

**Name: Aminah Tariq**

**Ayesha Arshad**

**Adina Zafar**

**Roll no: SP24-BSE-011**

**SP24-BSE-019**

**SP24-BSE-006**

**Course name: Object oriented programming**

**Teacher name: M. Shahid Bhatti**

**Assignment no: 04**

**Due Date: 19-12-2024**

**Blood Bank Management System**

The Blood Bank Management System is designed to streamline the management of blood donations and requests through three primary user roles: \*Admin, \*\*Donor, and \*\*Hospital. The system begins with a \*\*Login Screen\* that directs the user to their respective dashboards based on their credentials. Each dashboard offers tailored functionality to meet the specific needs of that role. Below is an overview of the project and its key components:

**Login Screen:**

The login screen authenticates users and determines their access level. After successful login, users are redirected to their respective dashboards:

1. Admin Dashboard

2. Donor Dashboard

3. Hospital Dashboard

Each dashboard offers unique functionalities to perform various operations within the system.

Admin Dashboard:

The Admin Dashboard serves as the central management hub for the blood bank system. Some key buttons and their functionalities include:

1. Manage Donors:

- Add new donors to the database.

- Update donor details (e.g., contact information, blood type).

- Remove inactive donors from the list.

2. Manage Hospitals:

- Register new hospitals in the system.

- Monitor hospital requests and track fulfillment status.

3. Inventory Management:

- View the availability of blood by type (e.g., A+, B-, AB+).

- Update stock after donations or blood usage.

4. Generate Reports:

- Create monthly or yearly summaries of donations and requests.

- Analyze trends in blood donation and usage.

**Donor Dashboard:**

**The Donor Dashboard is designed for individuals who wish to contribute blood. Key buttons include:**

**1. Schedule Donation:**

**- Select a date and time for donating blood.**

**- View nearby blood bank locations.**

**2. View Donation History:**

**- Display a record of past donations, including dates and blood volumes.**

**3. Eligibility Check:**

**- Determine if the donor is eligible to donate based on factors like age, weight, and health conditions.**

**Hospital Dashboard:**

The Hospital Dashboard facilitates the process of requesting and managing blood supplies for medical purposes. Key buttons include:

1. Request Blood

- Submit requests for specific blood types and quantities.

- View the status of pending or approved requests.

**2. View Request History**

- Display a log of past blood requests and their fulfillment status.

**3. Emergency Request**

- Trigger urgent blood requests for critical patients.

- Automatically notify admins for expedited processing.

**Additional Features:**

**1. Search Functionality(Available on All Dashboards)**

- Search for blood types, donors, or hospitals using specific criteria.

**2. Notifications:**

- Notify donors about upcoming donation drives.

- Alert hospitals about low stock levels for specific blood types.

This Blood Bank Management System ensures efficient operations by organizing donor registrations, inventory management, and hospital requests under a structured and user-friendly interface. The project highlights the importance of role-based access and real-time updates to meet the critical needs of the healthcare industry.

**SOURCE CODE:**

**Admin Dashboard.java**

package com.example.bloodbankjava;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.control.\*;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.VBox;  
import javafx.application.Application;  
import javafx.scene.Scene;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.layout.\*;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.stage.Stage;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.scene.layout.BorderPane;  
import java.util.List;  
  
public class AdminDashboard extends Application {  
 private User admin;  
 private VBox mainContent;  
 private VBox homeScreen;  
 public AdminDashboard(User admin) {  
 this.admin = admin;  
 }  
 @Override  
 public void start(Stage primaryStage) {  
 VBox sidebar = new VBox(15);  
 sidebar.setPadding(new Insets(20));  
 sidebar.setPrefWidth(225);  
 sidebar.setStyle("-fx-background-color: #C41E3A;");  
 Label titleLabel = new Label("Blood Bank Management");  
 titleLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 14));  
 titleLabel.setTextFill(Color.*WHITE*);  
 Button homeBtn = createSidebarButton("Home");  
 Button donationsBtn = createSidebarButton("Blood Donations");  
 Button requestsBtn = createSidebarButton("Requests");  
 Button usersBtn = createSidebarButton("Users");  
 Button logoutBtn = createSidebarButton("Logout");  
  
 sidebar.getChildren().addAll(  
 titleLabel, new Separator(), homeBtn, donationsBtn, requestsBtn, usersBtn, logoutBtn  
 );  
 mainContent = new VBox(15);  
 mainContent.setPadding(new Insets(20));  
 mainContent.setAlignment(Pos.*TOP\_LEFT*);  
 homeScreen = createHomeScreen();  
 mainContent.getChildren().setAll(homeScreen);  
 BorderPane root = new BorderPane();  
 root.setLeft(sidebar);  
 root.setCenter(mainContent);  
  
 Scene scene = new Scene(root, 900, 600);  
 primaryStage.setTitle("Admin Dashboard");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 homeBtn.setOnAction(e -> mainContent.getChildren().setAll(homeScreen));  
 donationsBtn.setOnAction(e -> mainContent.getChildren().setAll(createDonationManagementScreen()));  
 requestsBtn.setOnAction(e -> mainContent.getChildren().setAll(createRequestManagementScreen()));  
 usersBtn.setOnAction(e -> mainContent.getChildren().setAll(createManageUsersScreen()));  
 logoutBtn.setOnAction(e -> {  
 primaryStage.close();  
 new Main().start(new Stage());  
 });  
 }  
  
  
 private Button createSidebarButton(String text) {  
 Button button = new Button(text);  
 button.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 15));  
 button.setTextFill(Color.*WHITE*);  
 button.setStyle("-fx-background-color: transparent;");  
 button.setMaxWidth(Double.*MAX\_VALUE*);  
 button.setAlignment(Pos.*CENTER\_LEFT*);  
 return button;  
 }  
 private VBox createHomeScreen() {  
 VBox home = new VBox(15);  
 home.setPadding(new Insets(20));  
 home.setAlignment(Pos.*TOP\_LEFT*);  
 Label welcomeLabel = new Label("Welcome back Administrator!");  
 welcomeLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
 Label bloodStockTitle = new Label("Available Blood per group in Liters");  
 bloodStockTitle.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 16));  
   
 GridPane bloodGrid = createBloodGrid();  
 HBox statsBox = new HBox(20);  
 statsBox.setAlignment(Pos.*CENTER*);  
 VBox totalDonors = createStatBox("3", "Total Donors", "blue");  
 VBox totalDonatedToday = createStatBox("1", "Total Donated Today", "red");  
 VBox todaysRequests = createStatBox("1", "Today's Requests", "black");  
 VBox approvedRequests = createStatBox("1", "Today's Approved Requests", "green");  
 statsBox.getChildren().addAll(totalDonors, totalDonatedToday, todaysRequests, approvedRequests);  
 home.getChildren().addAll(welcomeLabel, bloodStockTitle, bloodGrid, statsBox);  
 return home;  
 }  
 private GridPane createBloodGrid() {  
 GridPane bloodGrid = new GridPane();  
 bloodGrid.setHgap(20);  
 bloodGrid.setVgap(15);  
 bloodGrid.setPadding(new Insets(10, 0, 10, 0));  
 String[][] bloodData = {  
 {"A+", "0"}, {"B+", "0"}, {"O+", "0.45"}, {"AB+", "0.45"},  
 {"A-", "0"}, {"B-", "0.45"}, {"O-", "0"}, {"AB-", "0"}  
 };  
 int row = 0, col = 0;  
 for (String[] bloodInfo : bloodData) {  
 VBox bloodBox = createBloodBox(bloodInfo[0], bloodInfo[1]);  
 bloodGrid.add(bloodBox, col, row);  
 col++;  
 if (col > 3) {  
 col = 0;  
 row++;  
 }  
 }  
 return bloodGrid;  
 }  
 private VBox createBloodBox(String bloodType, String quantity) {  
 VBox box = new VBox(10);  
 box.setPrefWidth(200);  
 box.setAlignment(Pos.*CENTER*);  
 box.setPadding(new Insets(10));  
 box.setStyle("-fx-border-color: #DDDDDD; -fx-border-radius: 10; -fx-background-radius: 10;");  
 ImageView iconView = new ImageView();  
 Image bloodIcon = new Image("C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\medicine.png"); // Relative file path  
 iconView.setImage(bloodIcon);  
 iconView.setFitHeight(24);  
 iconView.setFitWidth(24);  
 Label quantityLabel = new Label(quantity);  
 quantityLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 18));  
 Label bloodTypeLabel = new Label(bloodType);  
 bloodTypeLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 16));  
 box.getChildren().addAll(quantityLabel, bloodTypeLabel, iconView);  
 return box;  
 }  
 private VBox createManageUsersScreen() {  
 VBox manageUsersLayout = new VBox(20);  
 manageUsersLayout.setPadding(new Insets(20));  
 manageUsersLayout.setAlignment(Pos.*CENTER*);  
  
 Label title = new Label("Manage Users");  
 title.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
 TableView<User> userTable = new TableView<>();  
 TableColumn<User, String> nameColumn = new TableColumn<>("Name");  
 nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));  
 TableColumn<User, String> roleColumn = new TableColumn<>("Role");  
 roleColumn.setCellValueFactory(new PropertyValueFactory<>("role"));  
 TableColumn<User, String> emailColumn = new TableColumn<>("Email");  
 emailColumn.setCellValueFactory(new PropertyValueFactory<>("email"));  
 TableColumn<User, String> passwordColumn = new TableColumn<>("Password");  
 passwordColumn.setCellValueFactory(new PropertyValueFactory<>("password"));  
 userTable.getColumns().addAll( nameColumn, roleColumn, emailColumn, passwordColumn);  
 userTable.getItems().addAll(Userservice.*getAllUsers*());  
  
 Button addUserBtn = new Button("Add User");  
 Button editUserBtn = new Button("Edit User");  
 Button removeUserBtn = new Button("Remove User");  
 String buttonStyle = "-fx-background-color: #b81313; -fx-text-fill: white;";  
 addUserBtn.setStyle(buttonStyle);  
 editUserBtn.setStyle(buttonStyle);  
 removeUserBtn.setStyle(buttonStyle);  
  
 HBox buttonBox = new HBox(15, addUserBtn, editUserBtn, removeUserBtn);  
 buttonBox.setAlignment(Pos.*CENTER*);  
 manageUsersLayout.getChildren().addAll(title, userTable, buttonBox);  
 addUserBtn.setOnAction(e -> {  
 Dialog<User> dialog = createUserDialog("Add New User");  
 dialog.showAndWait().ifPresent(user -> {  
 Userservice.*addUser*(user);  
 userTable.getItems().add(user);  
 });  
 });  
 editUserBtn.setOnAction(e -> {  
 User selectedUser = userTable.getSelectionModel().getSelectedItem();  
 if (selectedUser == null) {  
 showAlert(Alert.AlertType.*WARNING*, "Select a User", "Please select a user to edit.");  
 return;  
 }  
 Dialog<User> dialog = createUserDialog("Edit User", selectedUser);  
 dialog.showAndWait().ifPresent(editedUser -> {  
 Userservice.*updateUser*(editedUser);  
 userTable.getItems().set(userTable.getSelectionModel().getSelectedIndex(), editedUser);  
 });  
 });  
 removeUserBtn.setOnAction(e -> {  
 User selectedUser = userTable.getSelectionModel().getSelectedItem();  
 if (selectedUser == null) {  
 showAlert(Alert.AlertType.*WARNING*, "Select a User", "Please select a user to remove.");  
 return;  
 }  
 if (showConfirmationDialog("Remove User", "Are you sure you want to remove this user?")) {  
 Userservice.*removeUser*(selectedUser);  
 userTable.getItems().remove(selectedUser);  
 }  
 });  
 return manageUsersLayout;  
 }  
 private VBox createStatBox (String value, String description, String color){  
 VBox statBox = new VBox(5);  
 statBox.setAlignment(Pos.*CENTER*);  
 statBox.setPadding(new Insets(10));  
 statBox.setStyle("-fx-border-color: #CCCCCC; -fx-border-radius: 10; -fx-background-radius: 10;");  
 Label valueLabel = new Label(value);  
 valueLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 18));  
 valueLabel.setTextFill(Color.*web*(color));  
 Label descLabel = new Label(description);  
 descLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 12));  
 statBox.getChildren().addAll(valueLabel, descLabel);  
 return statBox;  
 }  
 private Dialog<User> createUserDialog(String title) {  
 return createUserDialog(title, null);  
 }  
 private Dialog<User> createUserDialog(String title, User existingUser) {  
 Dialog<User> dialog = new Dialog<>();  
 dialog.setTitle(title);  
  
 ButtonType saveButtonType = new ButtonType("Save", ButtonBar.ButtonData.*OK\_DONE*);  
 dialog.getDialogPane().getButtonTypes().addAll(saveButtonType, ButtonType.*CANCEL*);  
 GridPane grid = new GridPane();  
 grid.setHgap(10);  
 grid.setVgap(10);  
 grid.setPadding(new Insets(20, 150, 10, 10));  
 TextField nameField = new TextField();  
 nameField.setPromptText("Name");  
 TextField emailField = new TextField();  
 emailField.setPromptText("Email");  
 ComboBox<String> roleComboBox = new ComboBox<>();  
 roleComboBox.getItems().addAll("Admin", "Donor", "Hospital");  
 if (existingUser != null) {  
 roleComboBox.setValue(existingUser.getRole());  
 }  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Password");  
 if (existingUser != null) {  
 nameField.setText(existingUser.getName());  
 emailField.setText(existingUser.getEmail());  
 passwordField.setText(existingUser.getPassword());  
 }  
  
 grid.add(new Label("Name:"), 0, 0);  
 grid.add(nameField, 1, 0);  
 grid.add(new Label("Email:"), 0, 1);  
 grid.add(emailField, 1, 1);  
 grid.add(new Label("Role:"), 0, 2);  
 grid.add(roleComboBox, 1, 2);  
 grid.add(new Label("Password:"), 0, 3);  
 grid.add(passwordField, 1, 3);  
 dialog.getDialogPane().setContent(grid);  
 dialog.setResultConverter(dialogButton -> {  
 if (dialogButton == saveButtonType) {  
 return new User(  
 existingUser != null ? existingUser.getUserId() : getNextUserId(),  
 nameField.getText(),  
 roleComboBox.getValue(),  
 emailField.getText(),  
 passwordField.getText()  
 );  
 }  
 return null;  
 });  
  
 return dialog;  
 }  
 private int getNextUserId() {  
  
 List<User> users = Userservice.*getAllUsers*();  
 return users.stream().mapToInt(User::getUserId).max().orElse(0) + 1;  
 }  
 private boolean showConfirmationDialog(String title, String content) {  
 Alert alert = new Alert(Alert.AlertType.*CONFIRMATION*, content, ButtonType.*YES*, ButtonType.*NO*);  
 alert.setTitle(title);  
 alert.showAndWait();  
 return alert.getResult() == ButtonType.*YES*;  
 }  
 public VBox createRequestManagementScreen() {  
 VBox requestManagementLayout = new VBox(20);  
 requestManagementLayout.setPadding(new Insets(20));  
 requestManagementLayout.setAlignment(Pos.*CENTER*);  
 Label title = new Label("Manage Requests");  
 title.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
  
