieval=true", "root",
"Shin@maya");
        String sql = "UPDATE cards SET " + dbColumn + " = ? WHERE card_id = ?";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, newValue);
        stmt.setString(2, cardId);
        int rowsAffected = stmt.executeUpdate();
        if (rowsAffected > 0) {
            JOptionPane.showMessageDialog(null, "Card updated successfully.");
        } else {
            JOptionPane.showMessageDialog(null, "Card ID not found.");
        }
        stmt.close();
```

```
            conn.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error updating card: " + ex.getMessage());
        }
    }
    private void SaleButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String year = YearField.getText();
        String month = MonthField.getText();
        loadSalesByMonthYear(SaleTable, year, month);
    }
    private void YearFieldFocusGained(java.awt.event.FocusEvent evt) {
        if(YearField.getText().equals("YYYY")){
            YearField.setText("");
            YearField.setForeground(Color.BLACK);
        }
    }
    private void YearFieldFocusLost(java.awt.event.FocusEvent evt) {
        // TODO add your handling code here:
        if(YearField.getText().isEmpty()){
            YearField.setText("YYYY");
            YearField.setForeground(Color.GRAY);
        }
    }
```

```
private void MonthFieldFocusGained(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(MonthField.getText().equals("MM")){
        MonthField.setText("");
        MonthField.setForeground(Color.BLACK);
    }
}
private void MonthFieldFocusLost(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
     if(MonthField.getText().isEmpty()){
        MonthField.setText("MM");
        MonthField.setForeground(Color.GRAY);
    }
}
private void EnableButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int cardId = getSelectedCardId();
        if (cardId != -1) {
            updateCardAvailability(cardId, 1);
        }
}
private void DisableButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
```

```
    int cardId = getSelectedCardId();
        if (cardId != -1) {
            updateCardAvailability(cardId, 0);
        }
 }
private void loadCards() {
    try (Connection conn = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/project?useSSL=false&allowPublicKeyRetrieval=true",
            "root", "Shin@maya")) {
        DefaultTableModel tableModel = (DefaultTableModel) DeleteTable.getModel();
        tableModel.setRowCount(0); // Clear previous rows
        String sql = "SELECT card_id, name, available FROM cards";
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while (rs.next()) {
            int cardId = rs.getInt("card_id");
            String cardName = rs.getString("name");
            int available = rs.getInt("available");
            String availableStatus = (available == 1) ? "Available" : "Unavailable";
            tableModel.addRow(new Object[]{cardId, cardName, availableStatus});
        }
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Database error: " + e.getMessage());
```

```
        }
    }

private int getSelectedCardId() {
    int selectedRow = DeleteTable.getSelectedRow();
    if (selectedRow != -1) {
        Object cardIdObj = DeleteTable.getValueAt(selectedRow, 0);
        if (cardIdObj != null) {
            return Integer.parseInt(cardIdObj.toString());         } else {
            JOptionPane.showMessageDialog(this, "Card ID is null. Please select a valid row.");
            return -1;
        } } else {
        JOptionPane.showMessageDialog(this, "Please select a card first.");
        return -1;
    }}
    private void updateCardAvailability(int cardId, int availableStatus) {
        try (Connection conn = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/project?useSSL=false&allowPublicKeyRetrieval=true",
                "root", "Shin@maya")) {
            String sql = "UPDATE cards SET available = ? WHERE card_id = ?";
            PreparedStatement pst = conn.prepareStatement(sql);
            pst.setInt(1, availableStatus);
            pst.setInt(2, cardId);
            int rowsUpdated = pst.executeUpdate();
```

```
            if (rowsUpdated > 0) {
                String message = (availableStatus == 1) ? "Card enabled successfully!" : "Card disabled
successfully!";
                JOptionPane.showMessageDialog(this, message);
                loadCards();
            } else {
                JOptionPane.showMessageDialog(this, "Failed to update card availability.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Database error: " + e.getMessage());
        }
    }
    public void loadSalesByMonthYear(JTable table, String yearStr, String monthStr) {
    if (yearStr.isEmpty() || monthStr.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please enter both Year and Month.");
        return;
    }
    try {
        int year = Integer.parseInt(yearStr);
        int month = Integer.parseInt(monthStr);
        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/project?useSSL=false&allowPublicKeyRetri
eval=true", "root", "Shin@maya");
        PreparedStatement stmt = conn.prepareStatement(
```

```
            "SELECT order_id, total_price, quantity, order_date FROM orders WHERE YEAR(order_date)
= ? AND MONTH(order_date) = ?"
