import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.HashMap;

public class Pay extends JFrame {

private CardLayout cardLayout;

private JPanel mainPanel;

// Login/Register data

private HashMap<String, String> userDatabase = new HashMap<>();

private String loggedInUser = null;

// Digital Wallet Data

private JTextArea transactionArea;

private JTextField amountField;

private JLabel nameLabel, userIdLabel, balanceLabel, timeLabel, dateLabel;

private double balance = 0.0;

// Welcome Screen Animation

private JLabel welcomeLabel;

public Pay() {

setTitle("Digital Wallet");

setSize(1000, 700); // Increased window size

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

cardLayout = new CardLayout();

mainPanel = new JPanel(cardLayout);

// Add Welcome, Login/Register, and Wallet Panels

mainPanel.add(createWelcomePanel(), "Welcome");

mainPanel.add(createLoginRegisterPanel(), "LoginRegister");

mainPanel.add(createWalletPanel(), "Wallet");

add(mainPanel);

// Show the Welcome screen initially

cardLayout.show(mainPanel, "Welcome");

}

// Updates the wallet information on the UI

private void updateWalletInfo() {

if (loggedInUser != null) {

nameLabel.setText("Name: " + loggedInUser);

userIdLabel.setText("User ID: " + loggedInUser);

balanceLabel.setText("Balance: $" + String.format("%.2f", balance));

}

}

// Updates the balance display

private void updateBalance() {

balanceLabel.setText("Balance: $" + String.format("%.2f", balance));

}

// Logs transactions in the transaction area

private void logTransaction(String transactionDetails) {

LocalDateTime now = LocalDateTime.now();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

String dateTime = now.format(formatter);

transactionArea.append(transactionDetails + " at " + dateTime + "\n");

}

private JPanel createWelcomePanel() {

JPanel welcomePanel = new JPanel(new BorderLayout());

welcomeLabel = new JLabel("Welcome to Digital Wallet", JLabel.CENTER);

welcomeLabel.setFont(new Font("Arial", Font.BOLD, 36)); // Increased font size

welcomeLabel.setForeground(Color.BLUE);

welcomePanel.add(welcomeLabel, BorderLayout.CENTER);

// Timer to animate the welcome message and switch to Login screen

Timer timer = new Timer(100, new ActionListener() {

private int opacity = 0;

@Override

public void actionPerformed(ActionEvent e) {

if (opacity < 255) {

opacity += 5;

welcomeLabel.setForeground(new Color(0, 0, 255, opacity));

} else {

((Timer) e.getSource()).stop();

cardLayout.show(mainPanel, "LoginRegister"); // Move to Login/Register screen after animation

}

}

});

timer.start();

return welcomePanel;

}

private JPanel createLoginRegisterPanel() {

JPanel loginPanel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(10, 10, 10, 10);

JLabel usernameLabel = new JLabel("Username:");

JLabel passwordLabel = new JLabel("Password:");

JTextField usernameField = new JTextField(15);

JPasswordField passwordField = new JPasswordField(15);

JButton loginButton = new JButton("Login");

JButton registerButton = new JButton("Register");

// Increased font size for labels and buttons

usernameLabel.setFont(new Font("Arial", Font.PLAIN, 20));

passwordLabel.setFont(new Font("Arial", Font.PLAIN, 20));

usernameField.setFont(new Font("Arial", Font.PLAIN, 20));

passwordField.setFont(new Font("Arial", Font.PLAIN, 20));

loginButton.setFont(new Font("Arial", Font.BOLD, 20));

registerButton.setFont(new Font("Arial", Font.BOLD, 20));

gbc.gridx = 0;

gbc.gridy = 0;

loginPanel.add(usernameLabel, gbc);

gbc.gridx = 1;

loginPanel.add(usernameField, gbc);

gbc.gridx = 0;

gbc.gridy = 1;

loginPanel.add(passwordLabel, gbc);

gbc.gridx = 1;

loginPanel.add(passwordField, gbc);

gbc.gridx = 0;

gbc.gridy = 2;

loginPanel.add(loginButton, gbc);

gbc.gridx = 1;

loginPanel.add(registerButton, gbc);

// Login Action

loginButton.addActionListener(e -> {

String username = usernameField.getText();

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

if (userDatabase.containsKey(username) && userDatabase.get(username).equals(password)) {

loggedInUser = username;

balance = 0.0; // Reset balance for simplicity

updateWalletInfo();

cardLayout.show(mainPanel, "Wallet");

} else {

JOptionPane.showMessageDialog(this, "Invalid login credentials!");

}

});

// Register Action

registerButton.addActionListener(e -> {

String username = usernameField.getText();

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

if (userDatabase.containsKey(username)) {

JOptionPane.showMessageDialog(this, "Username already exists!");