 TableView<Request> requestTable = new TableView<>();  
 TableColumn<Request, Integer> idColumn = new TableColumn<>("Request ID");  
 idColumn.setCellValueFactory(new PropertyValueFactory<>("requestId"));  
 TableColumn<Request, String> hospitalColumn = new TableColumn<>("Hospital");  
 hospitalColumn.setCellValueFactory(new PropertyValueFactory<>("hospitalName"));  
 TableColumn<Request, String> bloodTypeColumn = new TableColumn<>("Blood Type");  
 bloodTypeColumn.setCellValueFactory(new PropertyValueFactory<>("bloodType"));  
 TableColumn<Request, Integer> quantityColumn = new TableColumn<>("Quantity");  
 quantityColumn.setCellValueFactory(new PropertyValueFactory<>("quantity"));  
 TableColumn<Request, String> urgencyColumn = new TableColumn<>("Urgency");  
 urgencyColumn.setCellValueFactory(new PropertyValueFactory<>("urgency"));  
 TableColumn<Request, String> statusColumn = new TableColumn<>("Status");  
 statusColumn.setCellValueFactory(new PropertyValueFactory<>("status"));  
 statusColumn.setCellFactory(param -> new TableCell<Request, String>() {  
 @Override  
 protected void updateItem(String item, boolean empty) {  
 super.updateItem(item, empty);  
 if (empty || item == null) {  
 setText(null);  
 setStyle("");  
 } else {  
 setText(item);  
 if (item.equals("Pending")) {  
 setStyle("-fx-background-color: yellow;");  
 } else if (item.equals("Approved")) {  
 setStyle("-fx-background-color: green;");  
 } else if (item.equals("Rejected")) {  
 setStyle("-fx-background-color: red;");  
 }  
 }  
 }  
 });  
 TableColumn<Request, String> dateColumn = new TableColumn<>("Request Date");  
 dateColumn.setCellValueFactory(new PropertyValueFactory<>("requestDate"));  
 requestTable.getColumns().addAll(idColumn, hospitalColumn, bloodTypeColumn, quantityColumn, urgencyColumn, statusColumn, dateColumn);  
 requestTable.getItems().addAll(RequestService.*getAllRequests*());  
 Button approveBtn = new Button("Approve Request");  
 Button rejectBtn = new Button("Reject Request");  
 String buttonStyle = "-fx-background-color: #9c0f0f; -fx-text-fill: white;";  
 approveBtn.setStyle(buttonStyle);  
 rejectBtn.setStyle(buttonStyle);  
 HBox buttonBox = new HBox(15, approveBtn, rejectBtn);  
 buttonBox.setAlignment(Pos.*CENTER*);  
 requestManagementLayout.getChildren().addAll(title, requestTable, buttonBox);  
 approveBtn.setOnAction(e -> {  
 Request selectedRequest = requestTable.getSelectionModel().getSelectedItem();  
 if (selectedRequest == null) {  
 showAlert(Alert.AlertType.*WARNING*, "Select a Request", "Please select a request to approve.");  
 return;  
 }  
 selectedRequest.setStatus("Approved");  
 RequestService.*updateRequestStatus*(selectedRequest);  
 requestTable.getItems().set(requestTable.getSelectionModel().getSelectedIndex(), selectedRequest);  
 });  
 rejectBtn.setOnAction(e -> {  
 Request selectedRequest = requestTable.getSelectionModel().getSelectedItem();  
 if (selectedRequest == null) {  
 showAlert(Alert.AlertType.*WARNING*, "Select a Request", "Please select a request to reject.");  
 return;  
 }  
 selectedRequest.setStatus("Rejected");  
 RequestService.*updateRequestStatus*(selectedRequest);  
 requestTable.getItems().set(requestTable.getSelectionModel().getSelectedIndex(), selectedRequest);  
 });  
 return requestManagementLayout;  
 }  
 public VBox createDonationManagementScreen() {  
 VBox donationManagementLayout = new VBox(20);  
 donationManagementLayout.setPadding(new Insets(20));  
 donationManagementLayout.setAlignment(Pos.*CENTER*);  
 Label title = new Label("Manage Donations");  
 title.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
 TableView<Donation> donationTable = new TableView<>();  
 TableColumn<Donation, String> bloodTypeColumn = new TableColumn<>("Blood Type");  
 bloodTypeColumn.setCellValueFactory(new PropertyValueFactory<>("bloodType"));  
 TableColumn<Donation, String> dateColumn = new TableColumn<>("Donation Date");  
 dateColumn.setCellValueFactory(new PropertyValueFactory<>("donationDate"));  
  
 ObservableList<Donation> Data = FXCollections.*observableArrayList*(  
 DonationService.*getAllDonations*()  
 );  
 donationTable.getColumns().addAll(bloodTypeColumn, dateColumn);  
 donationTable.setItems(Data);  
 donationManagementLayout.getChildren().addAll(title, donationTable);  
 return donationManagementLayout;  
 }  
 private void showAlert(String title, String message) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
 private void showAlert(Alert.AlertType type, String title, String content) {  
 Alert alert = new Alert(type);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(content);  
 alert.showAndWait();  
 }  
 public static void main (String[]args){  
 *launch*(args);  
 }  
 }

**Donor Dashboard.java**

package com.example.bloodbankjava;  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.\*;  
import javafx.stage.Stage;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.scene.paint.Color;  
import javafx.scene.control.TableColumn;  
import javafx.scene.control.TableView;  
import javafx.scene.layout.VBox;  
import java.time.LocalDate;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
public class DonorDashboard extends Application {  
  
 private BorderPane rootLayout;  
 private VBox sidebar;  
 private VBox centralPanel;  
 private User loggedInUser;  
  
 public DonorDashboard(User loggedInUser) {  
 this.loggedInUser = loggedInUser;  
 }  
 @Override  
 public void start(Stage primaryStage) {  
 rootLayout = new BorderPane();  
 sidebar = createSidebar();  
 rootLayout.setLeft(sidebar);  
 centralPanel = new VBox();  
 centralPanel.setAlignment(Pos.*CENTER*);  
 centralPanel.setSpacing(20);  
 Label dashboardLabel = new Label("Donor Dashboard");  
 dashboardLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 dashboardLabel.setStyle(  
 "-fx-border-color: black; " +  
 "-fx-border-width: 2px; " +  
 "-fx-background-color: #c87d7d; " +  
 "-fx-padding: 10;"  
 );  
 Label welcomeLabel = new Label("Welcome, " + loggedInUser.getName() + "!");  
 welcomeLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
 Image donorImage = new Image("C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\Blood-1.png"); // Ensure the image is placed in the correct folder  
 ImageView imageView = new ImageView(donorImage);  
 imageView.setFitWidth(600);  
 imageView.setFitHeight(600);  
 imageView.setPreserveRatio(true);  
 StackPane imageContainer = new StackPane(imageView);  
 imageContainer.setStyle(  
 "-fx-border-color: #3c0505; " +  
 "-fx-border-width: 4px; " +  
 "-fx-padding: 10;"  
 );  
 centralPanel.getChildren().addAll(dashboardLabel, welcomeLabel, imageView);  
 rootLayout.setCenter(centralPanel);  
 Scene scene = new Scene(rootLayout, 900, 600);  
 primaryStage.setTitle("Donor Dashboard");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 private VBox createSidebar() {  
 VBox sidebar = new VBox(15);  
 sidebar.setPadding(new Insets(20));  
 sidebar.setStyle("-fx-background-color: #5A0A0A;");  
 Button viewHistory = new Button("View Donation History");  
 Button updateProfile = new Button("Update Profile");  
 Button healthTips = new Button("Health Tips");  
 Button events = new Button("Upcoming Events");  
 Button appointment = new Button("Request Appointment");  
 Button logout = new Button("Logout");  
  
 styleSidebarButton(viewHistory);  
 styleSidebarButton(updateProfile);  
 styleSidebarButton(healthTips);  
 styleSidebarButton(events);  
 styleSidebarButton(appointment);  
 styleSidebarButton(logout);  
   
 updateProfile.setOnAction(e -> showUpdateProfilePanel());  
 viewHistory.setOnAction(e -> showDonationHistory());  
 healthTips.setOnAction(e -> showHealthTips());  
 events.setOnAction(e -> showUpcomingEvents());  
 appointment.setOnAction(e -> showRequestAppointmentForm());  
 logout.setOnAction(e -> logout());  
 sidebar.getChildren().addAll(viewHistory, updateProfile, healthTips, events, appointment, logout);  
 return sidebar;  
 }  
 private void styleSidebarButton(Button button) {  
 button.setPrefWidth(200);  
 button.setStyle("-fx-background-color: white; -fx-text-fill: black; -fx-font-weight: bold;");  
 }  
  
 private void showUpdateProfilePanel() {  
 GridPane formLayout = new GridPane();  
 formLayout.setPadding(new Insets(20));  
 formLayout.setVgap(10);  
 formLayout.setHgap(10);  
 Label banner = new Label("WELCOME JANE");  
 banner.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 Label nameLabel = new Label("Name:");  
 TextField nameField = new TextField();  
 Label emailLabel = new Label("Email:");  
 TextField emailField = new TextField();  
 Label contactLabel = new Label("Contact:");  
 TextField contactField = new TextField();  
 Label passwordLabel = new Label("Password:");  
 PasswordField passwordField = new PasswordField();  
 Label genderLabel = new Label("Gender:");  
 ComboBox<String> genderCombo = new ComboBox<>();  
 genderCombo.getItems().addAll("Male", "Female", "Other");  
 Label bloodGroupLabel = new Label("Blood Group:");  
 ComboBox<String> bloodGroupCombo = new ComboBox<>();  
 bloodGroupCombo.getItems().addAll("A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-");  
 Label cityLabel = new Label("City:");  
 ComboBox<String> cityCombo = new ComboBox<>();  
 cityCombo.getItems().addAll("Lahore", "Multan", "Islamabad", "Quetta", "Peshawar");  
 Button saveButton = new Button("Save");  
 saveButton.setStyle("-fx-background-color: green; -fx-text-fill: white;");  
 saveButton.setOnAction(e -> handleSaveButton());  
   
 formLayout.add(nameLabel, 0, 0);  
 formLayout.add(nameField, 1, 0);  
 formLayout.add(emailLabel, 0, 1);  
 formLayout.add(emailField, 1, 1);  
 formLayout.add(contactLabel, 0, 2);  
 formLayout.add(contactField, 1, 2);  
 formLayout.add(passwordLabel, 0, 3);  
 formLayout.add(passwordField, 1, 3);  
 formLayout.add(genderLabel, 0, 4);  
 formLayout.add(genderCombo, 1, 4);  
 formLayout.add(bloodGroupLabel, 0, 5);  
 formLayout.add(bloodGroupCombo, 1, 5);  
 formLayout.add(cityLabel, 0, 6);  
 formLayout.add(cityCombo, 1, 6);  
 formLayout.add(saveButton, 1, 7);  
 centralPanel.getChildren().clear();  
 centralPanel.getChildren().addAll(banner,formLayout);  
 }  
 private void handleSaveButton() {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*, "Profile Updated Successfully!");  
 alert.showAndWait();  
 }  
  
 private void showDonationHistory() {  
 VBox historyLayout = new VBox(20);  
 historyLayout.setAlignment(Pos.*CENTER*);  
 Label banner = new Label("DONOR DASHBOARD\n WELCOME JANE");  
 banner.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 TableView<DonationRecord> table = new TableView<>();  
 table.setMaxWidth(600);  
 table.setMaxHeight(300);  
 table.setColumnResizePolicy(TableView.*CONSTRAINED\_RESIZE\_POLICY*);  
  