        );
        stmt.setInt(1, year);
        stmt.setInt(2, month);
        var rs = stmt.executeQuery();
        DefaultTableModel model = new DefaultTableModel();
        model.setColumnIdentifiers(new Object[]{"Order ID", "Amount", "Quantity", "Date"});
        while (rs.next()) {
            String orderId = rs.getString("order_id");
            double amount = rs.getDouble("total_price");
            int qty = rs.getInt("quantity");
            Date date = rs.getDate("order_date");
            model.addRow(new Object[]{orderId, amount, qty, date});
        }table.setModel(model);
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(null, "Year and Month must be numeric values.");
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error: " + ex.getMessage());
    }}
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new AdminFrame().setVisible(true);}});}
```

CHAPTER 5

SCREENSHOTS

MAINPAGE:

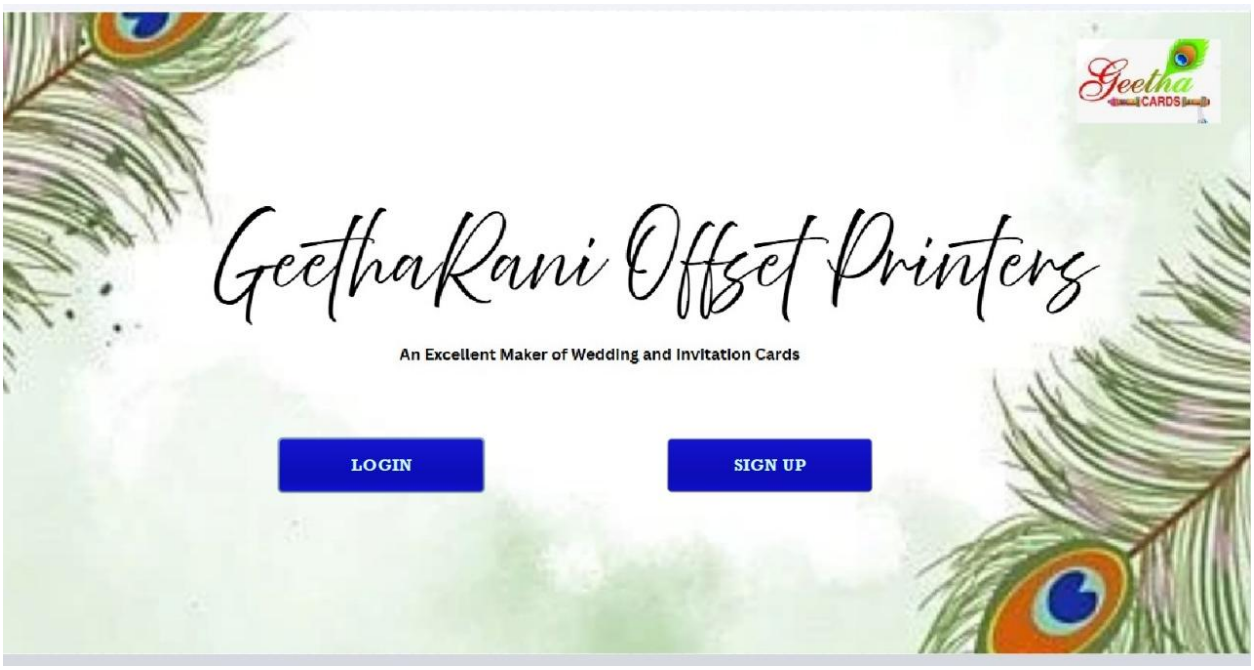


Fig 5.1.1

LOGIN FORM

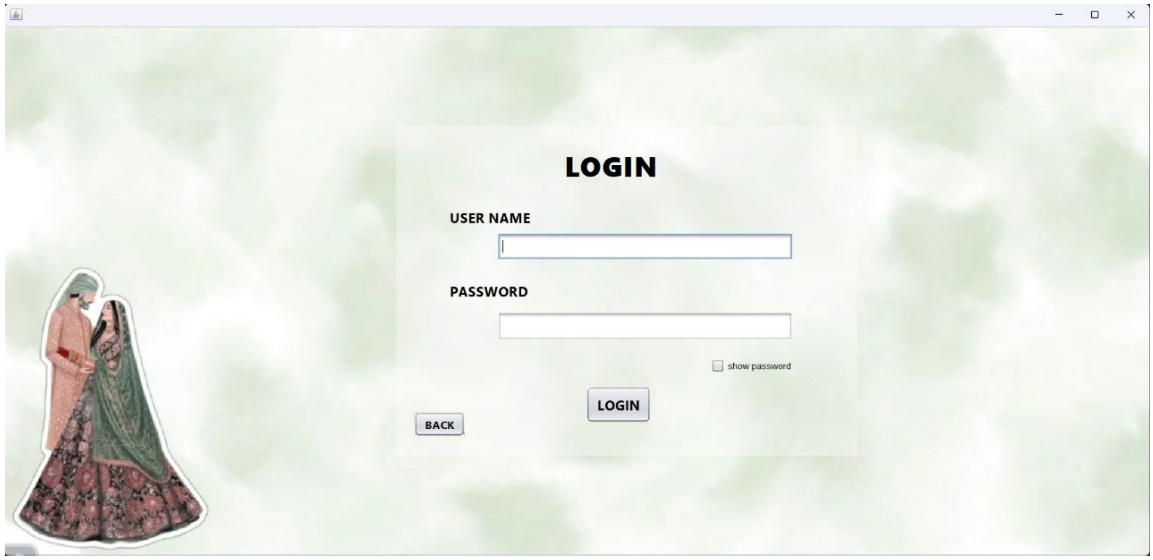


Fig 5.1.2

SIGN UP FRAME

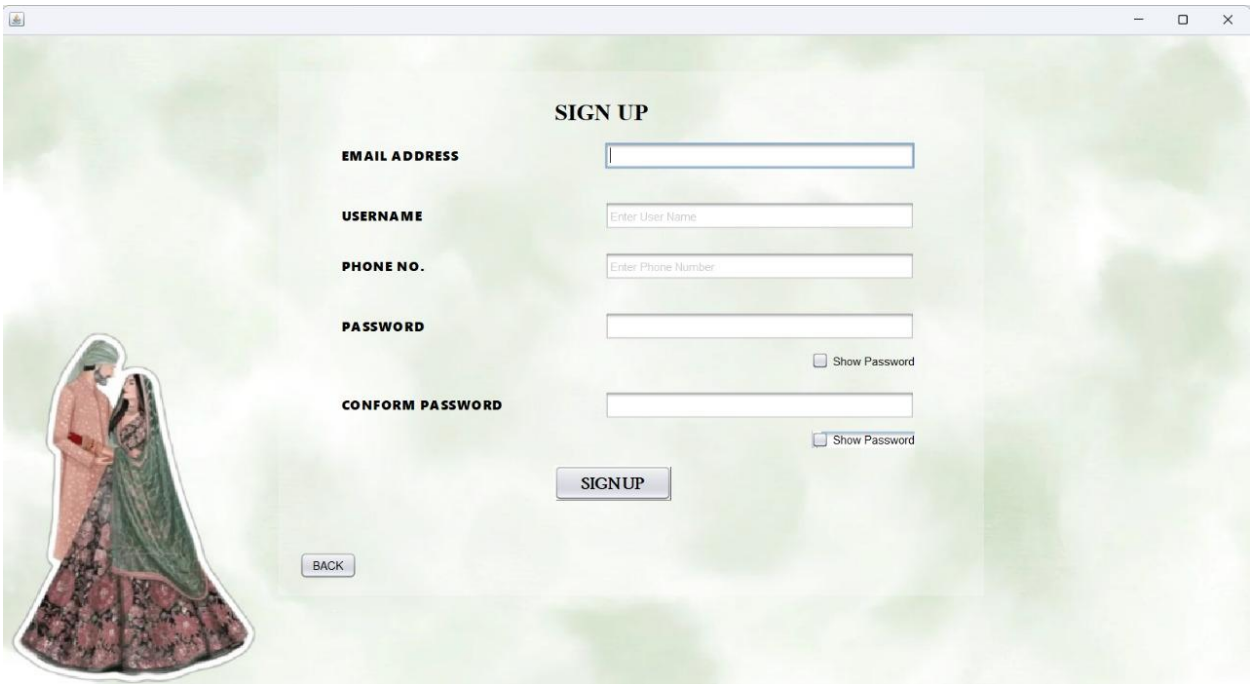


Fig 5.1.3

MAIN CARD FRAME:

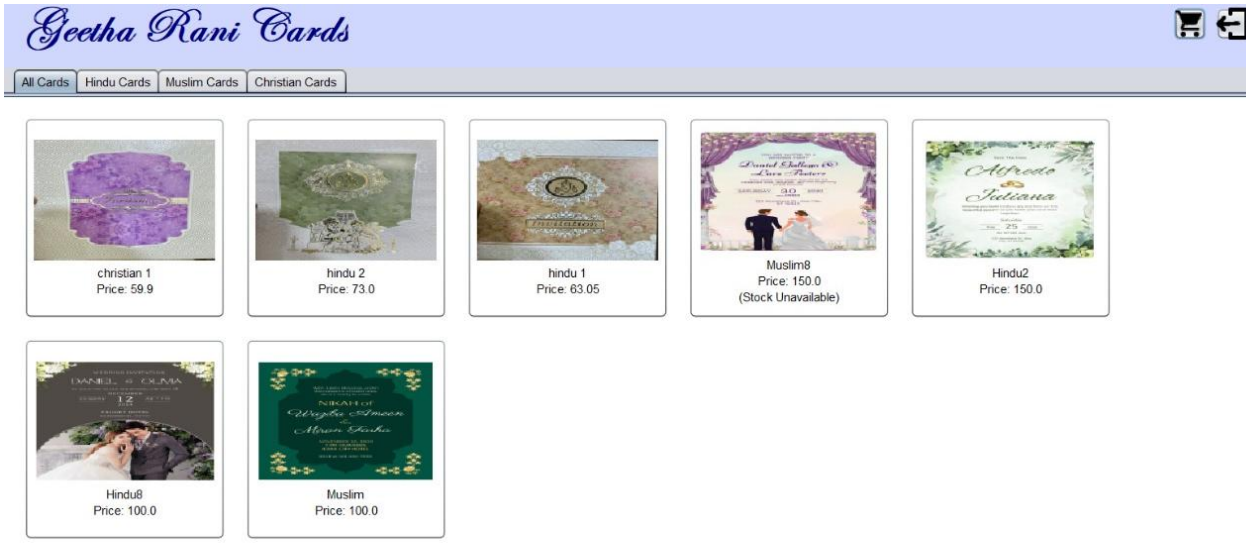


Fig 5.1.4

CARD FRAME:

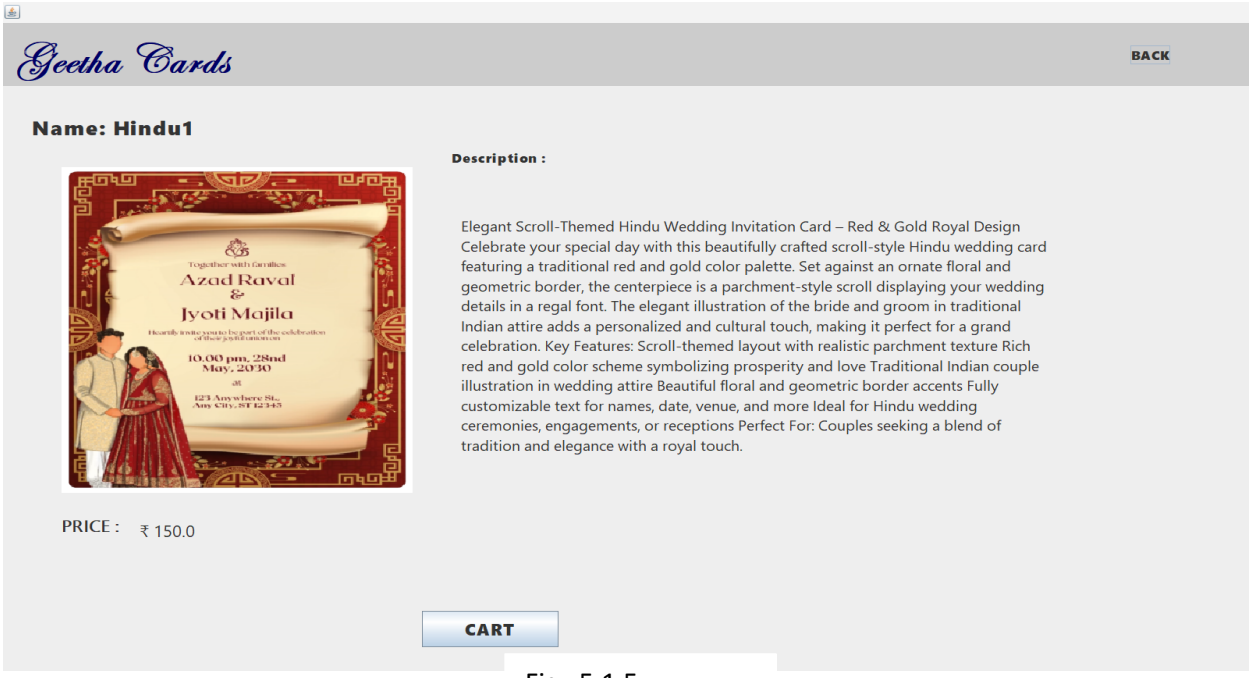


Fig 5.1.5

SHOPPING CART:

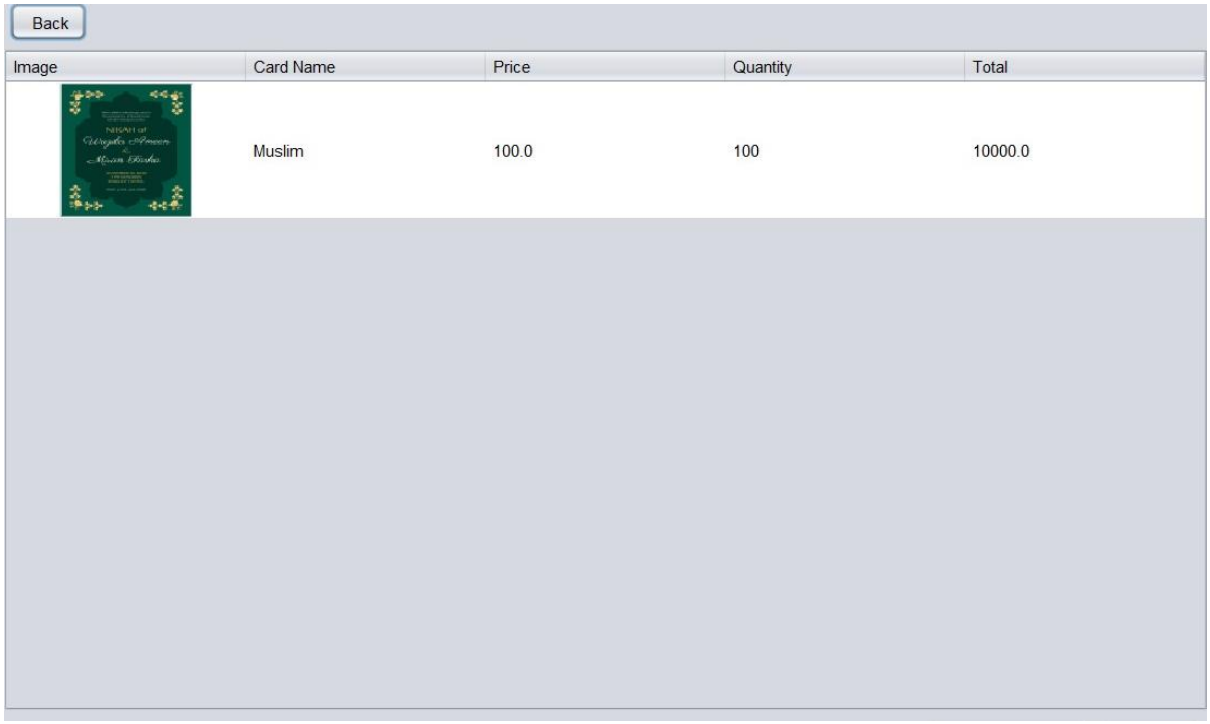


Fig 5.1.6

PURCHASE FRAME:

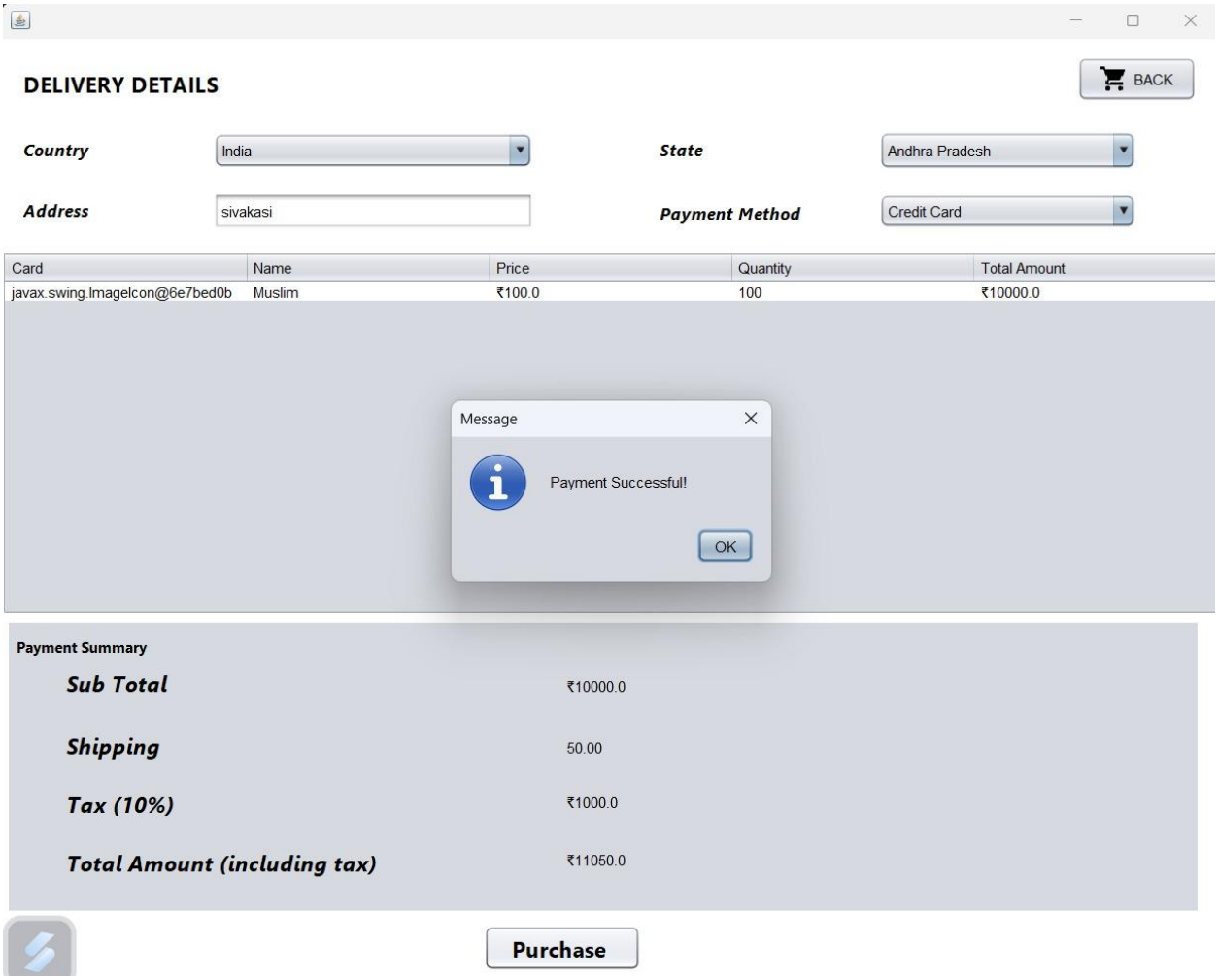


Fig 5.1.7

INVOICE