} else if (username.isEmpty() || password.isEmpty()) {

JOptionPane.showMessageDialog(this, "Username and Password cannot be empty!");

} else {

userDatabase.put(username, password);

JOptionPane.showMessageDialog(this, "Registration successful! Please log in.");

}

});

return loginPanel;

}

private JPanel createWalletPanel() {

JPanel walletPanel = new JPanel(new BorderLayout());

// Top Panel for User Info

JPanel topPanel = new JPanel(new GridLayout(6, 1)); // Adjusted for an extra label

nameLabel = new JLabel("Name: ");

userIdLabel = new JLabel("User ID: ");

balanceLabel = new JLabel("Balance: $0.00");

timeLabel = new JLabel("Current Time: ");

dateLabel = new JLabel("Current Date: ");

// Increased font size for labels

nameLabel.setFont(new Font("Arial", Font.PLAIN, 22));

userIdLabel.setFont(new Font("Arial", Font.PLAIN, 22));

balanceLabel.setFont(new Font("Arial", Font.PLAIN, 22));

timeLabel.setFont(new Font("Arial", Font.PLAIN, 22));

dateLabel.setFont(new Font("Arial", Font.PLAIN, 22));

topPanel.add(nameLabel);

topPanel.add(userIdLabel);

topPanel.add(balanceLabel);

topPanel.add(timeLabel);

topPanel.add(dateLabel);

walletPanel.add(topPanel, BorderLayout.NORTH);

// Center Panel for Transactions

transactionArea = new JTextArea();

transactionArea.setEditable(false);

transactionArea.setFont(new Font("Arial", Font.PLAIN, 18)); // Increased font size

JScrollPane scrollPane = new JScrollPane(transactionArea);

walletPanel.add(scrollPane, BorderLayout.CENTER);

// Bottom Panel for Actions

JPanel bottomPanel = new JPanel();

amountField = new JTextField(15);

amountField.setFont(new Font("Arial", Font.PLAIN, 20)); // Increased font size for input

bottomPanel.add(new JLabel("Amount:"));

bottomPanel.add(amountField);

JButton depositButton = new JButton("Deposit");

JButton withdrawButton = new JButton("Withdraw");

JButton logoutButton = new JButton("Logout");

JButton exitButton = new JButton("Exit");

// Resize buttons and increase font size

depositButton.setPreferredSize(new Dimension(150, 50));

withdrawButton.setPreferredSize(new Dimension(150, 50));

logoutButton.setPreferredSize(new Dimension(150, 50));

exitButton.setPreferredSize(new Dimension(150, 50));

depositButton.setFont(new Font("Arial", Font.BOLD, 20));

withdrawButton.setFont(new Font("Arial", Font.BOLD, 20));

logoutButton.setFont(new Font("Arial", Font.BOLD, 20));

exitButton.setFont(new Font("Arial", Font.BOLD, 20));

bottomPanel.add(depositButton);

bottomPanel.add(withdrawButton);

bottomPanel.add(logoutButton);

bottomPanel.add(exitButton);

walletPanel.add(bottomPanel, BorderLayout.SOUTH);

// Button Actions

depositButton.addActionListener(e -> {

try {

double amount = Double.parseDouble(amountField.getText());

if (amount > 0) {

balance += amount;

updateBalance();

logTransaction("Deposited $" + amount);

} else {

JOptionPane.showMessageDialog(this, "Enter a positive amount!");

}

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Invalid amount!");

}

});

withdrawButton.addActionListener(e -> {

try {

double amount = Double.parseDouble(amountField.getText());

if (amount > 0 && amount <= balance) {

String recipientId = JOptionPane.showInputDialog(this, "Enter recipient user ID:");

if (userDatabase.containsKey(recipientId)) {

balance -= amount;

updateBalance();

logTransaction("Transferred $" + amount + " to User ID: " + recipientId);

} else {

JOptionPane.showMessageDialog(this, "Recipient User ID not found!");

}

} else {

JOptionPane.showMessageDialog(this, "Invalid amount or insufficient balance!");

}

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Invalid amount!");

}

});

logoutButton.addActionListener(e -> {

loggedInUser = null;

balance = 0.0;

cardLayout.show(mainPanel, "LoginRegister");

});

exitButton.addActionListener(e -> System.exit(0));

// Timer to update time and date every second

Timer timeTimer = new Timer(1000, e -> {

LocalDateTime now = LocalDateTime.now();

DateTimeFormatter timeFormatter = DateTimeFormatter.ofPattern("HH:mm:ss");

DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

timeLabel.setText("Current Time: " + now.format(timeFormatter));

dateLabel.setText("Current Date: " + now.format(dateFormatter));

});

timeTimer.start();

return walletPanel;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new Pay().setVisible(true));

}

}

D:\java>javac Pay.java

D:\java>java Pay