 TableColumn<DonationRecord, String> dateColumn = new TableColumn<>("Date");  
 dateColumn.setCellValueFactory(data -> data.getValue().dateProperty());  
 TableColumn<DonationRecord, String> centerColumn = new TableColumn<>("Donation Center");  
 centerColumn.setCellValueFactory(data -> data.getValue().centerProperty());  
 TableColumn<DonationRecord, String> bloodTypeColumn = new TableColumn<>("Blood Type");  
 bloodTypeColumn.setCellValueFactory(data -> data.getValue().bloodTypeProperty());  
 table.getColumns().addAll(dateColumn, centerColumn, bloodTypeColumn);  
 table.getItems().addAll(  
 new DonationRecord("2024-03-15", "City Hospital", "A+"),  
 new DonationRecord("2023-12-01", "Red Cross Center", "B+"),  
 new DonationRecord("2023-08-19", "Health Clinic", "O+")  
 );  
 historyLayout.getChildren().add(table);  
 centralPanel.getChildren().clear();  
 centralPanel.getChildren().addAll(banner,historyLayout);  
 }  
  
 private void showHealthTips() {  
 VBox healthTipsLayout = new VBox(15);  
 Label banner = new Label("WELCOME JANE");  
 banner.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 healthTipsLayout.setPadding(new Insets(20));  
 healthTipsLayout.setAlignment(Pos.*TOP\_LEFT*);  
 Label title = new Label("Health Tips for Donors");  
 title.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 title.setTextFill(Color.*DARKRED*);  
 Label tipsContent = new Label(  
 "- Drink plenty of water before and after donation.\n" +  
 "- Eat a healthy, iron-rich meal before donating.\n" +  
 "- Avoid strenuous activities for at least 24 hours post-donation.\n" +  
 "- Inform the staff if you feel unwell or dizzy after donating.\n" +  
 "- Regular blood donation improves heart health and reduces harmful iron levels."  
 );  
 tipsContent.setFont(Font.*font*("Arial", 16));  
 tipsContent.setWrapText(true);  
 healthTipsLayout.getChildren().addAll(title, tipsContent);  
  
 centralPanel.getChildren().clear();  
 centralPanel.getChildren().addAll(banner,healthTipsLayout);  
 }  
   
 private void showUpcomingEvents() {  
 VBox eventsLayout = new VBox(58);  
 Label banner = new Label("WELCOME JANE");  
 banner.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 eventsLayout.setAlignment(Pos.*CENTER*);  
 eventsLayout.setStyle("-fx-background-color: #fde3e3; -fx-padding: 20px;");  
 Label event1 = createStyledEventLabel("1. Blood Donation Drive - City Center\n Date: 2024-12-21 | Time: 10:00 AM - 3:00 PM\n Location: City Center Park, Multan");  
 Label event2 = createStyledEventLabel("2. Health Awareness Camp\n Date: 2024-12-23 | Time: 9:00 AM - 1:00 PM\n Location: Health Center, Multan");  
 Label event3 = createStyledEventLabel("3. Emergency Blood Drive - Urgent Need\n Date: 2024-12-25 | Time: 12:00 PM - 6:00 PM\n Location: Multan Blood Bank");  
 eventsLayout.getChildren().addAll(banner, event1, event2, event3);  
 centralPanel.getChildren().clear();  
 centralPanel.getChildren().add(eventsLayout);  
 }  
  
 private Label createStyledEventLabel(String eventDetails) {  
 Label banner = new Label("WELCOME JOHN");  
 banner.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 Label eventLabel = new Label(eventDetails);  
 eventLabel.setStyle(  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-text-fill: #333333; " +  
 "-fx-background-color: #fde3e3; " +  
 "-fx-padding: 10px; " +  
 "-fx-border-radius: 10px; " +  
 "-fx-border-color: #ca7a7a; " +  
 "-fx-border-width: 2px; "  
  
 );  
 eventLabel.setMaxWidth(600);  
 eventLabel.setWrapText(true);  
 return eventLabel;  
 }  
 private void showRequestAppointmentForm() {  
 Label banner = new Label("WELCOME JANE");  
 banner.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 GridPane formLayout = new GridPane();  
 formLayout.setPadding(new Insets(20));  
 formLayout.setVgap(10);  
 formLayout.setHgap(10);  
 Label nameLabel = new Label("Name:");  
 TextField nameField = new TextField();  
 Label emailLabel = new Label("Email:");  
 TextField emailField = new TextField();  
 Label contactLabel = new Label("Contact Number:");  
 TextField contactField = new TextField();  
 Label bloodTypeLabel = new Label("Blood Type:");  
 ComboBox<String> bloodTypeCombo = new ComboBox<>();  
 bloodTypeCombo.getItems().addAll("A+", "A-", "B+", "B-", "O+", "O-", "AB+", "AB-");  
 Label centerLabel = new Label("Donation Center:");  
 ComboBox<String> centerCombo = new ComboBox<>();  
 centerCombo.getItems().addAll("City Hospital", "Red Cross Center", "Health Clinic");  
 Label dateLabel = new Label("Preferred Date:");  
 DatePicker datePicker = new DatePicker();  
 Label timeLabel = new Label("Preferred Time:");  
 ComboBox<String> timeCombo = new ComboBox<>();  
 timeCombo.getItems().addAll("10:00 AM", "12:00 PM", "2:00 PM", "4:00 PM");  
  
 Button submitButton = new Button("Request Appointment");  
 submitButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-weight: bold;");  
 submitButton.setOnAction(e -> handleAppointmentRequest(nameField.getText(), emailField.getText(), contactField.getText(),  
 bloodTypeCombo.getValue(), centerCombo.getValue(), datePicker.getValue(), timeCombo.getValue()));  
 formLayout.add(nameLabel, 0, 0);  
 formLayout.add(nameField, 1, 0);  
 formLayout.add(emailLabel, 0, 1);  
 formLayout.add(emailField, 1, 1);  
 formLayout.add(contactLabel, 0, 2);  
 formLayout.add(contactField, 1, 2);  
 formLayout.add(bloodTypeLabel, 0, 3);  
 formLayout.add(bloodTypeCombo, 1, 3);  
 formLayout.add(centerLabel, 0, 4);  
 formLayout.add(centerCombo, 1, 4);  
 formLayout.add(dateLabel, 0, 5);  
 formLayout.add(datePicker, 1, 5);  
 formLayout.add(timeLabel, 0, 6);  
 formLayout.add(timeCombo, 1, 6);  
 formLayout.add(submitButton, 1, 7);  
 centralPanel.getChildren().clear();  
 centralPanel.getChildren().addAll(banner,formLayout);  
 }  
 private void handleAppointmentRequest(String name, String email, String contact, String bloodType, String center,  
 LocalDate date, String time) {  
 System.*out*.println("Appointment Requested: " + name + ", " + email + ", " + bloodType + ", " + center + ", "  
 + date + " " + time);  
 showConfirmationMessage();  
 }  
 private void showConfirmationMessage() {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Appointment Request");  
 alert.setHeaderText("Your appointment has been successfully requested!");  
 alert.setContentText("You will receive a confirmation email shortly.");  
 alert.showAndWait();  
 }  
   
 private void logout() {  
 Stage stage = (Stage) rootLayout.getScene().getWindow();  
 stage.close();  
 new Main().start(new Stage());  
 }  
 public static void main(String[] args) {  
 User dummyUser = new User(1, "John Doe", "Donor", "john@example.com", "password123");  
 new DonorDashboard(dummyUser).start(new Stage());  
 }  
}

**Hospital Dashboard.java**

package com.example.bloodbankjava;  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.layout.\*;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.stage.Stage;  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
  
class HospitalDashboard extends Application {  
 private User hospital;  
  
 public HospitalDashboard(User hospital) {  
 this.hospital = hospital;  
 }  
 @Override  
 public void start(Stage primaryStage) {  
 BorderPane mainLayout = new BorderPane();  
 VBox sidebar = new VBox(15);  
 sidebar.setPadding(new Insets(20));  
 sidebar.setStyle("-fx-background-color: #610b0b;");  
 Label menuTitle = new Label("Menu");  
 menuTitle.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 18));  
 menuTitle.setTextFill(Color.*WHITE*);  
 Button pendingRequestsButton = new Button("Pending Requests");  
 Button completedRequestsButton = new Button("Completed Requests");  
 Button statisticsButton = new Button("Statistics");  
 Button contactInfoButton = new Button("Contact Info");  
 Button searchRequestsButton = new Button("Search Requests");  
 Button logoutButton = new Button("Logout");  
  
 String sidebarButtonStyle = "-fx-background-color: #610b0b; -fx-text-fill: white; -fx-font-size: 14; -fx-padding: 10;";  
 pendingRequestsButton.setStyle(sidebarButtonStyle);  
 completedRequestsButton.setStyle(sidebarButtonStyle);  
 statisticsButton.setStyle(sidebarButtonStyle);  
 contactInfoButton.setStyle(sidebarButtonStyle);  
 contactInfoButton.setOnAction(e->{  
 });  
 searchRequestsButton.setStyle(sidebarButtonStyle);  
 logoutButton.setStyle(sidebarButtonStyle);  
  
 pendingRequestsButton.setMaxWidth(Double.*MAX\_VALUE*);  
 completedRequestsButton.setMaxWidth(Double.*MAX\_VALUE*);  
 statisticsButton.setMaxWidth(Double.*MAX\_VALUE*);  
 contactInfoButton.setMaxWidth(Double.*MAX\_VALUE*);  
 searchRequestsButton.setMaxWidth(Double.*MAX\_VALUE*);  
 logoutButton.setMaxWidth(Double.*MAX\_VALUE*);  
 sidebar.getChildren().addAll(menuTitle, pendingRequestsButton, completedRequestsButton, statisticsButton, contactInfoButton, searchRequestsButton, logoutButton);  
   
 HBox header = new HBox();  
 header.setPadding(new Insets(10));  
 header.setAlignment(Pos.*CENTER*);  
 header.setStyle("-fx-background-color: #610b0b;");  
 Label titleLabel = new Label("Hospital Dashboard - " + hospital.getName());  
 titleLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
  
 titleLabel.setTextFill(Color.*WHITE*);  
 header.getChildren().add(titleLabel);  
 VBox contentArea = new VBox(20);  
 contentArea.setPadding(new Insets(20));  
 contentArea.setAlignment(Pos.*CENTER*);  
 Label welcomeLabel = new Label("Welcome to the Hospital Dashboard");  
 welcomeLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 welcomeLabel.setStyle("-fx-text-fill: #610b0b;");  
 Image hospitalImage = new Image("C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\hospitalicon.png");  
 ImageView imageView = new ImageView(hospitalImage);  
 imageView.setFitWidth(300);  
 imageView.setFitHeight(300);  
 imageView.setPreserveRatio(true);  
  
 contentArea.getChildren().addAll(welcomeLabel, imageView);  
 pendingRequestsButton.setOnAction(e -> showPendingRequests(contentArea));  
 completedRequestsButton.setOnAction(e -> showCompletedRequests(contentArea));  
 contactInfoButton.setOnAction(e -> showContactInfo(contentArea));  
 statisticsButton.setOnAction(e -> {  
 StatisticsScreen statisticsScreen = new StatisticsScreen();  
 statisticsScreen.start(new Stage());  
 });  
 searchRequestsButton.setOnAction(e -> showSearchRequests(contentArea));  
 logoutButton.setOnAction(e -> {  
 primaryStage.close();  
 new Main().start(new Stage());  
 });  
 mainLayout.setLeft(sidebar);  
 mainLayout.setTop(header);  
 mainLayout.setCenter(contentArea);  
 Scene scene = new Scene(mainLayout, 1000, 600);  
 primaryStage.setTitle("Hospital Dashboard");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
 private void showPendingRequests(VBox contentArea) {  
 contentArea.getChildren().clear();  
 Label label = new Label("Pending Requests");  
 label.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
 contentArea.getChildren().add(label);  
 List<String> requests = filterRequestsByStatus("Pending");  
 for (String request : requests) {  
 Label requestLabel = new Label(request);  
 requestLabel.setFont(Font.*font*("Arial", 14));  
 contentArea.getChildren().add(requestLabel);  
 }  
 }  
 private void showCompletedRequests(VBox contentArea) {  
 contentArea.getChildren().clear();  
 Label label = new Label("Completed Requests");  
 label.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
 contentArea.getChildren().add(label);  
 List<String> requests = filterRequestsByStatus("Completed");  
 for (String request : requests) {  
 Label requestLabel = new Label(request);  
 requestLabel.setFont(Font.*font*("Arial", 14));  
 contentArea.getChildren().add(requestLabel);  
 }  
 }  
  
 private void showContactInfo(VBox contentArea) {  
 contentArea.getChildren().clear();  
 Label contactTitle = new Label("Hospital Contact Information");  
 contactTitle.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
 contentArea.getChildren().add(contactTitle);  
 Label hospitalAddress = new Label("Address: 123 Health St, City, Country");  
 Label hospitalPhone = new Label("Phone: +1234567890");  
 Label hospitalEmail = new Label("Email: contact@hospital.com");  
 hospitalAddress.setFont(Font.*font*("Arial", 14));  
 hospitalPhone.setFont(Font.*font*("Arial", 14));  
 hospitalEmail.setFont(Font.*font*("Arial", 14));  
 VBox contactInfoLayout = new VBox(10);  
 contactInfoLayout.setAlignment(Pos.*TOP\_LEFT*);  
 contactInfoLayout.getChildren().addAll(hospitalAddress, hospitalPhone, hospitalEmail);  
  
 Label feedbackTitle = new Label("We value your feedback!");  
 feedbackTitle.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 18));  
 TextArea feedbackText = new TextArea();  
 feedbackText.setPromptText("Enter your feedback here...");  
 feedbackText.setPrefHeight(100);  
 Button submitFeedbackButton = new Button("Submit Feedback");  
 submitFeedbackButton.setStyle("-fx-background-color: #610b0b; -fx-text-fill: white; -fx-font-size: 14;");  
  
 submitFeedbackButton.setOnAction(e -> {  
 String feedback = feedbackText.getText().trim();  
 if (!feedback.isEmpty()) {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Feedback Submitted");  
 alert.setHeaderText("Thank you for your feedback!");  
 alert.setContentText("Your feedback has been submitted successfully.");  
 alert.showAndWait();  
 feedbackText.clear();  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Feedback Error");  
 alert.setHeaderText("Feedback Empty");  
 alert.setContentText("Please provide feedback before submitting.");  
 alert.showAndWait();  
 }  
 });  
   
 Label signInTitle = new Label("Sign In Using Email");  
 signInTitle.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 18));  
 TextField emailField = new TextField();  
 emailField.setPromptText("Enter your email");  
 Button signInButton = new Button("Sign In");  
 signInButton.setStyle("-fx-background-color: #610b0b; -fx-text-fill: white; -fx-font-size: 14;");  
 signInButton.setOnAction(e -> {  
 String email = emailField.getText().trim();  
 if (!email.isEmpty() && email.contains("@")) {  
 // Simulate sign-in process  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Sign In Successful");  
 alert.setHeaderText("Welcome!");  
 alert.setContentText("You have successfully signed in with: " + email);  
 alert.showAndWait();  
 emailField.clear();  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Sign In Error");  
 alert.setHeaderText("Invalid Email");  
 alert.setContentText("Please enter a valid email address.");  
 alert.showAndWait();  
 }  
 });  
  