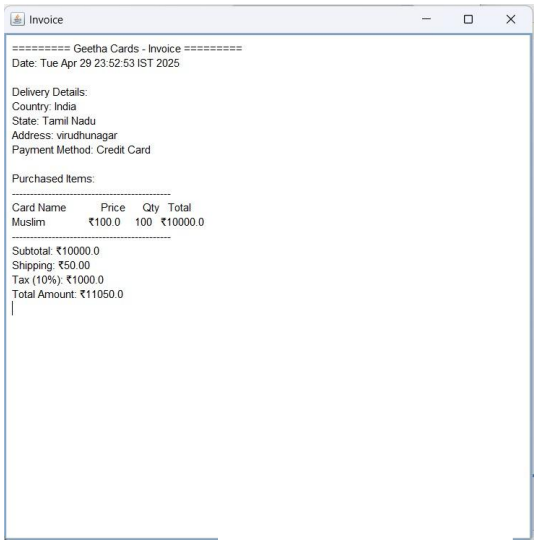


Fig 5.1.8

GMAIL INVOICE :

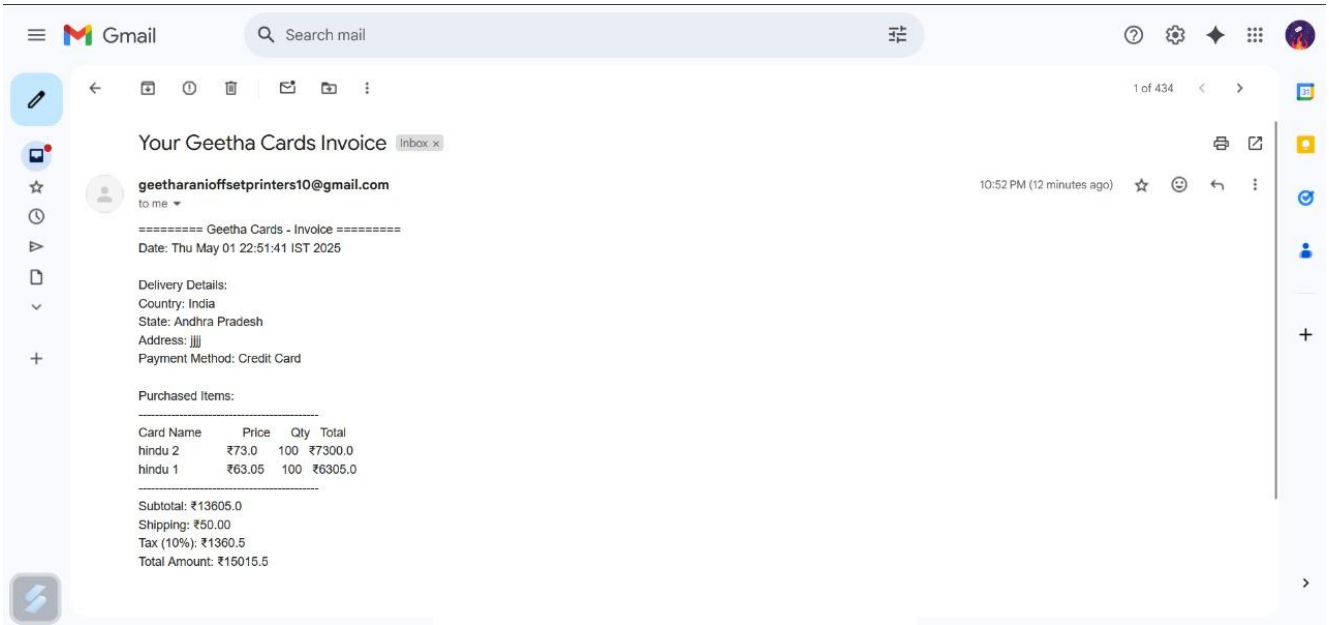
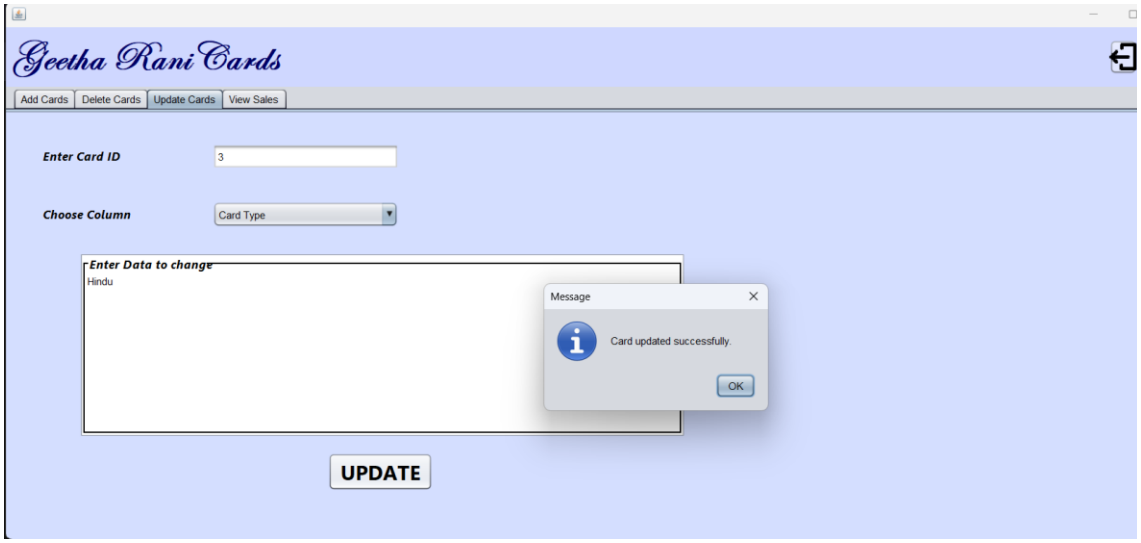


Fig 5.1.9

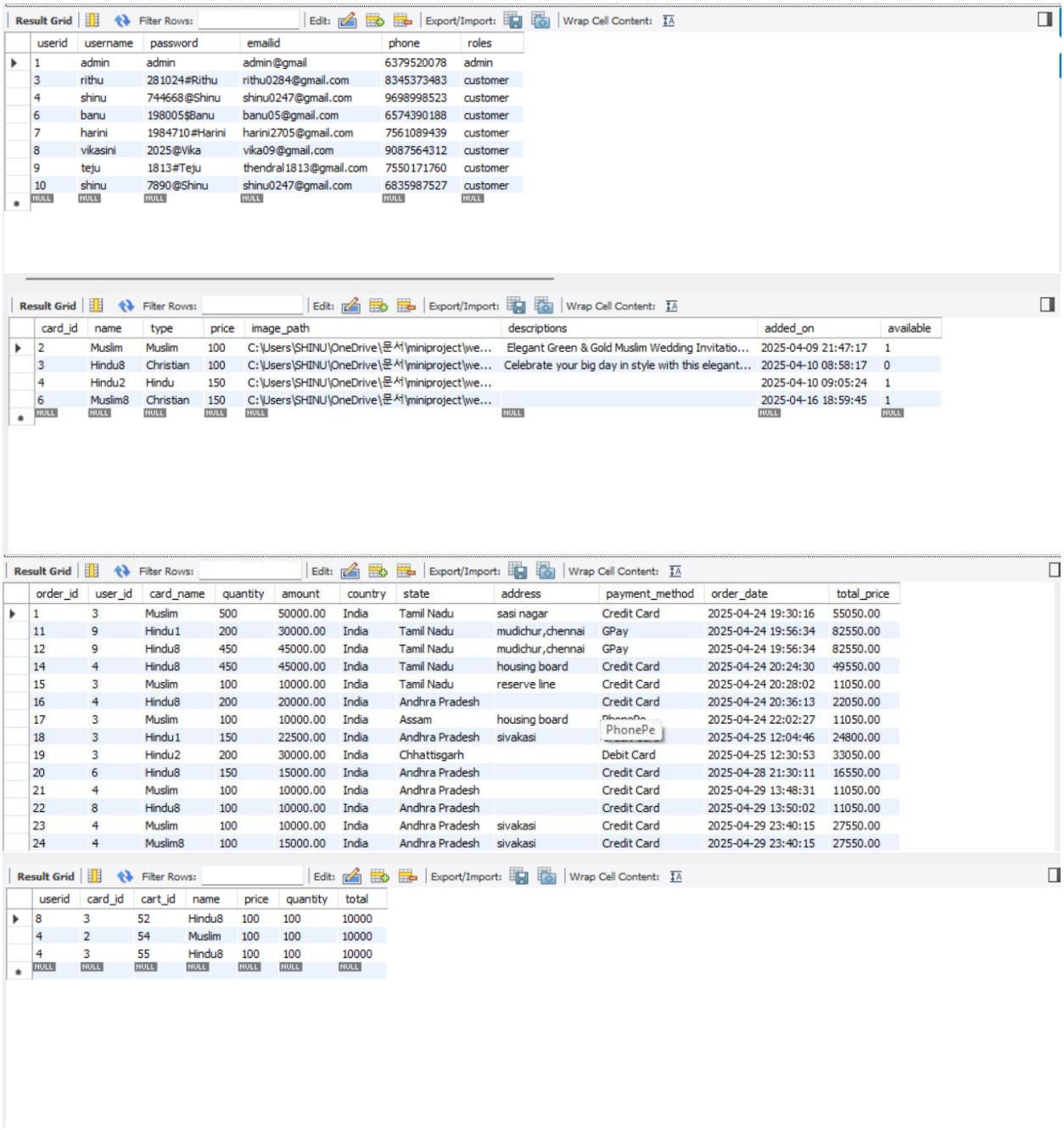
ADMIN PAGE:




Geetha Rani Cards
Add Cards
Delete Cards
Update Cards
View Sales
Card Name
hindu 20
Card Type
Hindu
Hindu
Christian
Muslim
Card Price
63.0
Image
niniproject\wedding cards\HINDU\hindu 18.jpg
Choose URL
Add Card
Image Preview
Description
Celebrate your special day with elegance through this beautifully crafte
Product Details :-
Card Material :- Premium Metallic Board
Orientation :- Landscape
Card Style :- Single Fold
Dimension :- 8.5" X 8.5"
Geetha Rani Cards
Add Cards
Delete Cards
Update Cards
View Sales
DELETE CARD
Card ID
Card Name
Status
2 Muslim Available
3 Hindu8 Available
4 Hindu2 Available
6 Muslim8 Available
7 hindu 1 Available
8 hindu 2 Available
9 christian 1 Available
Message
Card enabled successfully!