 VBox signInLayout = new VBox(10);  
 signInLayout.setAlignment(Pos.*CENTER\_LEFT*);  
 signInLayout.getChildren().addAll(signInTitle, emailField, signInButton);  
 VBox feedbackLayout = new VBox(10);  
 feedbackLayout.setAlignment(Pos.*CENTER\_LEFT*);  
 feedbackLayout.getChildren().addAll(feedbackTitle, feedbackText, submitFeedbackButton);  
 VBox finalLayout = new VBox(20);  
 finalLayout.setPadding(new Insets(20));  
 finalLayout.setAlignment(Pos.*CENTER*);  
 finalLayout.getChildren().addAll(contactInfoLayout, feedbackLayout, signInLayout);  
 contentArea.getChildren().add(finalLayout);  
 }  
  
 private void showSearchRequests(VBox contentArea) {  
 contentArea.getChildren().clear();  
 Label label = new Label("Search Requests by Blood Group");  
 label.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
 TextField bloodGroupField = new TextField();  
 bloodGroupField.setPromptText("Enter Blood Group (e.g., A+)");  
 Button searchButton = new Button("Search");  
 searchButton.setStyle("-fx-background-color: #610b0b; -fx-text-fill: white; -fx-font-size: 14;");  
 VBox searchLayout = new VBox(10);  
 searchLayout.setAlignment(Pos.*CENTER*);  
 searchLayout.getChildren().addAll(label, bloodGroupField, searchButton);  
 contentArea.getChildren().add(searchLayout);  
 searchButton.setOnAction(e -> {  
 String bloodGroup = bloodGroupField.getText().trim();  
 if (!bloodGroup.isEmpty()) {  
 List<String> filteredRequests = filterRequestsByBloodGroup(bloodGroup);  
 contentArea.getChildren().clear();  
 Label resultLabel = new Label("Search Results for Blood Group: " + bloodGroup);  
 resultLabel.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 20));  
 contentArea.getChildren().add(resultLabel);  
  
 for (String request : filteredRequests) {  
 Label requestLabel = new Label(request);  
 requestLabel.setFont(Font.*font*("Arial", 14));  
 contentArea.getChildren().add(requestLabel);  
 }  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText("Blood Group Missing");  
 alert.setContentText("Please enter a valid blood group to search.");  
 alert.showAndWait();  
 }  
 });  
 }  
 private List<String> filterRequestsByStatus(String status) {  
 List<String> filteredRequests = new ArrayList<>();  
 try (BufferedReader reader = new BufferedReader(new FileReader("C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\dons.txt"))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 if (line.contains("Status: " + status)) {  
 filteredRequests.add(line);  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return filteredRequests;  
 }  
 private List<String> filterRequestsByBloodGroup(String bloodGroup) {  
 List<String> filteredRequests = new ArrayList<>();  
 try (BufferedReader reader = new BufferedReader(new FileReader("C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\dons.txt"))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 if (line.contains("Blood Group: " + bloodGroup)) {  
 filteredRequests.add(line);  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return filteredRequests;  
 }  
}

**Donor Servicee.java**

package com.example.bloodbankjava;  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
import java.nio.file.StandardOpenOption;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.List;  
  
public class DonorService {  
 private static final String *DONOR\_FILE* = "C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\donor.txt";  
 private static final String *DONATION\_FILE* = "C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\donations.txt";  
 public static List<Donor> getAllDonors() {  
 List<String> lines = FileReaderWriter.*readFile*(*DONOR\_FILE*);  
 List<Donor> donors = new ArrayList<>();  
 for (String line : lines) {  
 String[] parts = line.split(",");  
 donors.add(new Donor(parts[0], Integer.*parseInt*(parts[1]), parts[2], parts[3], parts[4], parts[5]));  
 }  
 return donors;  
 }  
 public static void addDonor(Donor donor) {  
 FileReaderWriter.*appendToFile*(*DONOR\_FILE*, donor.toString());  
 }  
 public static void addDonation(int donorId, String bloodType) throws IOException {  
 String donationRecord = donorId + "," + bloodType + "," + LocalDate.*now*() + "\n";  
 Files.*write*(Paths.*get*(*DONATION\_FILE*), donationRecord.getBytes(), StandardOpenOption.*CREATE*, StandardOpenOption.*APPEND*);  
 }  
 public static String getDonationHistory(int donorId) throws IOException {  
 List<String> lines = Files.*readAllLines*(Paths.*get*(*DONATION\_FILE*));  
 StringBuilder history = new StringBuilder();  
 for (String line : lines) {  
 String[] parts = line.split(",");  
 if (Integer.*parseInt*(parts[0]) == donorId) {  
 history.append("Blood Type: ").append(parts[1])  
 .append(", Date: ").append(parts[2]).append("\n");  
 }  
 }  
 return history.length() > 0 ? history.toString() : "No donation history found.";  
 }  
 public static boolean canDonate(String lastDonationDate) {  
 return true;  
 }  
}

**SignUp Screen .java**

package com.example.bloodbankjava;  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class SignUpScreen extends Application {  
 private static final String *FILE\_PATH* = "C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\user.txt";  
 @Override  
 public void start(Stage primaryStage) {  
 VBox layout = new VBox();  
 layout.setAlignment(Pos.*CENTER*);  
 layout.setSpacing(15);  
 layout.setPadding(new Insets(20));  
 layout.setStyle("-fx-background-color: #000000;");  
 Label titleLabel = new Label("Sign-Up");  
 titleLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold; -fx-text-fill: #ffffff");  
   
 TextField nameField = new TextField();  
 nameField.setPromptText("Enter Name");  
 nameField.setPrefWidth(300);  
 ComboBox<String> genderBox = new ComboBox<>();  
 genderBox.getItems().addAll("Male", "Female", "Other");  
 genderBox.setPromptText("Select Gender");  
 genderBox.setPrefWidth(300);  
   
 ComboBox<String> bloodGroupBox = new ComboBox<>();  
 bloodGroupBox.getItems().addAll("A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-");  
 bloodGroupBox.setPromptText("Select Blood Group");  
 bloodGroupBox.setPrefWidth(300);  
 ComboBox<String> cityBox = new ComboBox<>();  
 cityBox.getItems().addAll("New York", "Los Angeles", "Chicago", "Houston", "Phoenix"); // Add more cities if needed  
 cityBox.setPromptText("Select City");  
 cityBox.setPrefWidth(300);  
 TextField contactField = new TextField();  
 contactField.setPromptText("Enter Contact Number");  
 contactField.setPrefWidth(300);  
   
 TextField emailField = new TextField();  
 emailField.setPromptText("Enter Email");  
 emailField.setPrefWidth(300);  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Enter Password");  
 passwordField.setPrefWidth(300);  
 Button signUpButton = new Button("Sign Up");  
 signUpButton.setStyle("-fx-background-color: #0d47a1; -fx-text-fill: white; -fx-font-size: 16px;");  
 signUpButton.setPrefWidth(300);  
  
 signUpButton.setOnAction(e -> {  
 String name = nameField.getText();  
 String gender = genderBox.getValue();  
 String bloodGroup = bloodGroupBox.getValue();  
 String city = cityBox.getValue();  
 String contact = contactField.getText();  
 String email = emailField.getText();  
 String role = "Donor";  
 String password = passwordField.getText();  
 if (name.isEmpty() || gender == null || bloodGroup == null || city == null || contact.isEmpty()  
 || email.isEmpty() || role.isEmpty() || password.isEmpty()) {  
 showAlert("Error", "All fields are required!", Alert.AlertType.*ERROR*);  
 return;  
 }  
 int newUserId = generateUserId();  
 User newUser = new User(newUserId, name, gender, bloodGroup, city, contact, role, email, password);  
 try {  
 writeUserToFile(newUser);  
 showAlert("Success", "Sign-Up Successful! You can now log in.", Alert.AlertType.*INFORMATION*);  
 primaryStage.close();  
 } catch (IOException ex) {  
 showAlert("Error", "Could not save user data. Please try again.", Alert.AlertType.*ERROR*);  
 ex.printStackTrace();  
 }  
 });  
 layout.getChildren().addAll(  
 titleLabel, nameField, genderBox, bloodGroupBox, cityBox, contactField,  
 emailField, passwordField, signUpButton  
 );  
 Scene scene = new Scene(layout, 500, 550);  
 primaryStage.setTitle("Sign-Up");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
 private int generateUserId() {  
 return (int) (Math.*random*() \* 100);  
 }  
 private void writeUserToFile(User user) throws IOException {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(*FILE\_PATH*, true))) {  
 String userData = user.getUserId() + "," + user.getName() + ","  
 + user.getRole() + "," + user.getEmail() + "," + user.getPassword();  
 writer.write(userData);  
 writer.newLine();  
 }  
 }  
 private void showAlert(String title, String message, Alert.AlertType type) {  
 Alert alert = new Alert(type);  
 alert.setTitle(title);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

**User.java**

package com.example.bloodbankjava;  
public class User {  
 private int userId;  
 private String name;  
 private String role;  
 private String email;  
 private String password;  
 private String bloodGroup;  
 private String city;  
 private String contact;  
 private String gender;  
  
 public User(String name, String email) {  
 this.name = name;  
 this.email = email;  
 }  
 public User(int userId, String name, String role, String email, String password) {  
 this.userId = userId;  
 this.name = name;  
 this.role = role;  
 this.email = email;  
 this.password = password;  
 }  
 public User(int newUserId, String name, String gender, String bloodGroup, String city, String contact, String role, String email, String password){  
 this.userId = newUserId;  
 this.name = name;  
 this.role = role;  
 this.email = email;  
 this.password = password;  
 }  
 public int getUserId() {  
 return userId;  
 }  
 public void setUserId(int userId) {  
 this.userId = userId;  
 }  
 public String getName() {  
 return this.name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public String getRole() {  
 return role;  
 }  
 public void setRole(String role) {  
 this.role = role;  
 }  
 public String getEmail() {  
 return email;  
 }  
 public void setEmail(String email) {  
 this.email = email;  
 }  
 public String getPassword() {  
 return password;  
 }  
 public void setPassword(String password) {  
 this.password = password;  
 }  
 public String toString() {  
 return name + " (" + email + ")";  
 }  
 public String getBloodGroup() {  
 return bloodGroup;  
 }  
 public void setBloodGroup( String bloodGroup) {  
 this.bloodGroup = bloodGroup;  
 }  
 public String getCity() {  
 return city;  
 }  
 public void setCity( String city) {  
 this.city = city;  
 }  
 public String getContact() {  
 return contact;  
 }  
 public void setContact(String contact) {  
 this.contact = contact;  
 }  
 public String getGender() {  
 return gender;  
 }  
 public void setGender(String gender) {  
 this.gender = gender;  
 }  
}

**User Service.java**

package com.example.bloodbankjava;  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
public class Userservice {  
   
 private static final String *USER\_FILE* = "C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\user.txt";  
 public static List<User> getAllUsers() {  
 List<User> users = new ArrayList<>();  
 try (BufferedReader reader = new BufferedReader(new FileReader(*USER\_FILE*))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] data = line.split(",");  
 if (data.length == 5) {  
 int id = Integer.*parseInt*(data[0]);  
 String name = data[1];  
 String role = data[2];  
 String email = data[3];  
 String password = data[4];  
 users.add(new User(id, name, role, email, password));  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return users;  
 }  
  
 public User login(String email, String password) {  
 return null; }  
 public static void addUser(User user) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(*USER\_FILE*, true))) {  
 writer.write(user.getUserId() + "," + user.getName() + "," + user.getRole() + "," + user.getEmail() + "," + user.getPassword());  
 writer.newLine();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 public static void updateUser(User user) {  
 List<User> users = *getAllUsers*();  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(*USER\_FILE*))) {  
 for (User u : users) {  
 if (u.getUserId() == user.getUserId()) {  
 writer.write(user.getUserId() + "," + user.getName() + "," + user.getRole() + "," + user.getEmail() + "," + user.getPassword());  
 } else {  
 writer.write(u.getUserId() + "," + u.getName() + "," + u.getRole() + "," + u.getEmail() + "," + u.getPassword());  
 }  
 writer.newLine();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 public static void removeUser(User user) {  
 List<User> users = *getAllUsers*();  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(*USER\_FILE*))) {  
 for (User u : users) {  
 if (u.getUserId() != user.getUserId()) {  
 writer.write(u.getUserId() + "," + u.getName() + "," + u.getRole() + "," + u.getEmail() + "," + u.getPassword());  
 writer.newLine();  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

**Main.java**

package com.example.bloodbankjava;  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.stage.Stage;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.VBox;  
import java.util.List;  
  
public class Main extends Application {  
 private static final String *FILE\_PATH* = "C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\user.txt";  
 @Override  
 public void start(Stage primaryStage) {  
 HBox mainLayout = new HBox();  
 mainLayout.setAlignment(Pos.*CENTER\_LEFT*);  
 mainLayout.setStyle("-fx-background-color: #000000;");  
 VBox formLayout = new VBox();  
 formLayout.setAlignment(Pos.*CENTER\_LEFT*);  
 formLayout.setSpacing(20);  
  
 Label title = new Label("Blood Donation System");  
 title.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 32));  
 title.setTextFill(Color.*web*("#FFFFFF"));  
 TextField usernameField = new TextField();  
 usernameField.setPromptText("Enter Username");  
 usernameField.setFont(Font.*font*(16));  
 usernameField.setPrefWidth(300);  
  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Enter Password");  
 passwordField.setFont(Font.*font*(16));  
 passwordField.setPrefWidth(300);  
 HBox buttonLayout = new HBox();  
 buttonLayout.setAlignment(Pos.*CENTER\_LEFT*);  
 buttonLayout.setSpacing(20);  
  
 Button loginButton = new Button("Login");  
 loginButton.setStyle("-fx-background-color: #0d47a1; -fx-text-fill: white; -fx-font-size: 18; -fx-padding: 10;");  
 loginButton.setPrefWidth(200);  
 Button signUpButton = new Button("Sign Up");  
 signUpButton.setStyle("-fx-background-color: #4caf50; -fx-text-fill: white; -fx-font-size: 18; -fx-padding: 10;");  
 signUpButton.setPrefWidth(200);  
 buttonLayout.getChildren().addAll(loginButton, signUpButton);  
  
 formLayout.setPadding(new Insets(50));  
 formLayout.setSpacing(30);  
 formLayout.getChildren().addAll(title, usernameField, passwordField, buttonLayout);  
 Image image = new Image("C:\\Users\\amina\\Documents\\OOPS!!!\\PRACTICE\\bloodbankjava\\src\\main\\java\\com\\example\\bloodbankjava\\WhatsApp Image 2024-12-12 at 20.41.14\_dca6454d.jpg");  
 ImageView imageView = new ImageView(image);  
 imageView.setFitWidth(400);  
 imageView.setFitHeight(600);  
 imageView.setPreserveRatio(true);  
 mainLayout.getChildren().addAll(formLayout, imageView);  
  
 signUpButton.setOnAction(e -> {  
 SignUpScreen signUpScreen = new SignUpScreen();  
 signUpScreen.start(new Stage());  
 });  
  
 loginButton.setOnAction(e -> {  
 String username = usernameField.getText();  
 String password = passwordField.getText();  
  