OK
ENABLE CARD
DISABLE CARD
Geetha Rani Cards
Add Cards
Delete Cards
Update Cards
View Sales
VIEW SALES
Year
2025
Month
04
View Sale
Order ID
Amount
Quantity
Date
1 55050.0 500 2025-04-24
11 82550.0 200 2025-04-24
12 82550.0 450 2025-04-24
14 49550.0 450 2025-04-24
15 11050.0 100 2025-04-24
16 22050.0 200 2025-04-24
17 11050.0 100 2025-04-24
18 24800.0 150 2025-04-25
19 33050.0 200 2025-04-25
20 16550.0 150 2025-04-28
21 11050.0 100 2025-04-29
22 11050.0 100 2025-04-29
23 27550.0 100 2025-04-29
24 27550.0 100 2025-04-29
26 11050.0 100 2025-04-29
27 11050.0 100 2025-04-30
28 27550.0 250 2025-04-30
29 16550.0 100 2025-04-30

Fig 5.1.10

TABLES:



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

userid	username	password	emailid	phone	roles
1	admin	admin	admin@gmail	6379520078	admin
3	rithu	281024#Rithu	rithu0284@gmail.com	8345373483	customer
4	shinu	744668@Shinu	shinu0247@gmail.com	9698998523	customer
6	banu	198005$Banu	banu05@gmail.com	6574390188	customer
7	harini	1984710#Harini	harini2705@gmail.com	7561089439	customer
8	vikasini	2025@Vika	vika09@gmail.com	9087564312	customer
9	teju	1813#Teju	thendral1813@gmail.com	7550171760	customer
10	shinu	7890@Shinu	shinu0247@gmail.com	6835987527	customer

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

card_id	name	type	price	image_path	descriptions	added_on	available
2	Muslim	Muslim	100	C:\Users\SHINU\OneDrive\문서\miniproject\we...	Elegant Green & Gold Muslim Wedding Invitatio...	2025-04-09 21:47:17	1
3	Hindu8	Christian	100	C:\Users\SHINU\OneDrive\문서\miniproject\we...	Celebrate your big day in style with this elegant...	2025-04-10 08:58:17	0
4	Hindu2	Hindu	150	C:\Users\SHINU\OneDrive\문서\miniproject\we...		2025-04-10 09:05:24	1
6	Muslim8	Christian	150	C:\Users\SHINU\OneDrive\문서\miniproject\we...		2025-04-16 18:59:45	1

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

order_id	user_id	card_name	quantity	amount	country	state	address	payment_method	order_date	total_price
1	3	Muslim	500	50000.00	India	Tamil Nadu	sasi nagar	Credit Card	2025-04-24 19:30:16	55050.00
11	9	Hindu1	200	30000.00	India	Tamil Nadu	mudichur,chennai	GPay	2025-04-24 19:56:34	82550.00
12	9	Hindu8	450	45000.00	India	Tamil Nadu	mudichur,chennai	GPay	2025-04-24 19:56:34	82550.00
14	4	Hindu8	450	45000.00	India	Tamil Nadu	housing board	Credit Card	2025-04-24 20:24:30	49550.00
15	3	Muslim	100	10000.00	India	Tamil Nadu	reserve line	Credit Card	2025-04-24 20:28:02	11050.00
16	4	Hindu8	200	20000.00	India	Andhra Pradesh		Credit Card	2025-04-24 20:36:13	22050.00
17	3	Muslim	100	10000.00	India	Assam	housing board	PhonePe	2025-04-24 22:02:27	11050.00
18	3	Hindu1	150	22500.00	India	Andhra Pradesh	sivakasi		2025-04-25 12:04:46	24800.00
19	3	Hindu2	200	30000.00	India	Chhattisgarh		Debit Card	2025-04-25 12:30:53	33050.00
20	6	Hindu8	150	15000.00	India	Andhra Pradesh		Credit Card	2025-04-28 21:30:11	16550.00
21	4	Muslim	100	10000.00	India	Andhra Pradesh		Credit Card	2025-04-29 13:48:31	11050.00
22	8	Hindu8	100	10000.00	India	Andhra Pradesh		Credit Card	2025-04-29 13:50:02	11050.00
23	4	Muslim	100	10000.00	India	Andhra Pradesh	sivakasi	Credit Card	2025-04-29 23:40:15	27550.00
24	4	Muslim8	100	15000.00	India	Andhra Pradesh	sivakasi	Credit Card	2025-04-29 23:40:15	27550.00

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

userid	card_id	cart_id	name	price	quantity	total
8	3	52	Hindu8	100	100	10000
4	2	54	Muslim	100	100	10000
4	3	55	Hindu8	100	100	10000

Fig 5.1.11

CHAPTER 6

CONCLUSION

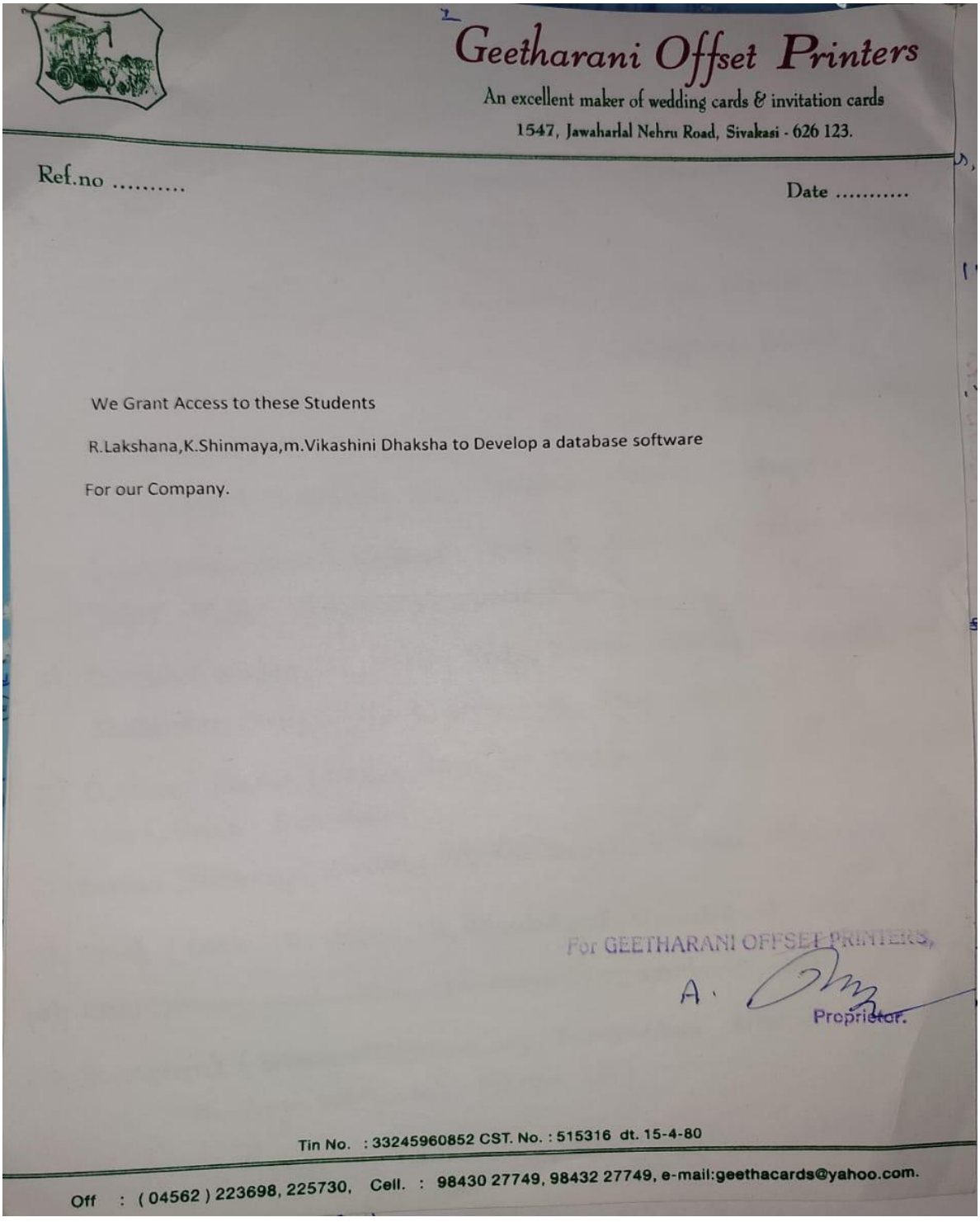
Thus the system allows users to sign up, log in, view cards categorized by religion, add them to a cart, and finalize purchases with address and payment information. At the same time, it provides an admin panel where administrators can add, update, delete cards, and view monthly sales reports, thereby making business management straightforward.Overall, "GeethaRani wedding Cards" successfully automates the traditional wedding card shopping process into a simple, digital format, saving time for customers and providing efficient business tracking for shop owners. The project not only helped in understanding real-time software development but also improved skills in Java, JDBC, Swing, MySQL, and software design practices.

CHAPTER 7

REFERENCES

1. **Java Official Documentation** Used for learning about Java Swing, JDBC, exception handling. https://docs.oracle.com/javase/
2. **MySQL Documentation** database setup, and stored procedures. https://dev.mysql.com/doc/
3. **Stack Overflow** debug issues, understand errors, and find code solutions. https://stackoverflow.com/
4. **GeeksforGeeks** https://www.geeksforgeeks.org/
5. **W3Schools** https://www.w3schools.com/sql/
6. **TutorialsPoint** reference for Java and database integration.
7. **GitHub** For learning structure and flow of similar projects
8. **YouTube Tutorials** For visual guidance on Java GUI building and database connectivity

APPENDIX



Geetharani Offset Printers

An excellent maker of wedding cards & invitation cards

1547, Jawaharlal Nehru Road, Sivakasi - 626 123.

Ref.no .....

Date .....

We Grant Access to these Students

R.Lakshana,K.Shinmaya,m.Vikashini Dhaksha to Develop a database software

For our Company.

For GEETHARANI OFFSET PRINTERS,

A.

Proprietor.

Tin No. : 33245960852 CST. No. : 515316 dt. 15-4-80

Off : (04562) 223698, 225730, Cell. : 98430 27749, 98432 27749, e-mail:geethacards@yahoo.com.

Fig 7.1.1

3

3

3

3

3

3

3