 User loggedInUser = getUserByUsername(username);  
 if (loggedInUser != null && loggedInUser.getPassword().equals(password)) {  
 if ("Hospital".equalsIgnoreCase(loggedInUser.getRole())) {  
 new HospitalDashboard(loggedInUser).start(new Stage());  
 } else if ("Donor".equalsIgnoreCase(loggedInUser.getRole())) {  
 new DonorDashboard(loggedInUser).start(new Stage());  
 } else if ("Admin".equalsIgnoreCase(loggedInUser.getRole())) {  
 new AdminDashboard(loggedInUser).start(new Stage());  
 } else {  
 showAlert("Error", "Unknown role. Please check your credentials.", Alert.AlertType.*ERROR*);  
 }  
 primaryStage.close();  
 } else {  
 showAlert("Error", "Invalid username or password.", Alert.AlertType.*ERROR*);  
 }  
 });  
 Scene scene = new Scene(mainLayout, 800, 500);  
 primaryStage.setTitle("Login");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
 private User getUserByUsername(String username) {  
 try {  
 List<String> lines = FileReaderWriter.*readFile*(*FILE\_PATH*);  
 for (String line : lines) {  
 String[] parts = line.split(",");  
 if (parts.length == 5) {  
 int id = Integer.*parseInt*(parts[0].trim());  
 String name = parts[1].trim();  
 String role = parts[2].trim();  
 String email = parts[3].trim();  
 String password = parts[4].trim();  
 if (username.equals(email)) {  
 return new User(id, name, role, email, password);  
 }  
 }  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
 private void showAlert(String title, String message, Alert.AlertType type) {  
 Alert alert = new Alert(type);  
 alert.setTitle(title);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

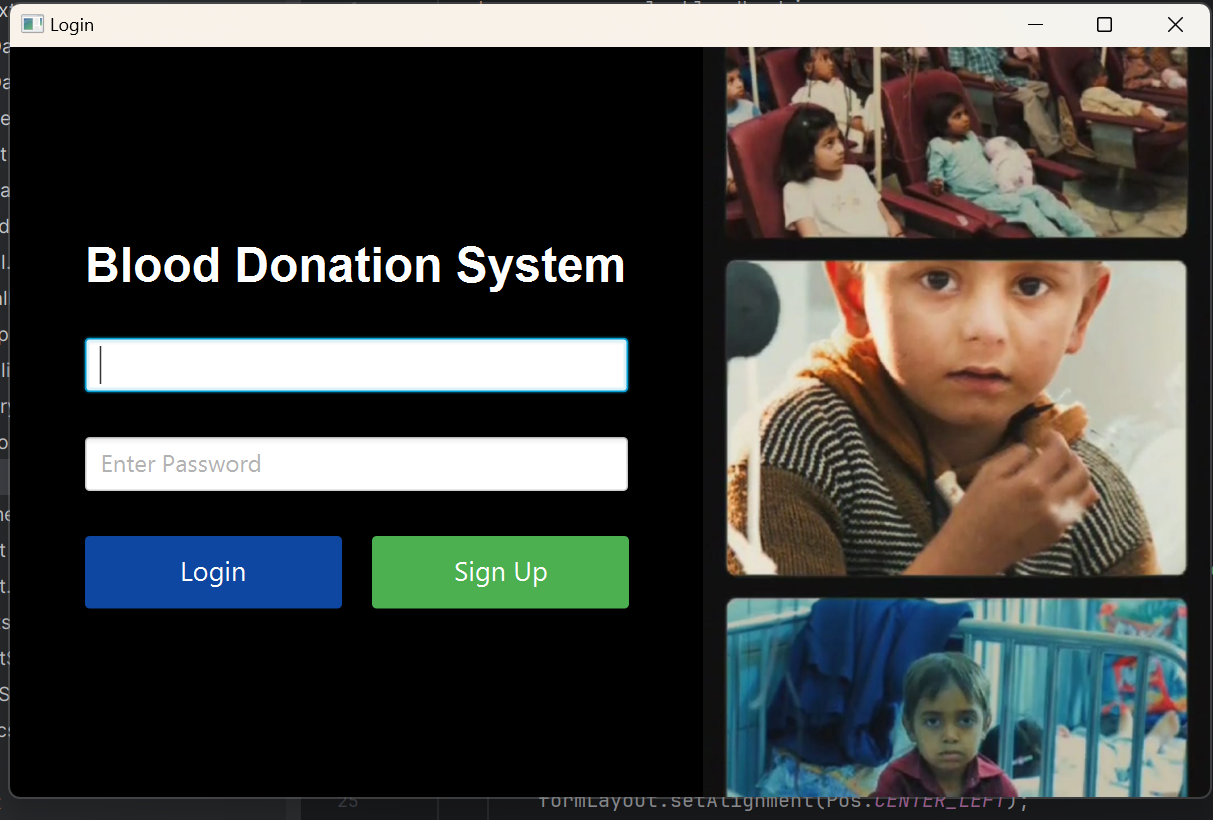
**Other Classes:**

The other classes included in the code are:

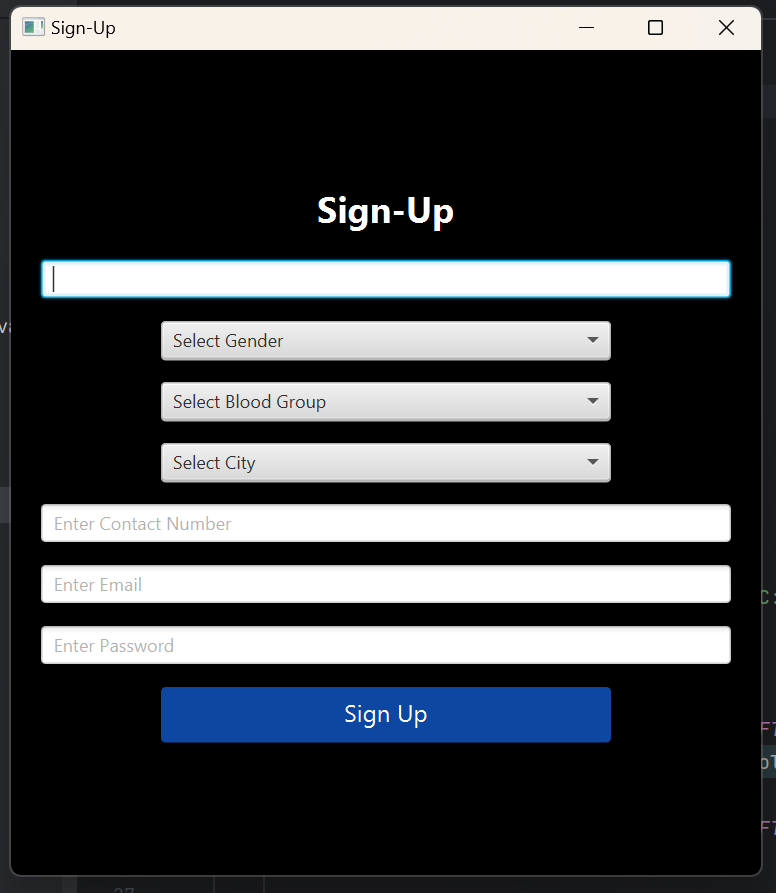
* Admin Dashboard Controller
* Blood Inventory
* Blood Inventory Service
* Blood Request
* Donation
* Donation Record
* Donation Service
* Donor
* Donor Dashboard Controller
* File Reader Writer
* Login Controller
* Request
* Request Service
* Statistics Screen

**OUTPUT SCREENS:**

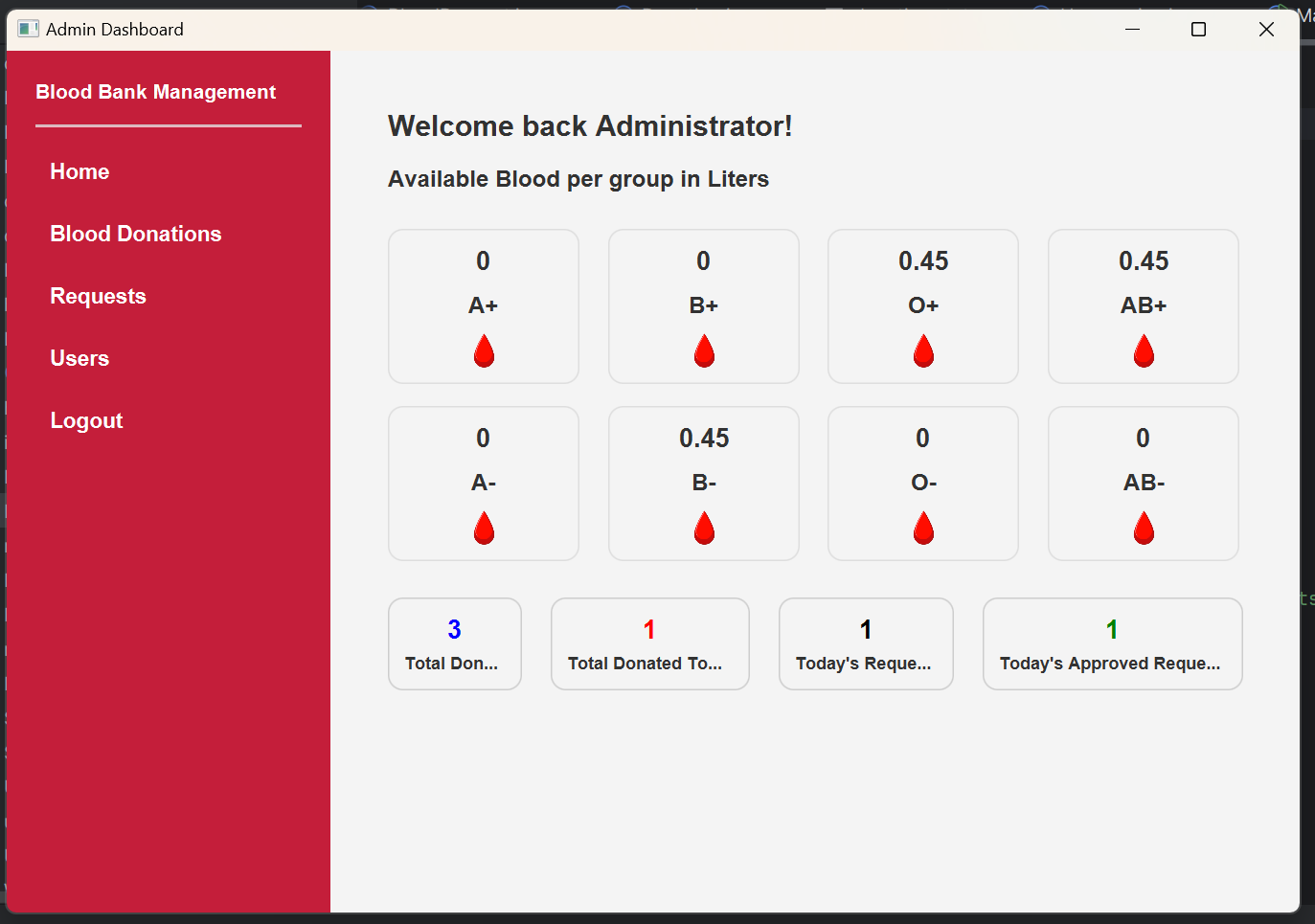
**Main Screen:**

****

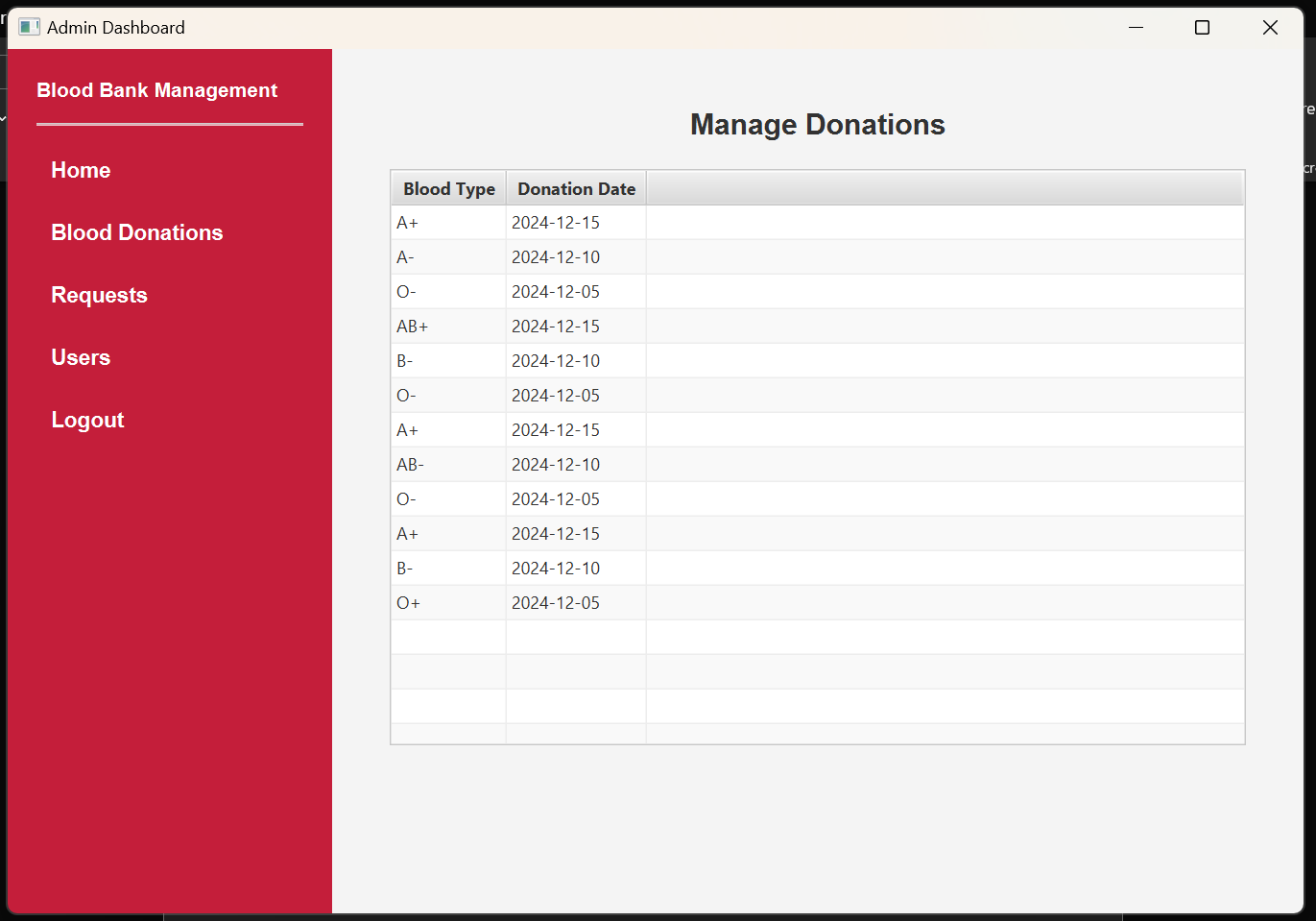
SignUp Screen:

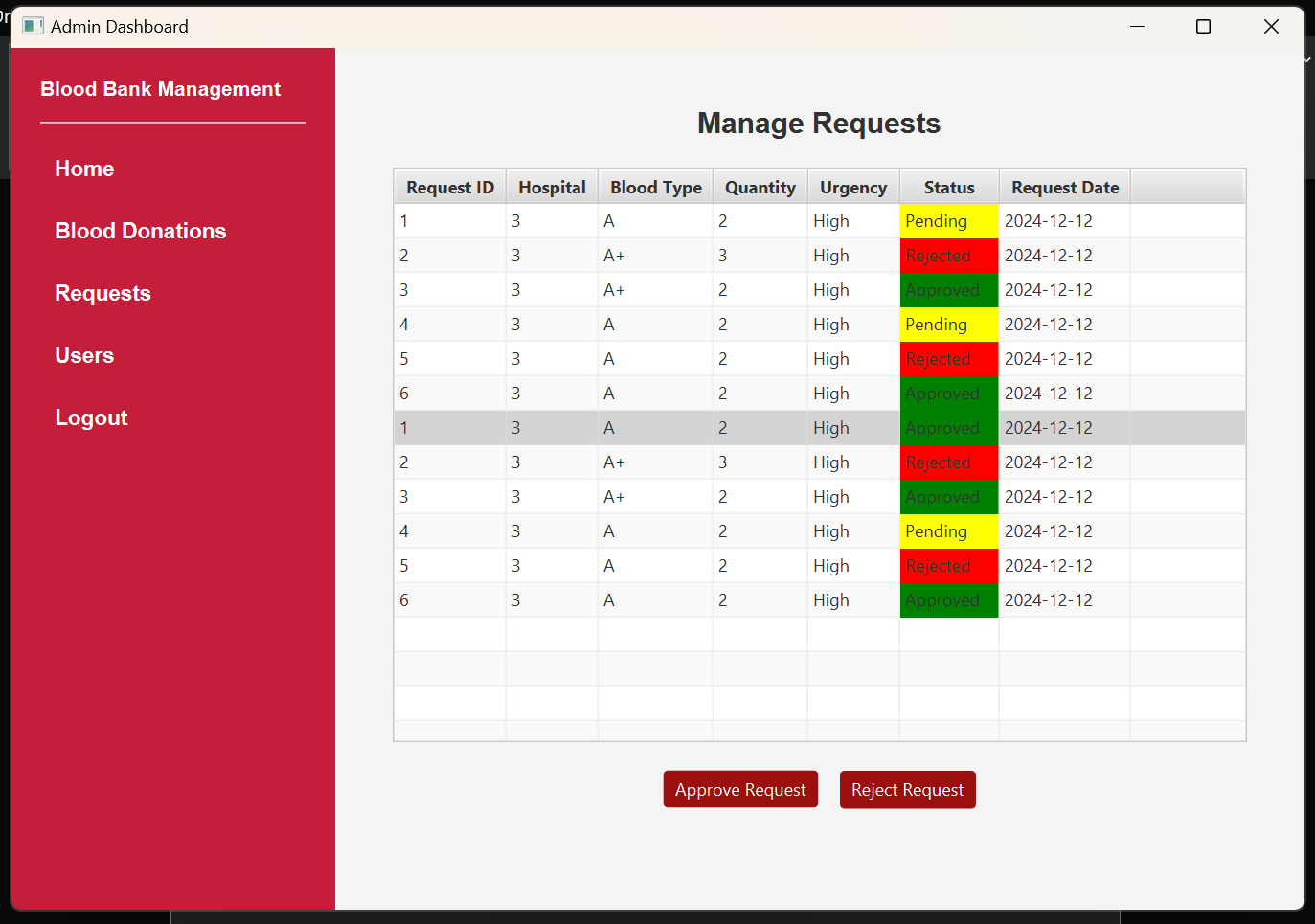


Admin Dashboard:

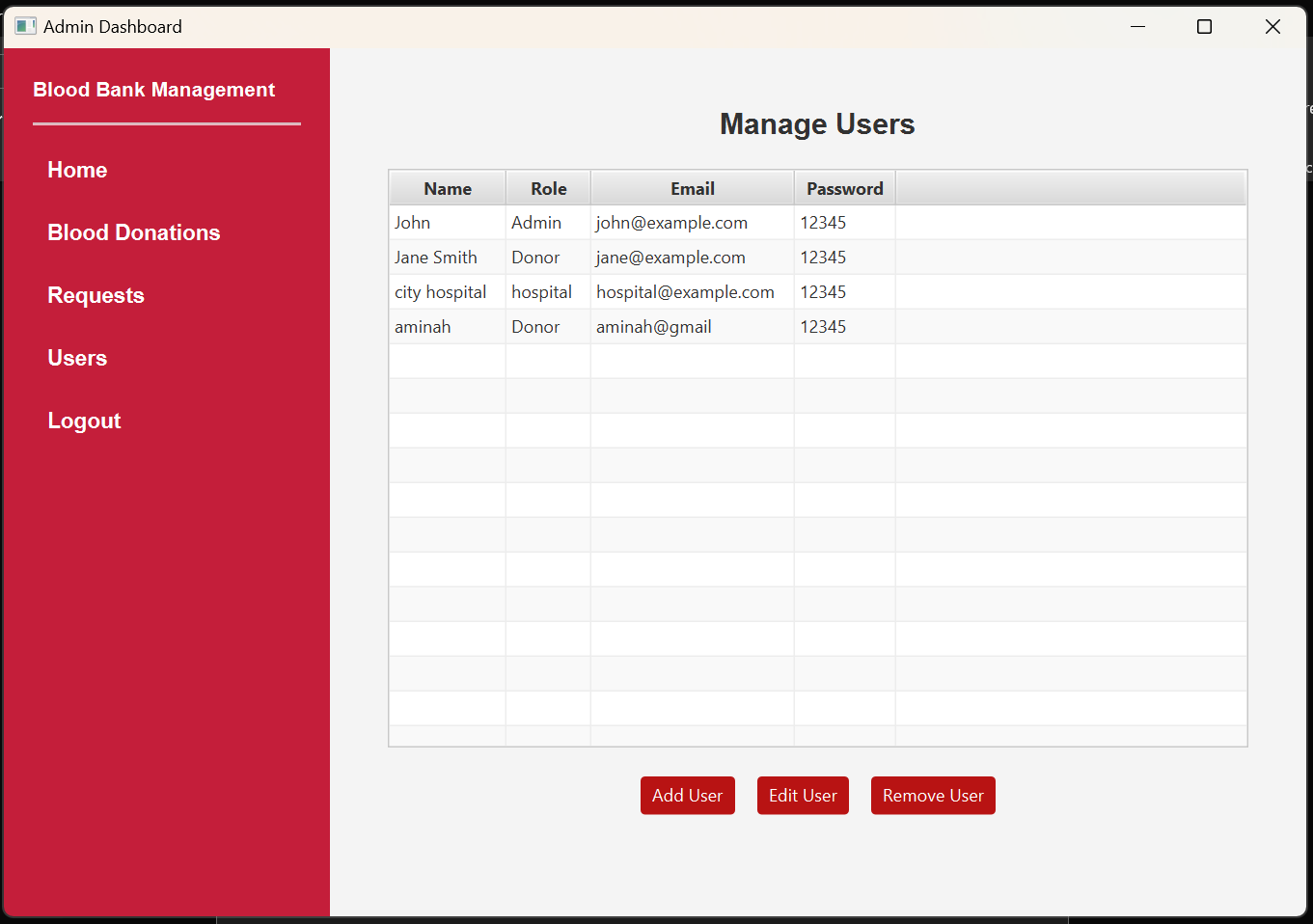
Home Page:

:

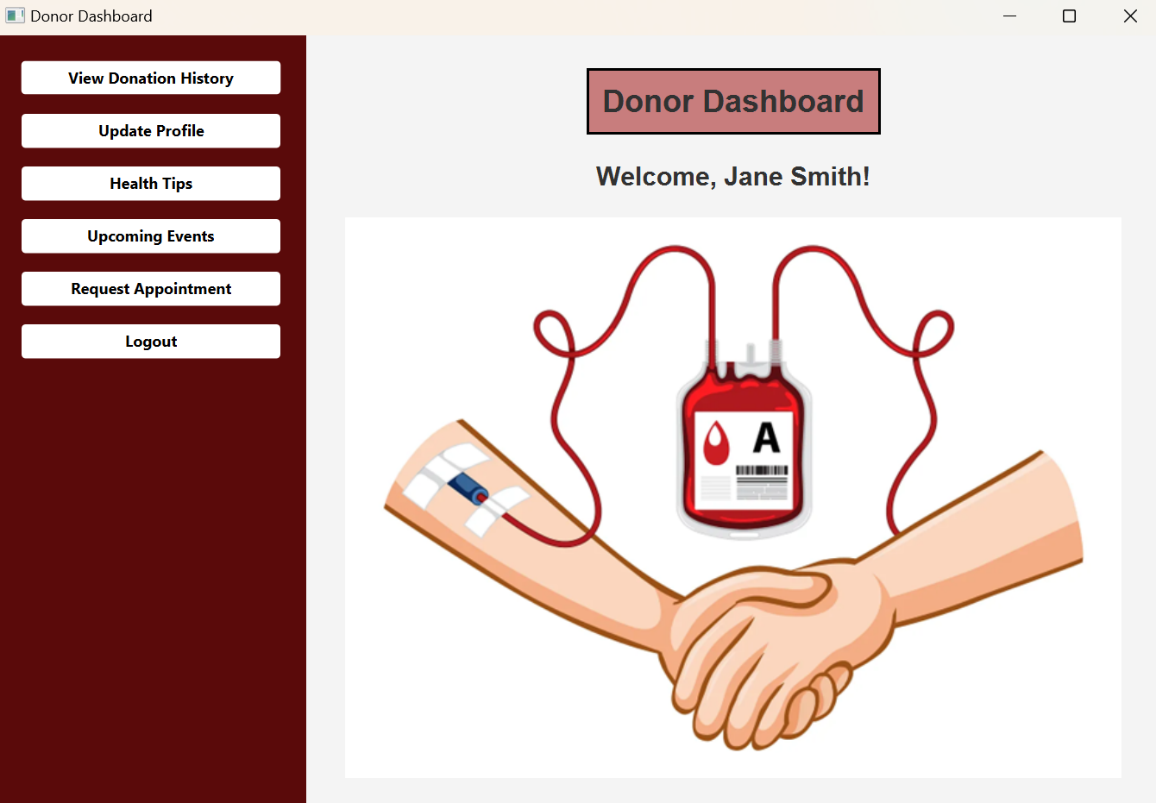
Blood Donations:

Requests:

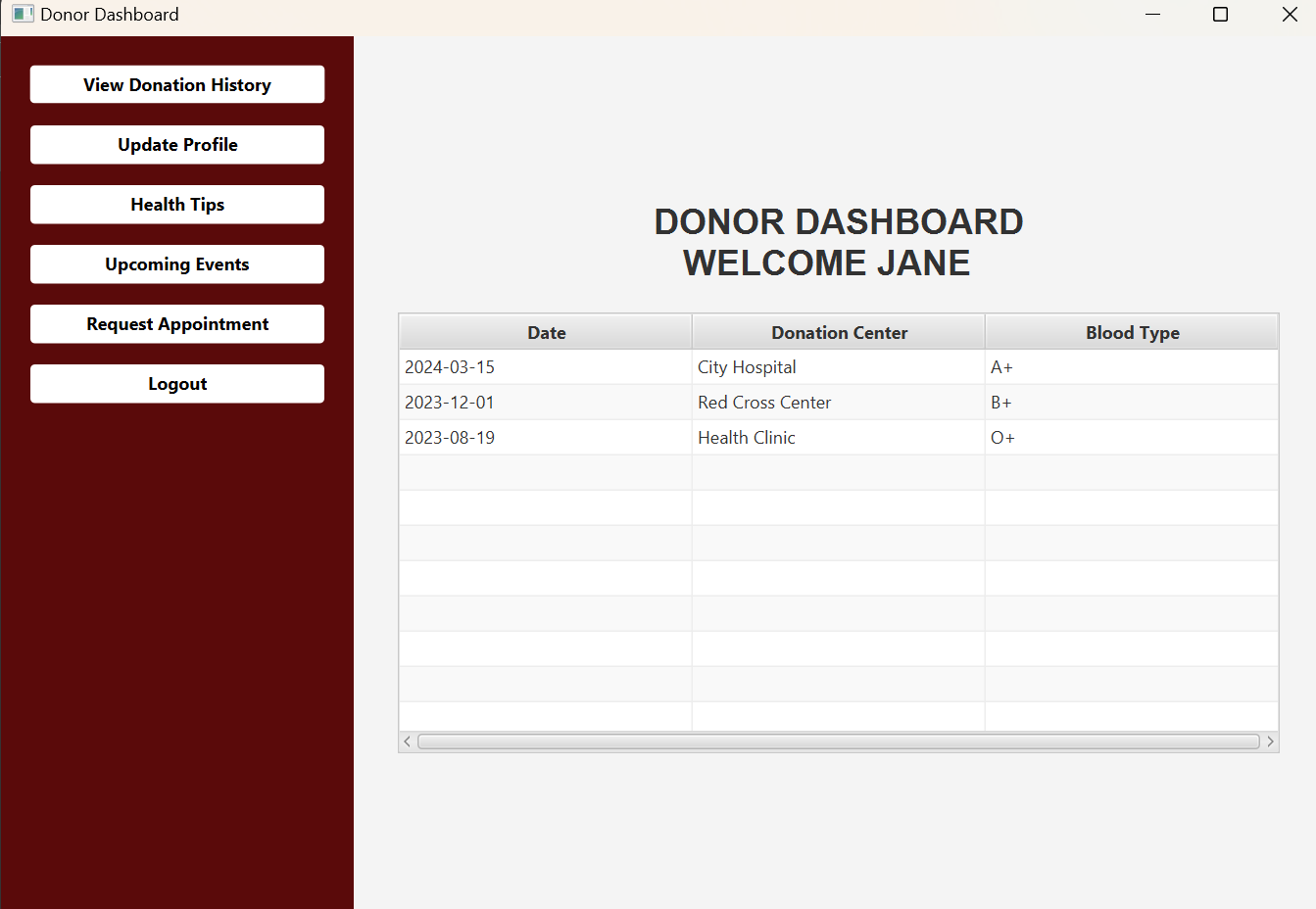
Users:



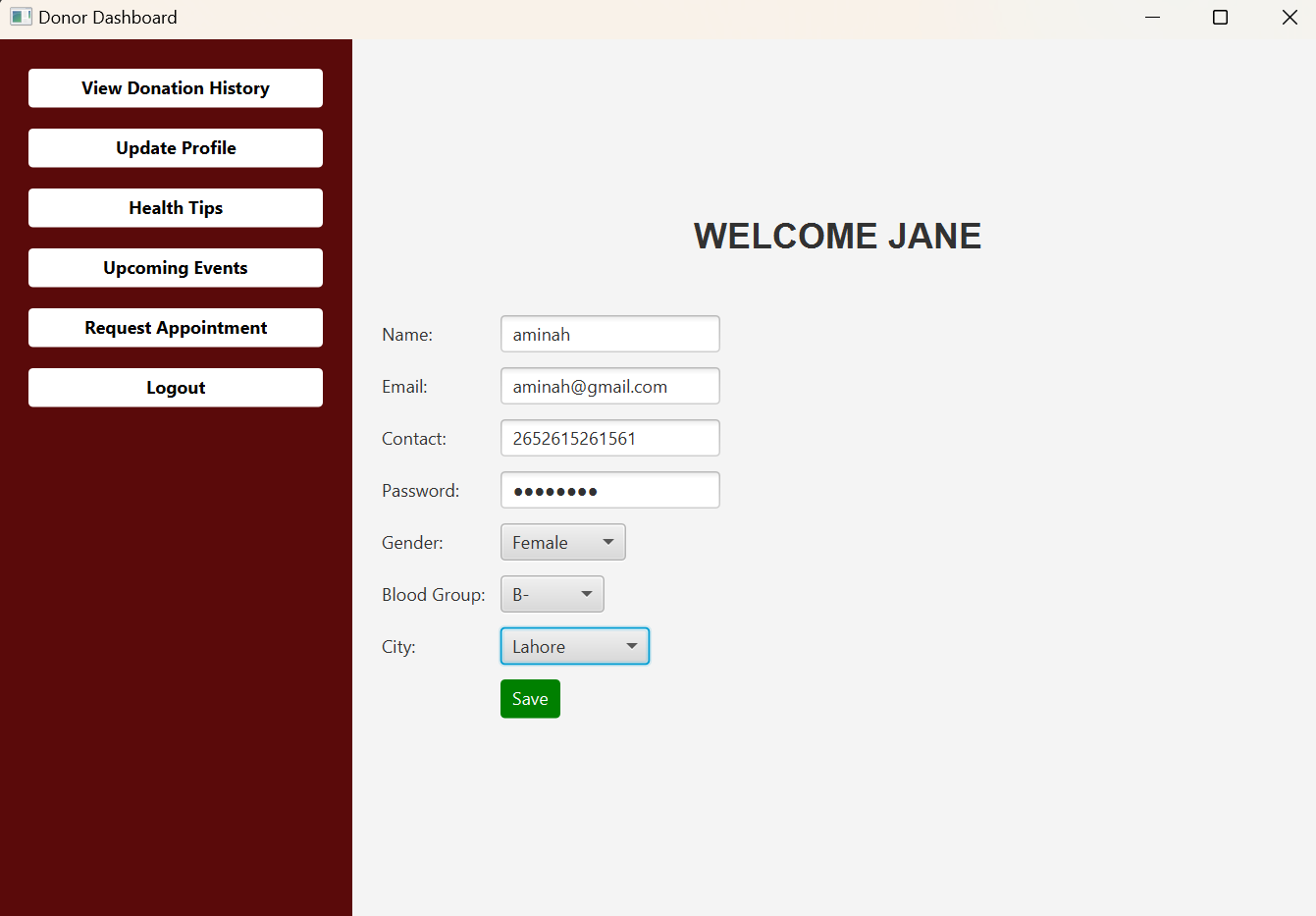
Donor Dashboard:



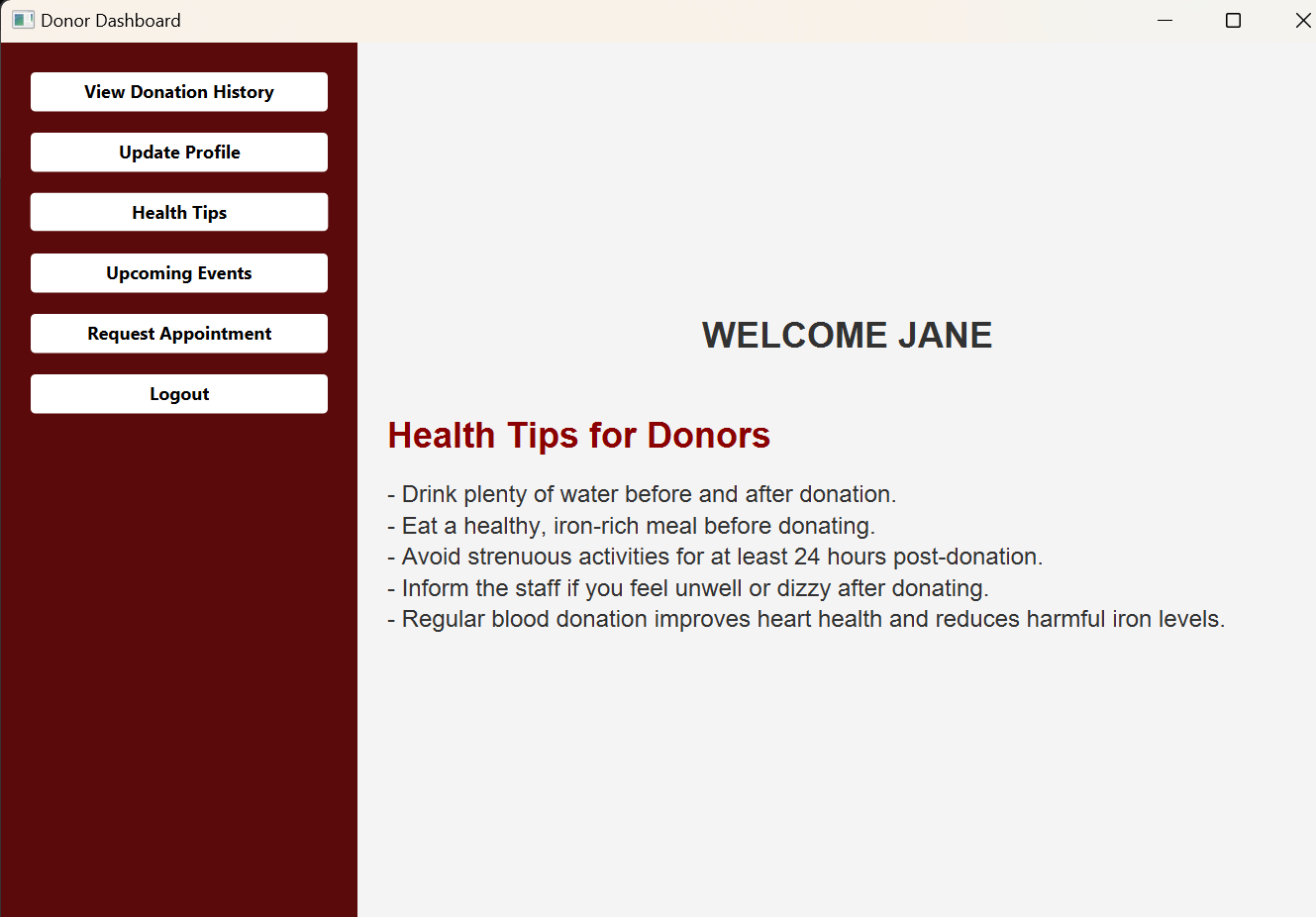
View Donation History:



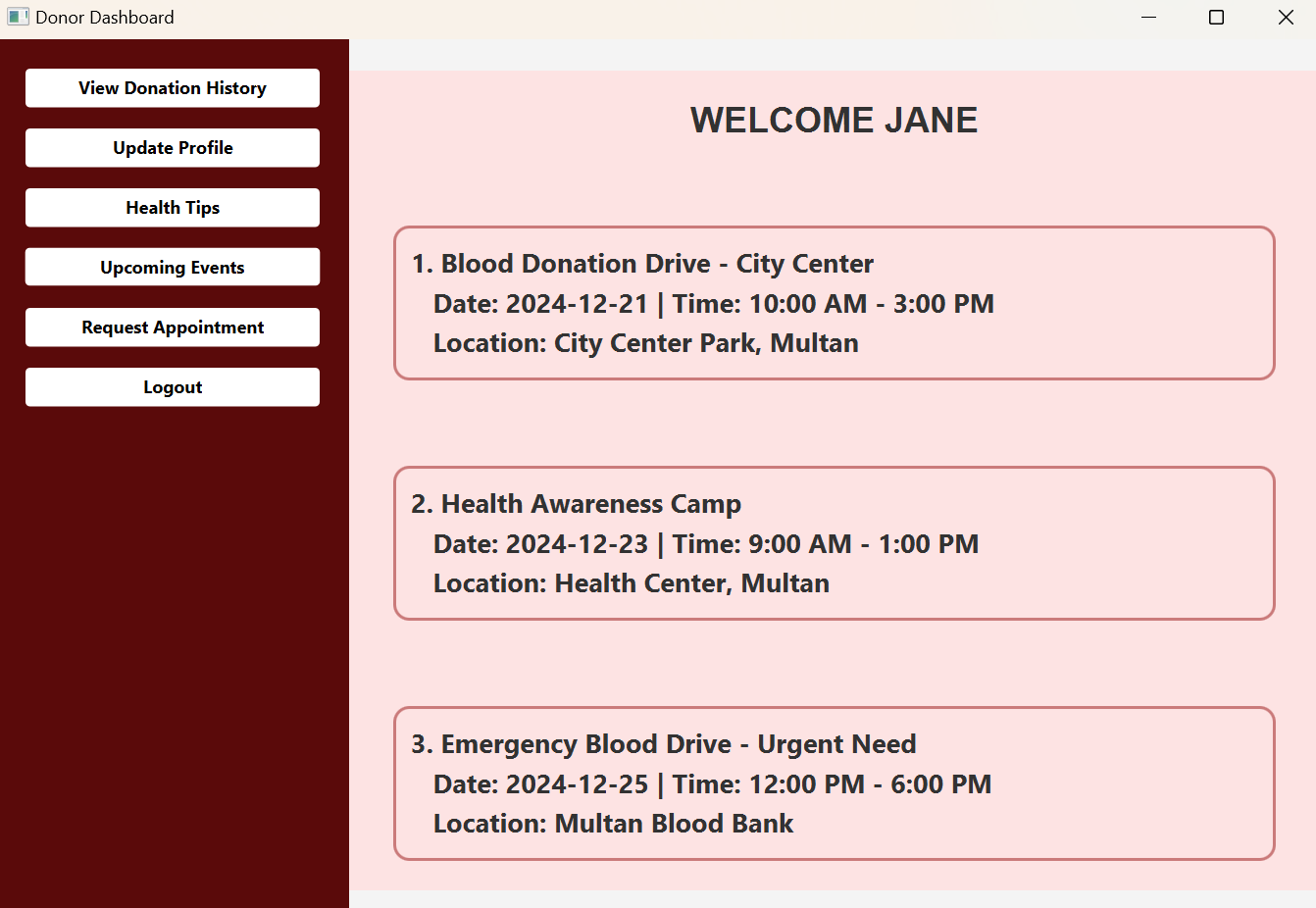
Update Profile:



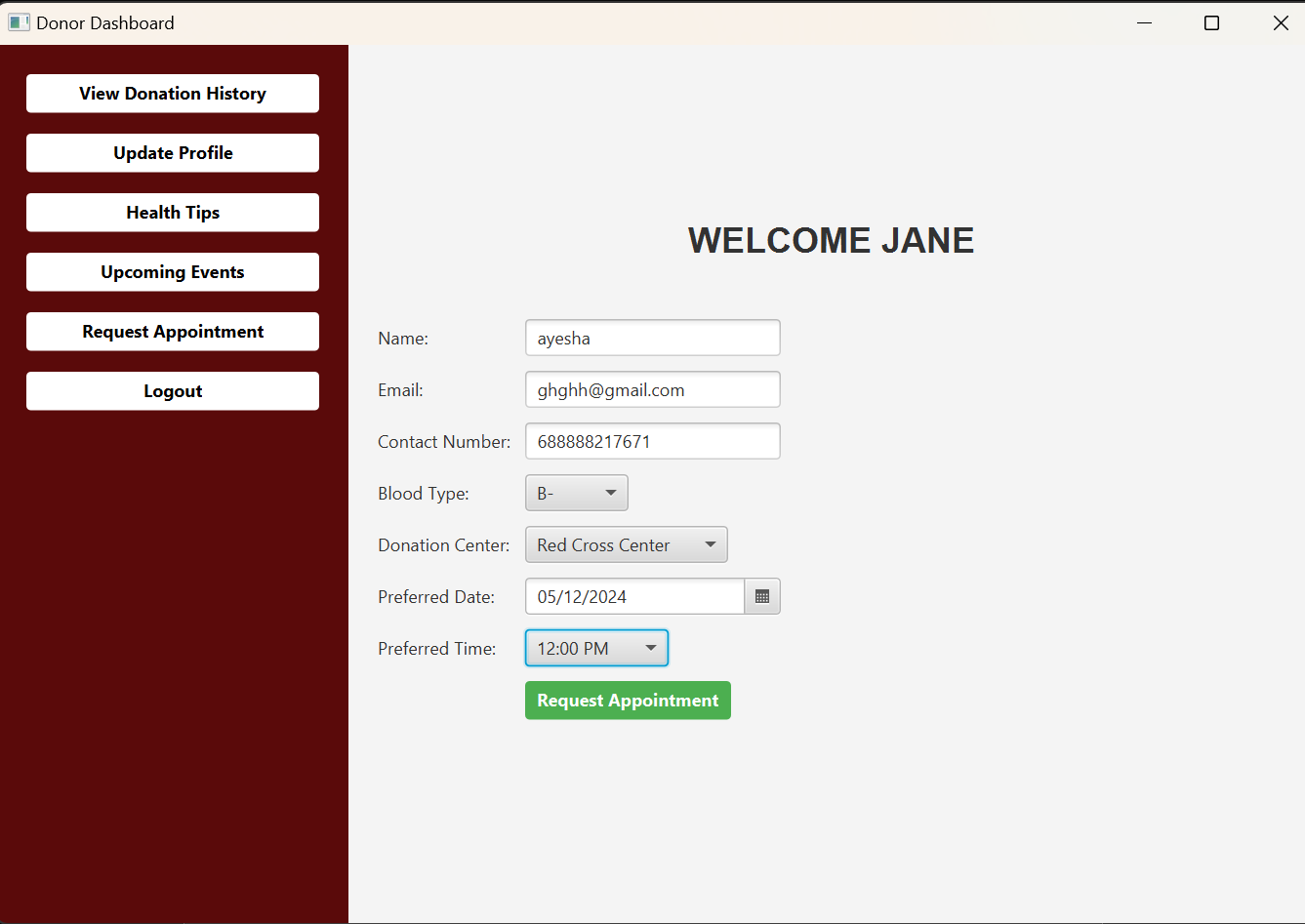
Health Tips:



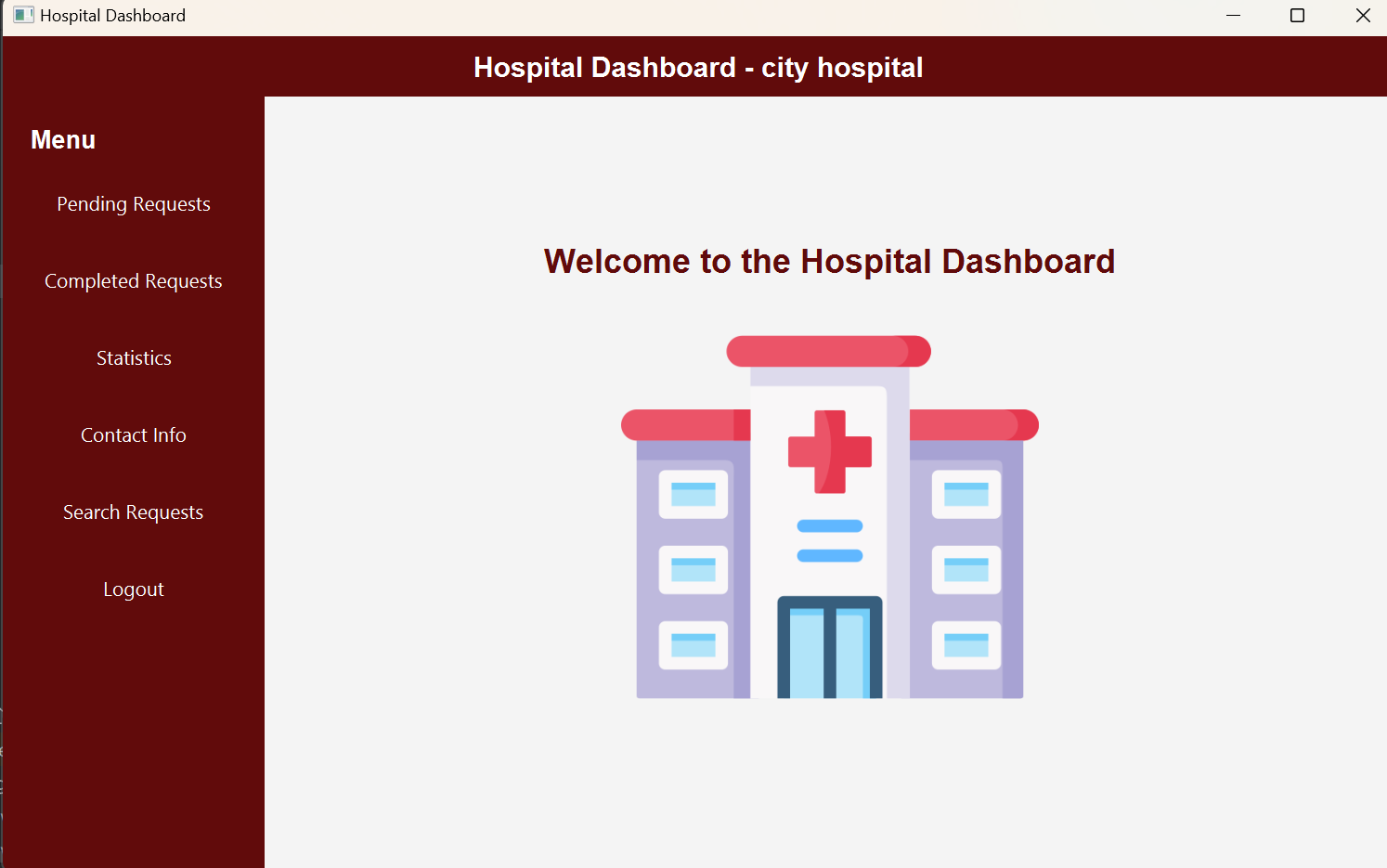
Upcoming Events:



Request Appointment:



Hospital Dashboard:

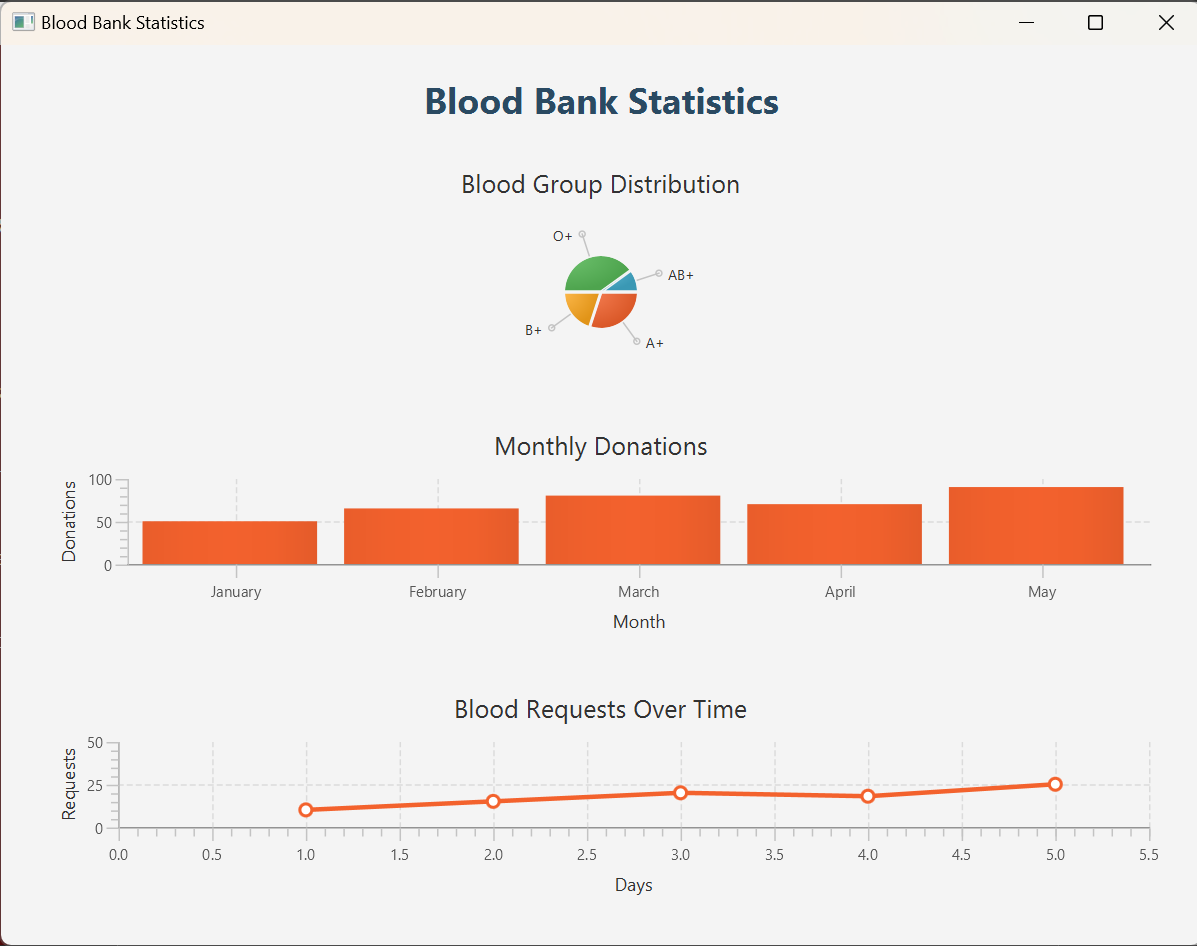


Pending requests:

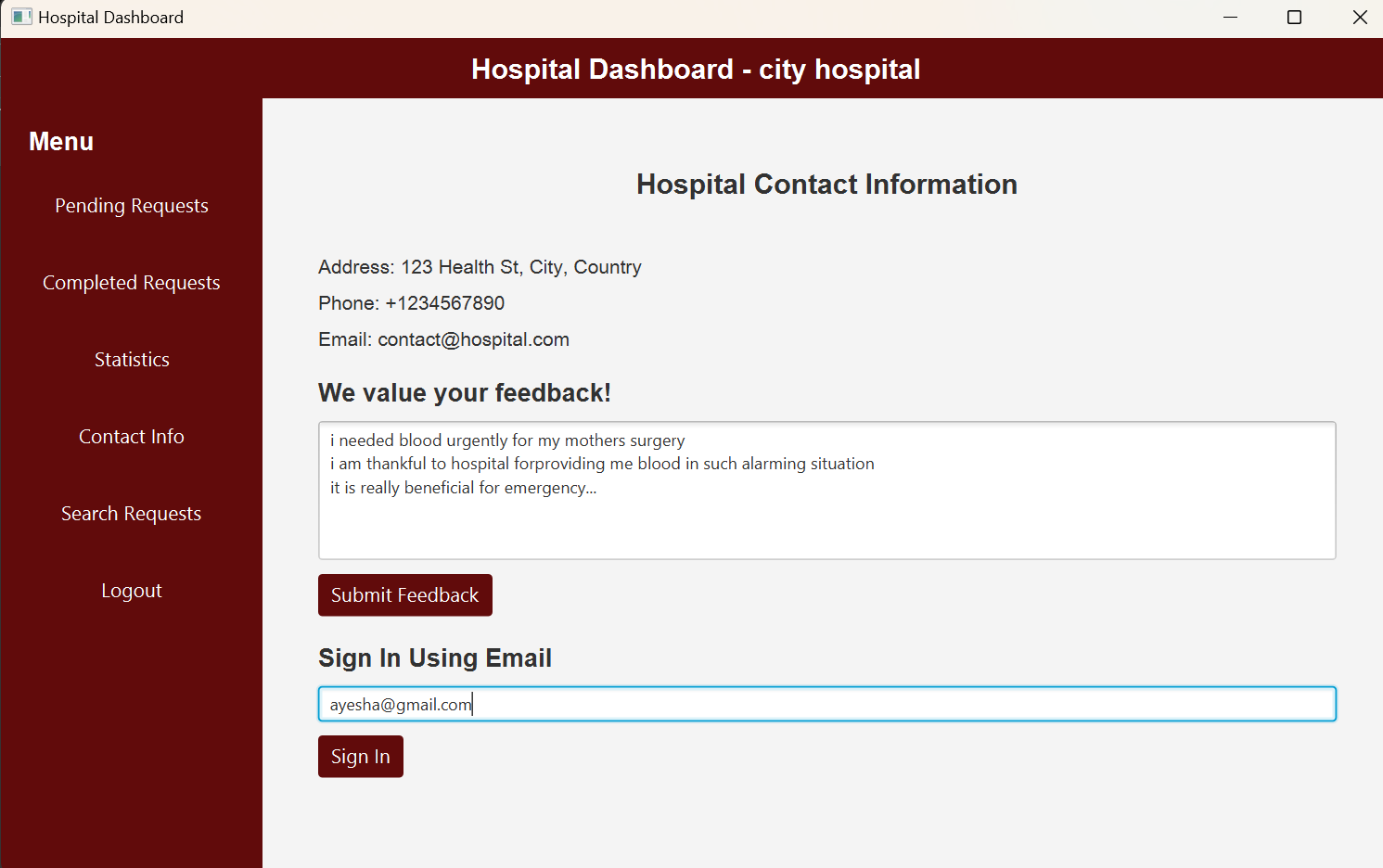


Completed requests:



Statistics: 

Contact Info:



Search requests:

