Vaccination Management System

For North South University

Semester: Spring 2023

Course Title: CSE215L

Section: 06

Date: 14th June 2023

Group Number: 03

Group Members:

- Ibrahim Bakhtiyar (ID: 2012898642)
- Kazi Asafin Islam (ID: 2112500642)
- Md. Masud Rana Hridoy (ID: 2211399042)
- Md. Istiaque Ali (ID: 2212019642)
- Umar Hasan (ID: 2222739642)

FEATURES

Our vaccination management system application is primarily designed to facilitate managing and tracking COVID-19 vaccination processes. It has the following features:

1. **Admin Login:** The application includes an admin login functionality that allows authorized administrators to access the system.

2. **User Authentication:** The system authenticates the admin users based on a predefined username and password.

3. **Appointment List:** The application displays a list of appointments in a tabular format. The appointment list contains information such as name, age, NSUer status, NID number, phone number, first-time status, password, center name, appointment date, and vaccination status.

4. **Appointment Data Storage:** The application reads appointment data from a text file ("registration.txt") and populates the appointment list accordingly.

5. **Table View:** The appointment list is presented in a table view, allowing users to browse and view appointment details easily.

6. **Appointment Completion:** Admin users can mark an appointment as completed. Selecting a row in the appointment list allows the admin to update the vaccination status from "No" to "Yes". The system also updates the appointment data in the text file accordingly.

7. **Certificate Generation:** Users can generate a vaccination certificate after getting vaccinated.

8. **Visual Styling:** The application uses colors and styling to enhance the visual appeal of the user interface. It includes a customized title icon and applies colors like navy blue, NSU gold, and NSU green to different UI components.

9. **Navigation:** The application provides navigation options to return to the previous page or return to the landing page.

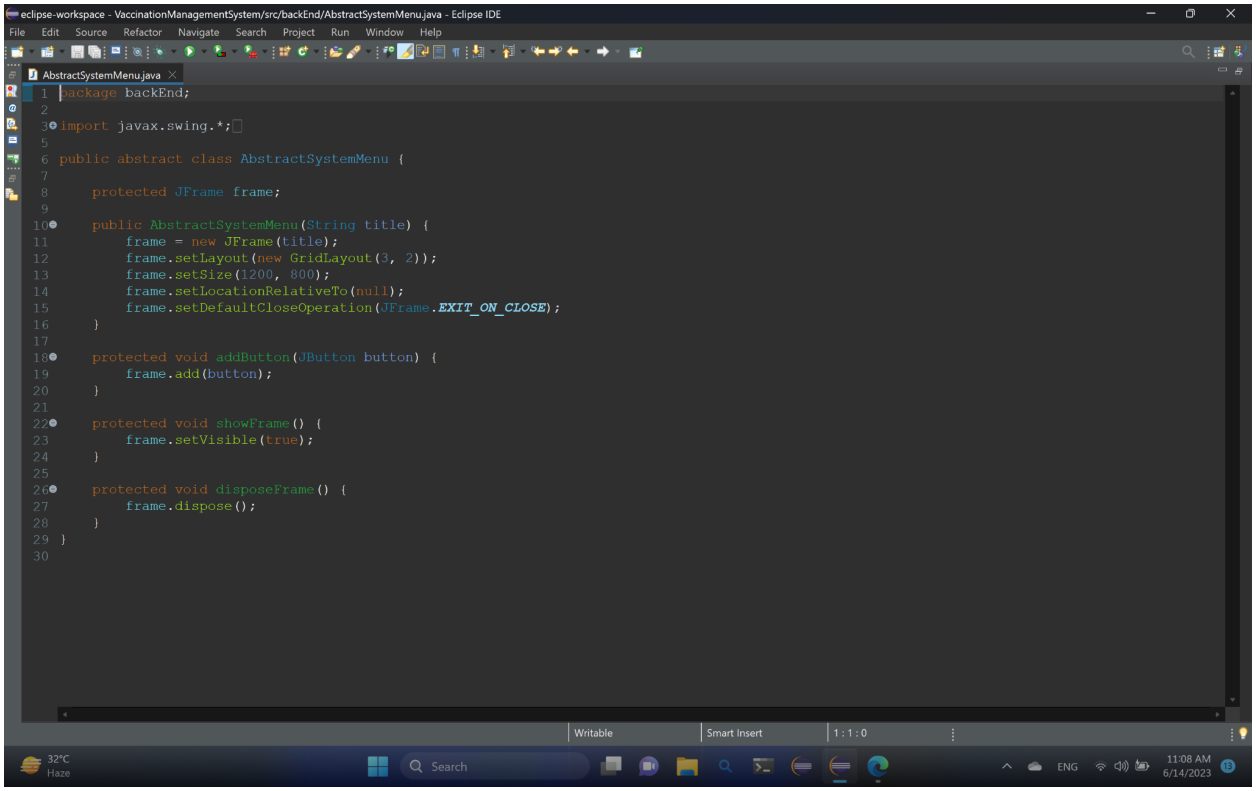
10. **Event Handling:** The application utilizes event handling mechanisms to respond to user interactions, such as mouse clicks and button presses.

11. **User Feedback:** The system provides feedback to the users through message dialogs, informing them about login status, successful appointments, and invalid credentials.

These features enable admin users to log in, view appointment details, update the vaccination status, and navigate the application.

DESCRIPTION OF EACH CLASS

The AbstractSystemMenu Class:



The class `AbstractSystemMenu` is an abstract class defined in the `backEnd` package. It provides a framework for creating system menus using the Swing library in Java. Let's go through each method in the class:

1. Constructor `AbstractSystemMenu(String title)`: The class's constructor takes a `title` parameter representing the menu title. It initializes a `JFrame` object named `frame`, sets the layout of the frame to a `GridLayout` with 3 rows and 2 columns, sets the size of the frame to 1200x800 pixels, centers the frame on the screen, and sets the default close operation to exit the application when the frame is closed.

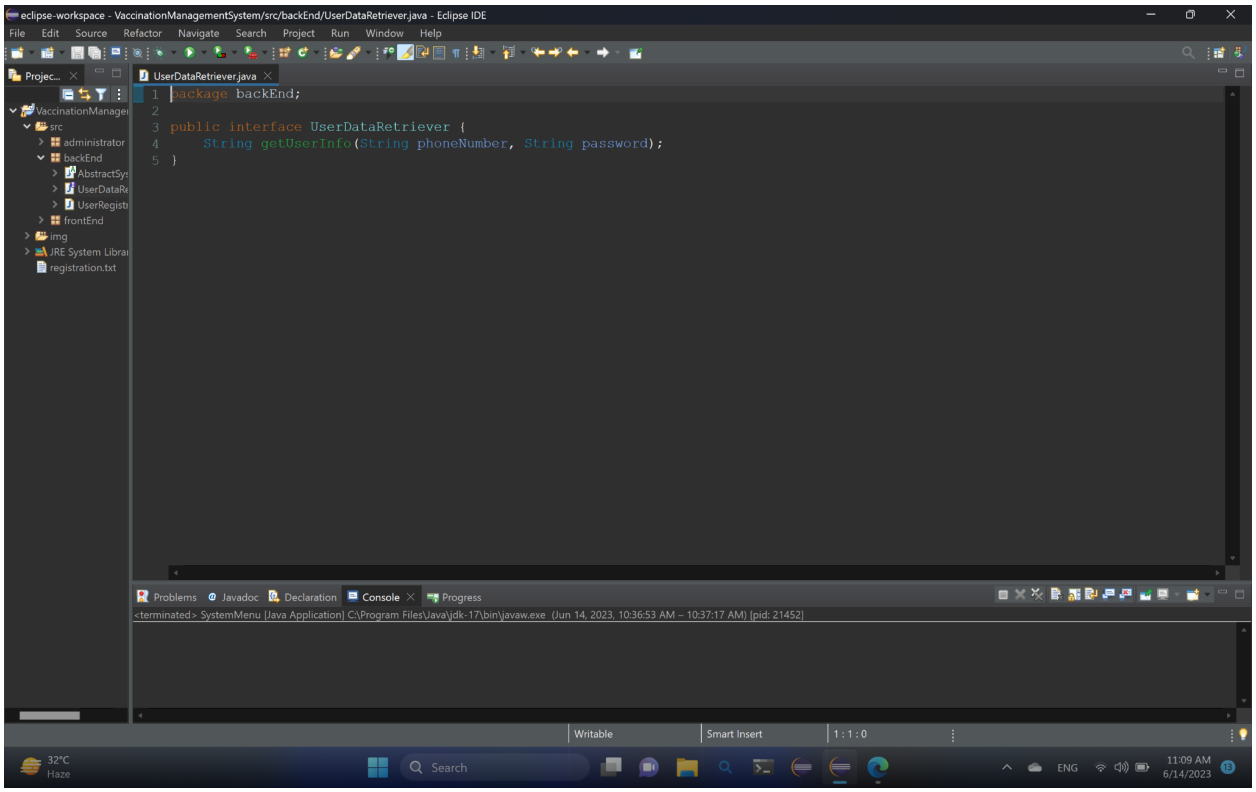
2. `protected void addButton(JButton button)`: This method takes a `JButton` parameter named `button` and adds it to the `frame`. It is marked as `protected`, which means it can be accessed by subclasses of `AbstractSystemMenu`.

3. `protected void showFrame()`: This method makes the `frame` visible on the screen. It sets the visibility of the frame to `true`, which makes it appear on the screen.

4. `protected void disposeFrame()`: This method disposes of the `frame`. It releases any resources the frame uses and removes them from the screen.

Overall, `AbstractSystemMenu` provides a basic structure for creating system menus by setting up the frame and providing methods to add buttons, show the frame, and dispose of the frame. Subclasses can extend this class to add additional functionality to the system menu.

The UserDataRetriever Class:



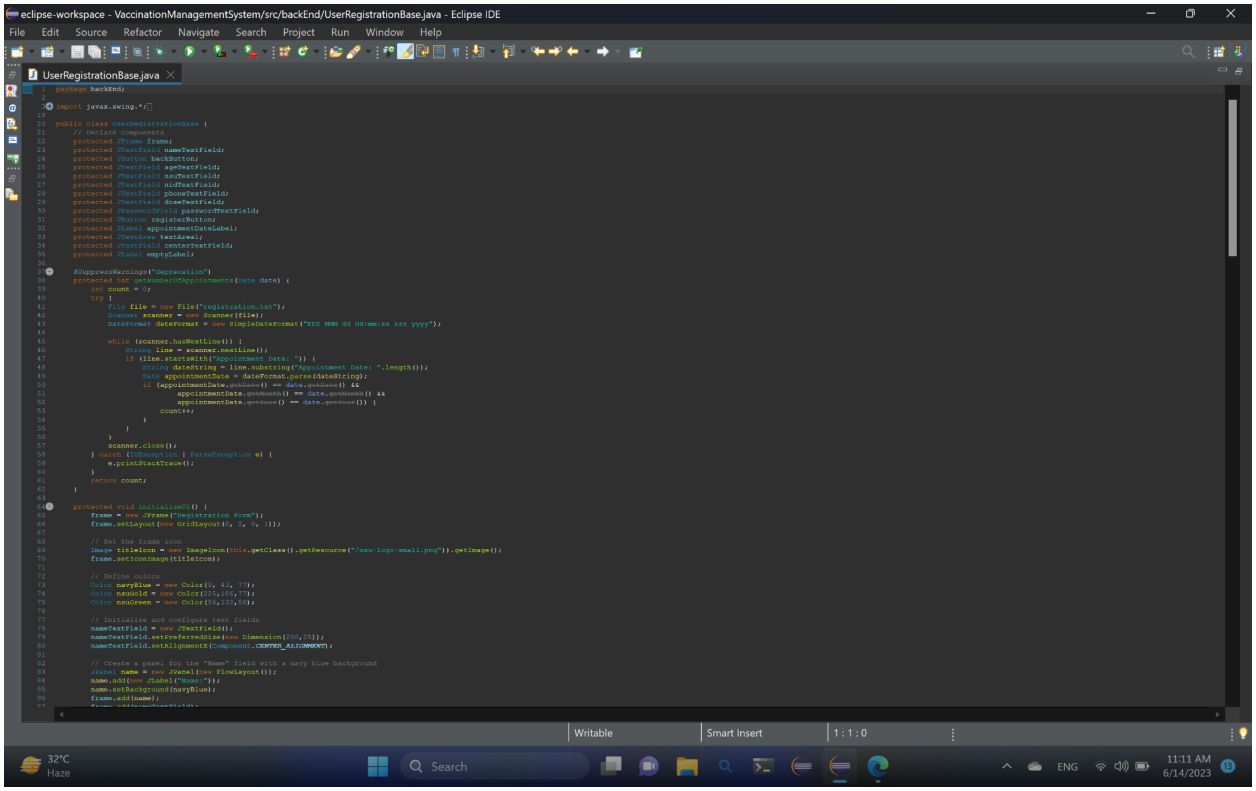
The class `UserDataRetriever` is an interface defined in the `backEnd` package. It specifies a contract for classes that retrieve user information. Let's go through the methods defined in this interface:

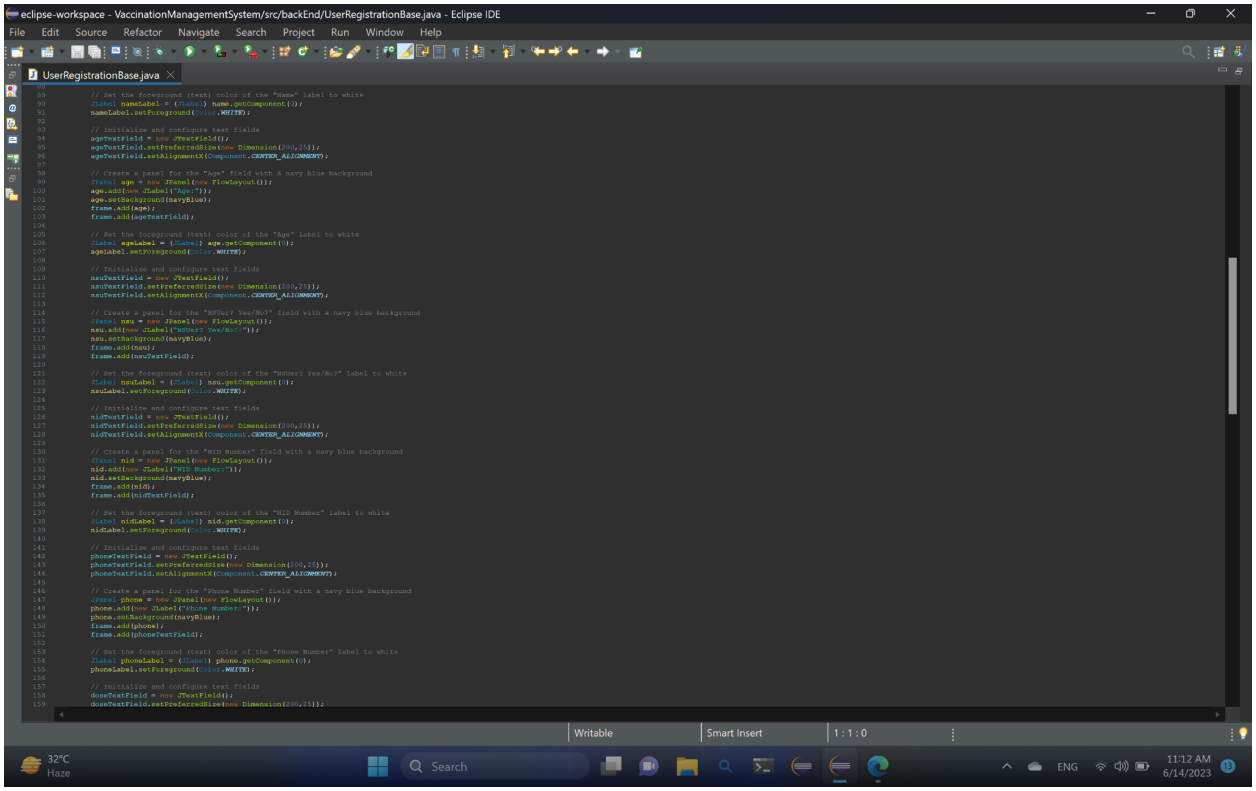
`String getUserInfo(String phoneNumber, String password)`: The interface declares this method. It takes two parameters: `phoneNumber` and `password`, both of type `String`. The purpose of this method is to retrieve user information based on the provided `phoneNumber` and `password`. It returns a `String` that represents the user information.

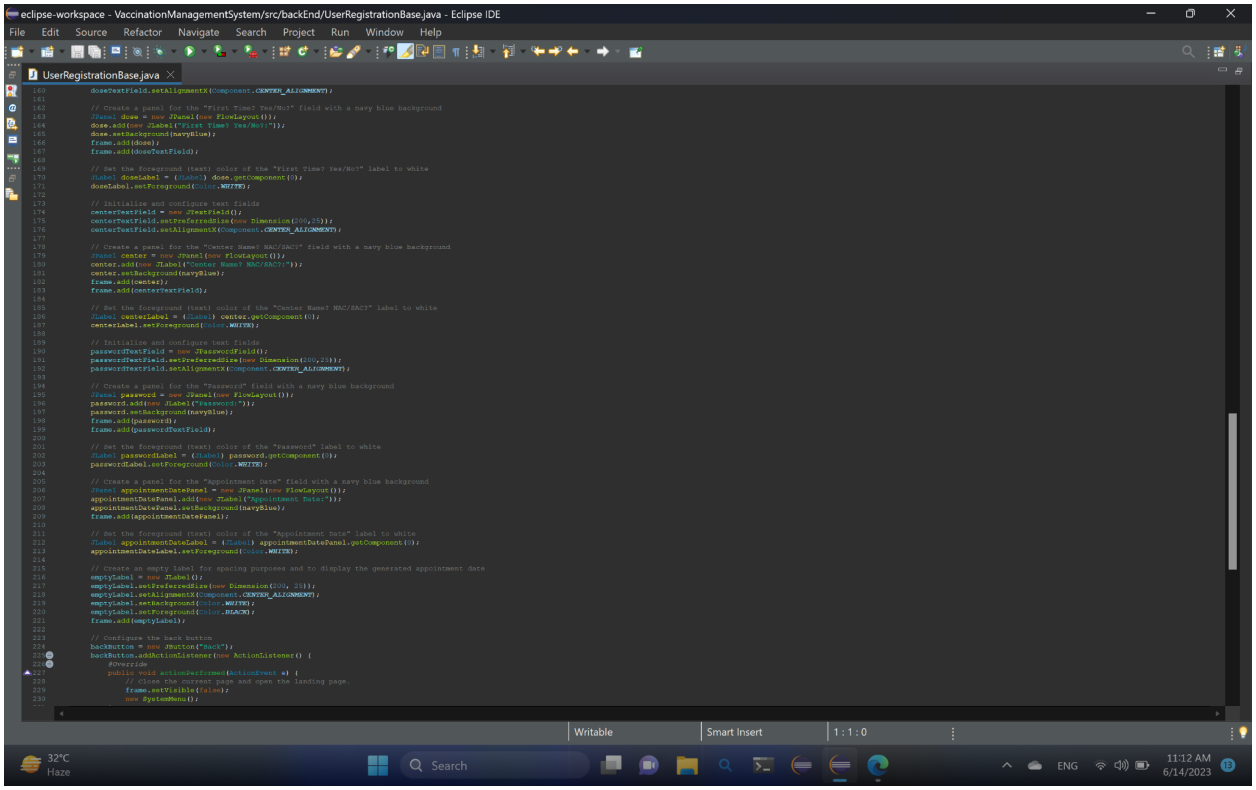
Interfaces in Java can only declare methods, but they do not provide any implementation. Classes that implement this interface need to provide their own implementation of the `getUserInfo` method.

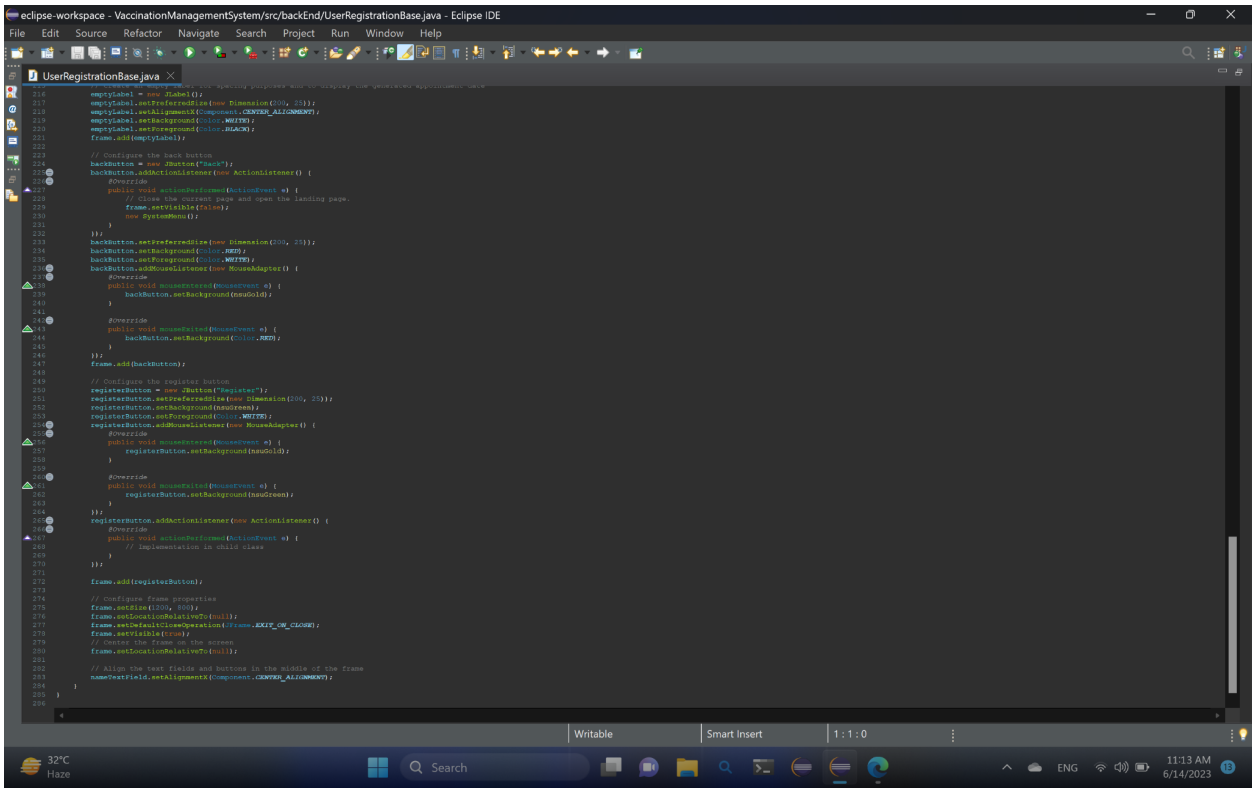
In summary, the `UserDataRetriever` interface specifies a method `getUserInfo` that retrieves user information based on a phone number and password. Classes that implement this interface must provide their implementation for this method.

The UserRegistrationBase Class:









The class `UserRegistrationBase` is defined in the `backEnd` package. It represents a base class for user registration in a graphical user interface (GUI) application. Let's describe each component and method in the class:

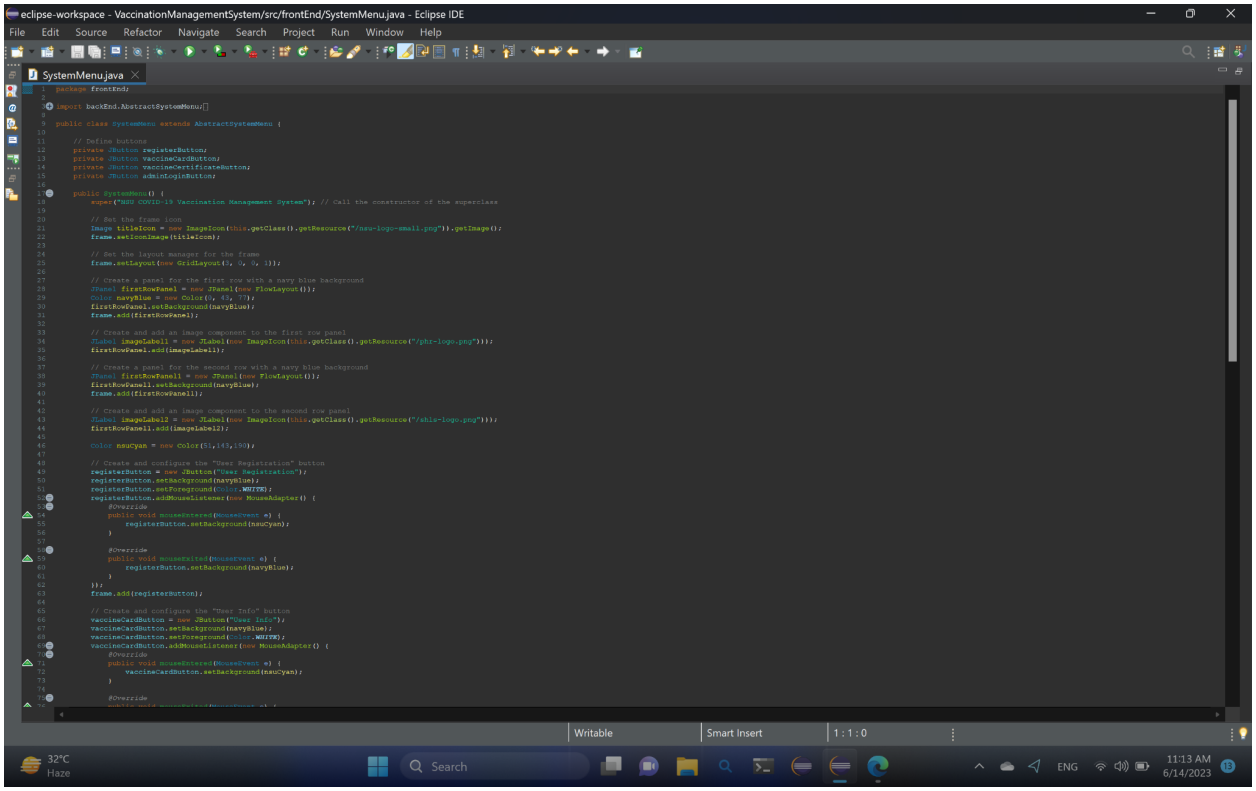
1. Class Components:
   - `frame`: A `JFrame` object that represents the main window of the GUI.
   - Text Fields:
     - `nameTextField`: A `JTextField` for entering the user's name.
     - `ageTextField`: A `JTextField` for entering the user's age.
     - `nsuTextField`: A `JTextField` for indicating whether the user is an NSUer.
     - `nidTextField`: A `JTextField` for entering the user's NID (National ID) number.
     - `phoneTextField`: A `JTextField` for entering the user's phone number.
     - `doseTextField`: A `JTextField` for indicating whether it's the user's first time.
     - `centerTextField`: A `JTextField` for entering the center's name (NAC/SAC).
     - `passwordTextField`: A `JPasswordField` for entering the user's password.
   - `backButton`: A `JButton` for returning to the previous page.
   - `registerButton`: A `JButton` for registering the user.
   - `appointmentDateLabel`: A `JLabel` for displaying the appointment date.
   - `textArea1`: A `JTextArea` for displaying text (currently unused).
   - `emptyLabel`: A `JLabel` for spacing and displaying the generated appointment date.

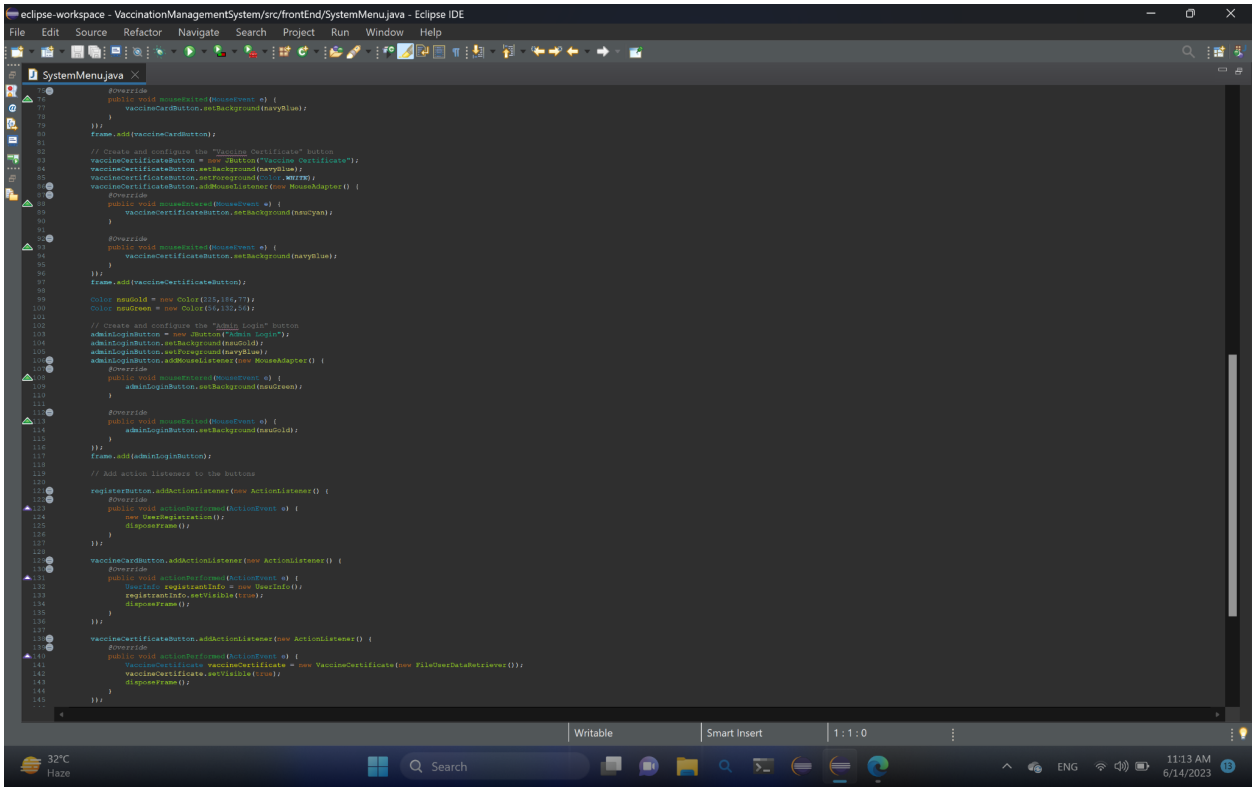
2. Method `getNumberOfAppointments(Date date)`: This method takes a `Date` parameter named `date` and returns an `int`. It reads from a file named "registration.txt" to count the appointments made on a specific date. It uses a `Scanner` to read the file line by line, extracts the appointment date from each line, compares it with the provided `date`, and increments the `count` if there's a match. It handles potential exceptions like `IOException` and `ParseException`.

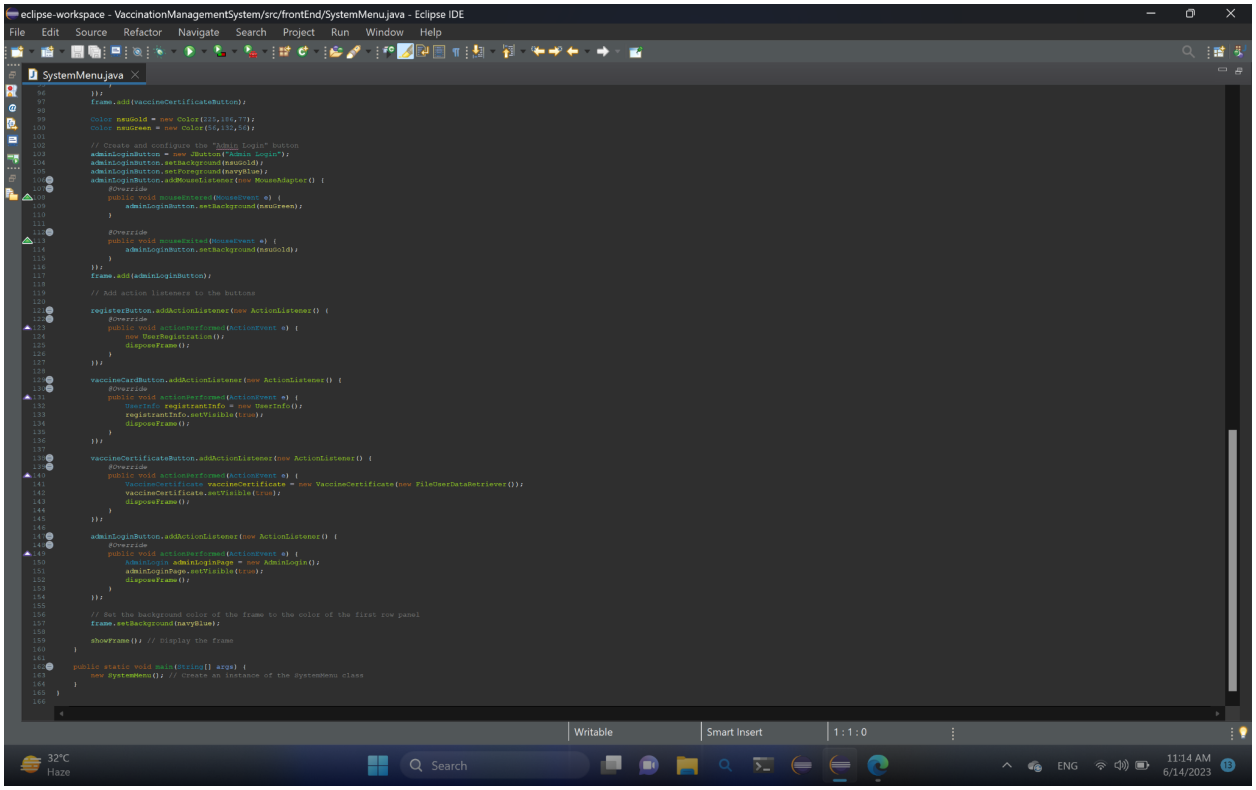
3. Method `initializeUI()`: This method sets up the user registration GUI. It configures various components of the GUI, such as the frame layout, colors, text fields, buttons, labels, and event listeners. It sets the size, location, and close operation of the frame. It also makes the frame visible and centers it on the screen.

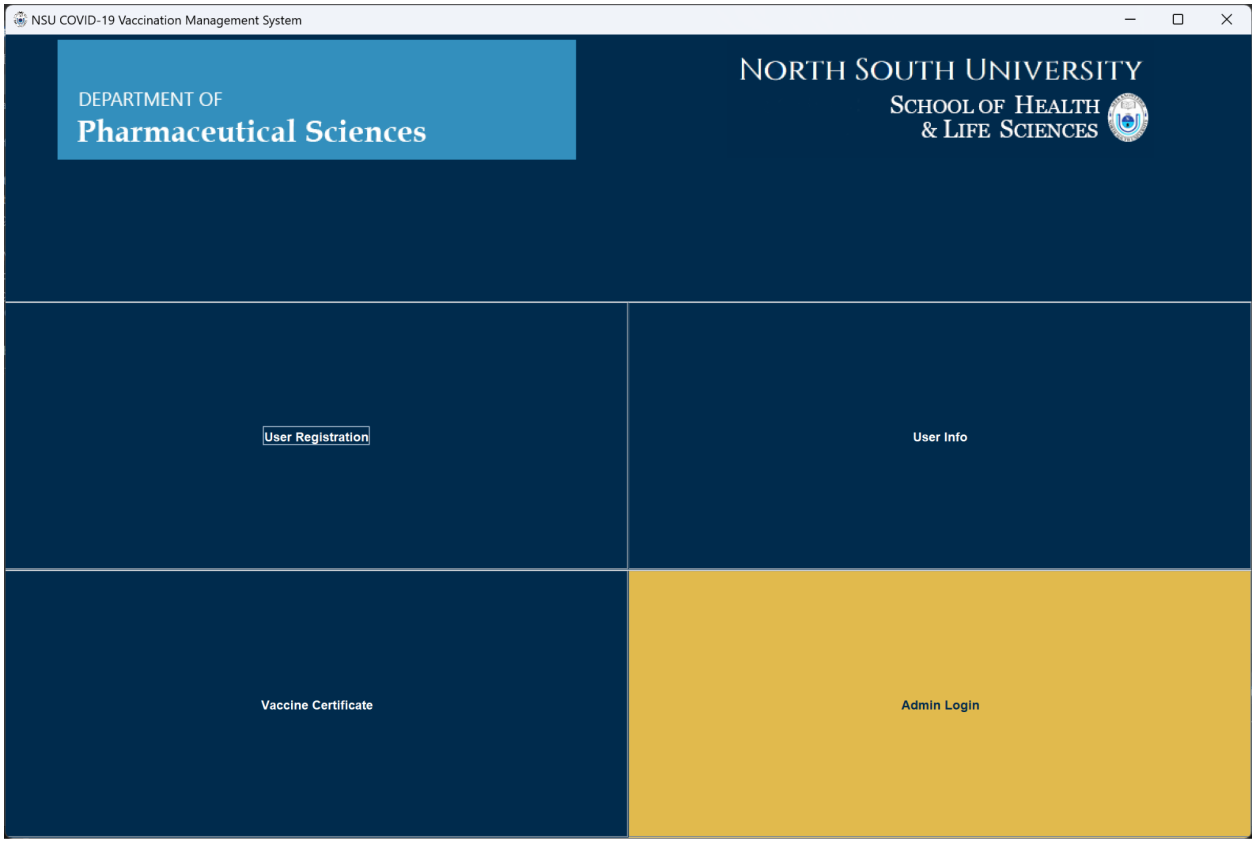
Overall, the `UserRegistrationBase` class provides a basic framework for building a user registration form in a GUI application. Subclasses can extend this class to add specific functionality to handle user registration events and implement the `actionPerformed` method for the `registerButton` to define the registration logic.

The SystemMenu Class:






eclipse-workspace - VaccinationManagementSystem/src/frontEnd/SystemMenu.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
SystemMenu.java
Writable
Smart Insert
1 : 1 : 0
32°C
Haze
Search
ENG
11:14 AM
6/14/2023


NSU COVID-19 Vaccination Management System
DEPARTMENT OF
Pharmaceutical Sciences
NORTH SOUTH UNIVERSITY
SCHOOL OF HEALTH
& LIFE SCIENCES
User Registration
User Info
Vaccine Certificate
Admin Login

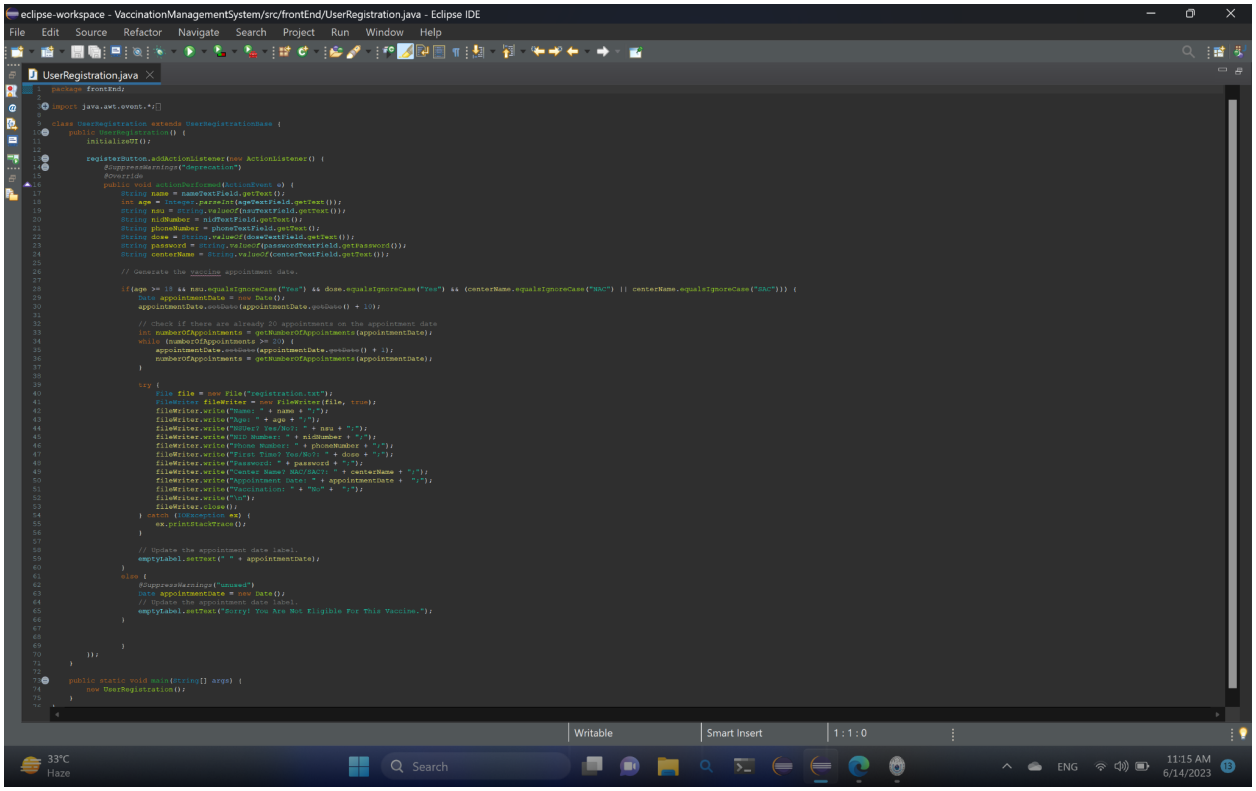
The `SystemMenu` class is the main class responsible for creating and managing the system menu of the NSU COVID-19 Vaccination Management System. It extends the `AbstractSystemMenu` class and implements the system menu functionality.

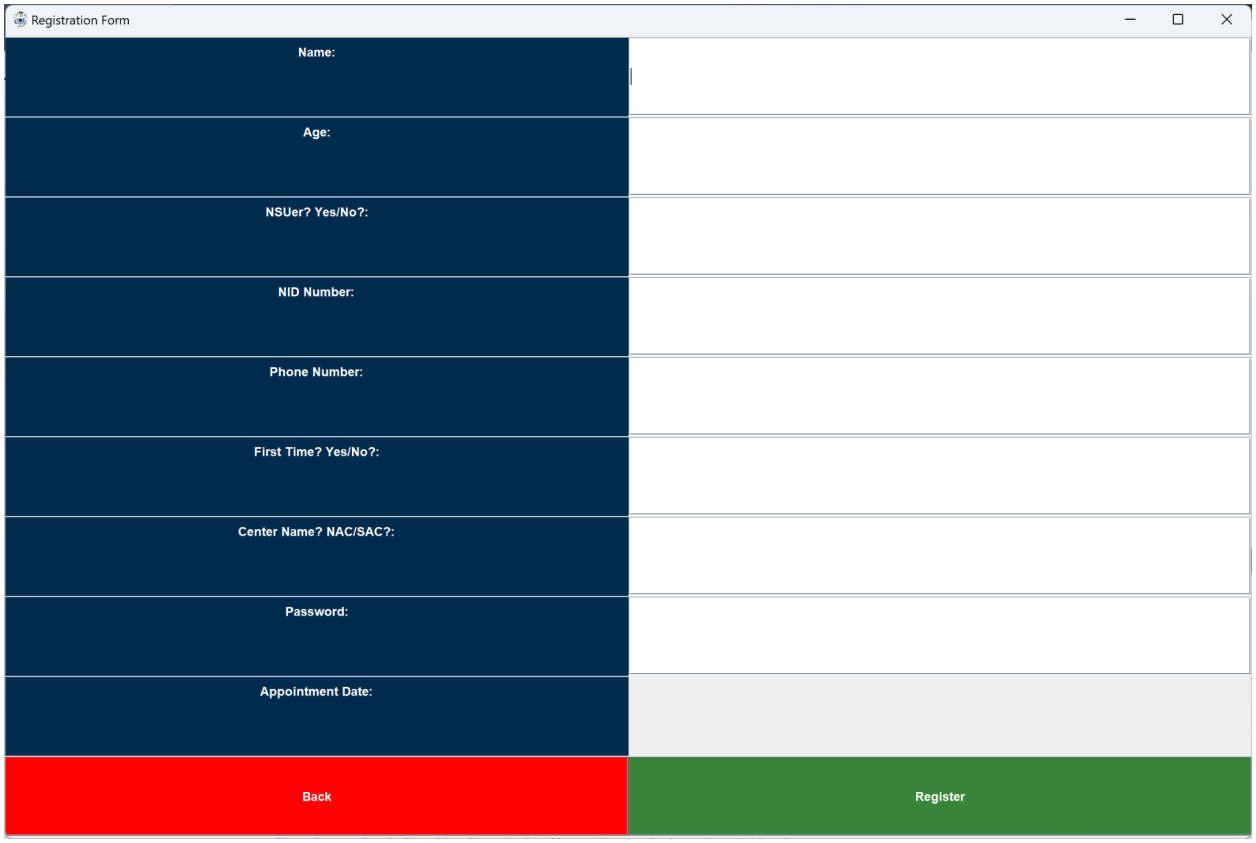
The class contains the following methods:

1. `SystemMenu()` constructor:
   - Initializes the system menu by calling the superclass's constructor (`AbstractSystemMenu`) with the specified title.
   - Sets the frame icon using an image file (`nsu-logo-small.png`).
   - Configures the layout manager of the frame as a `GridLayout` with 3 rows and 0 columns.
   - Creates two panels (`firstRowPanel` and `firstRowPanel1`) with a navy blue background color for the first and second rows.
   - Adds an image component (`JLabel`) to each panel using image files (`phr-logo.png` and `shls-logo.png`).
   - Creates and configures four buttons (`registerButton`, `vaccineCardButton`, `vaccineCertificateButton`, `adminLoginButton`) for user registration, user info, vaccine certificate, and admin login.
   - Adds mouse listeners to change the background color of the buttons when the mouse enters or exits.
   - Adds the buttons to the frame.
   - Adds action listeners to the buttons to handle button click events and perform corresponding actions.
   - Sets the background color of the frame to the color of the first row panel.
   - Calls the `showFrame()` method to display the frame.

2. `main(String[] args)` method:
   - The entry point of the program.
   - Creates an instance of the `SystemMenu` class, which initializes and displays the system menu.

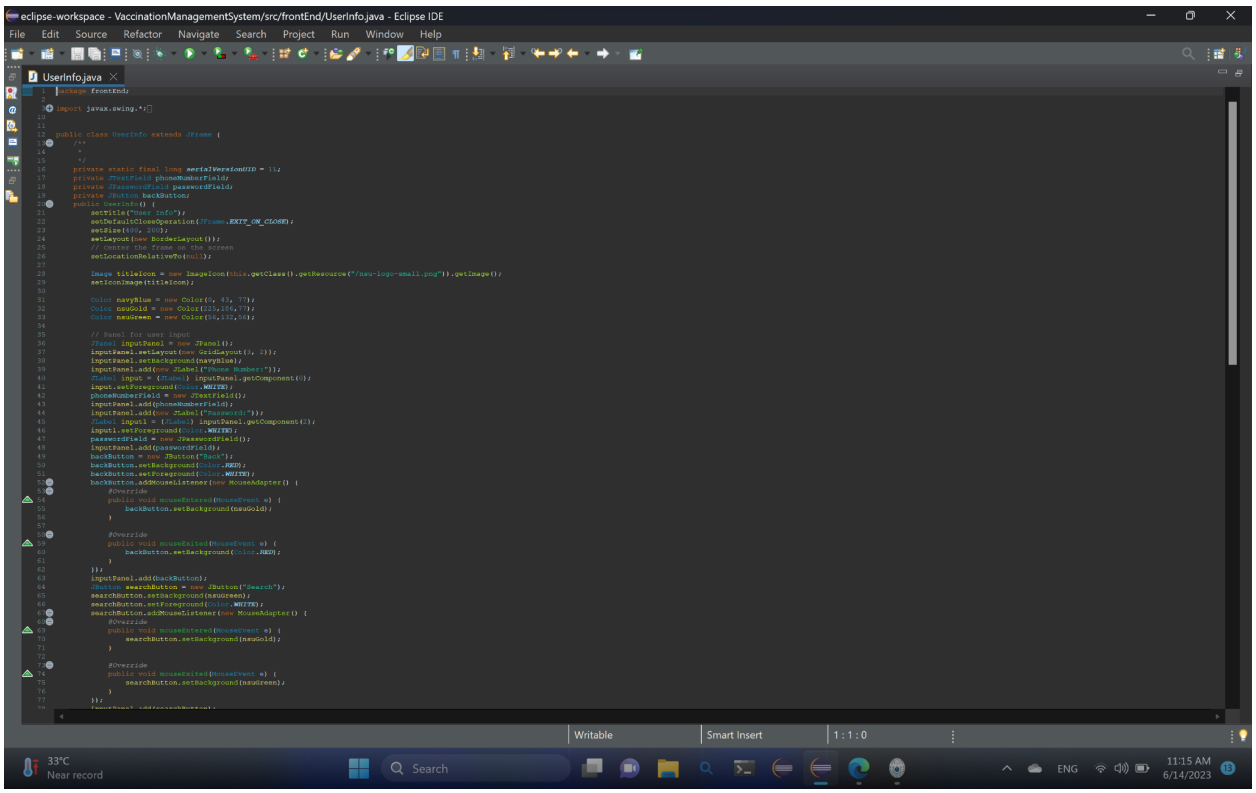
The UserRegistration Class:

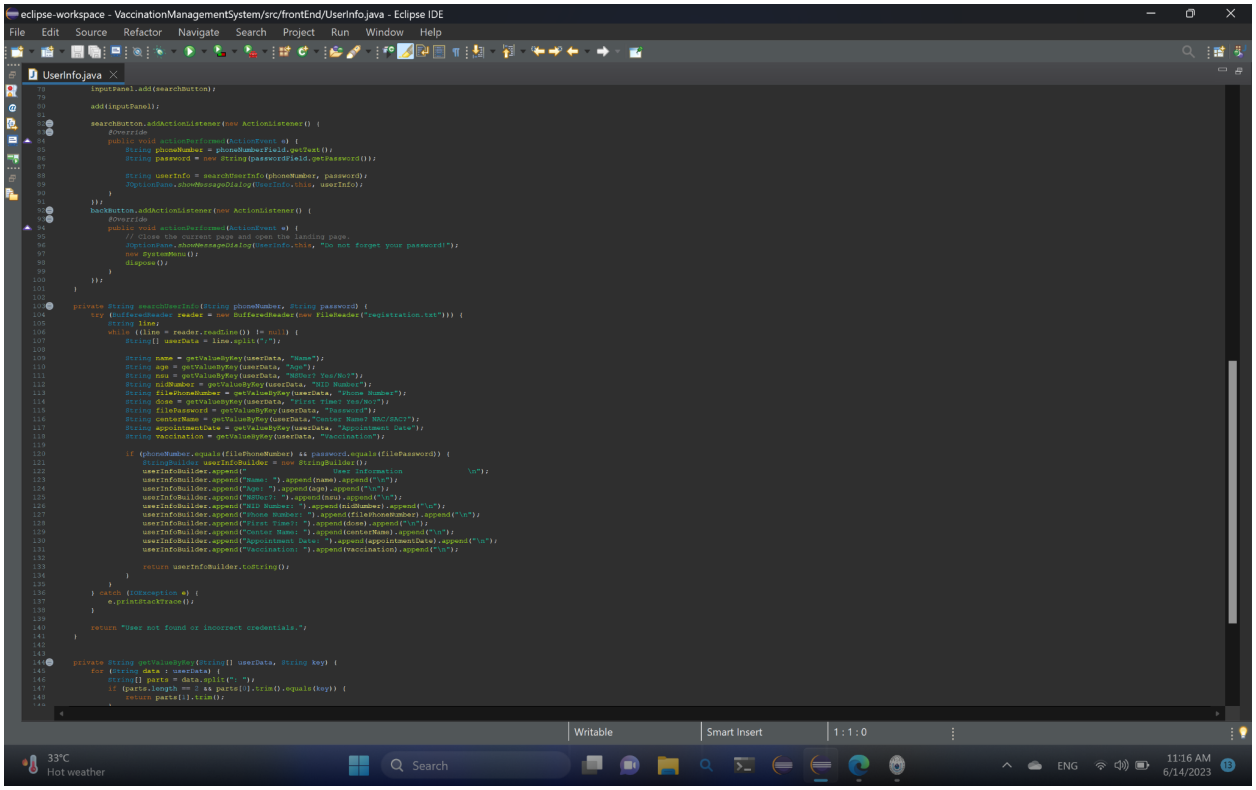


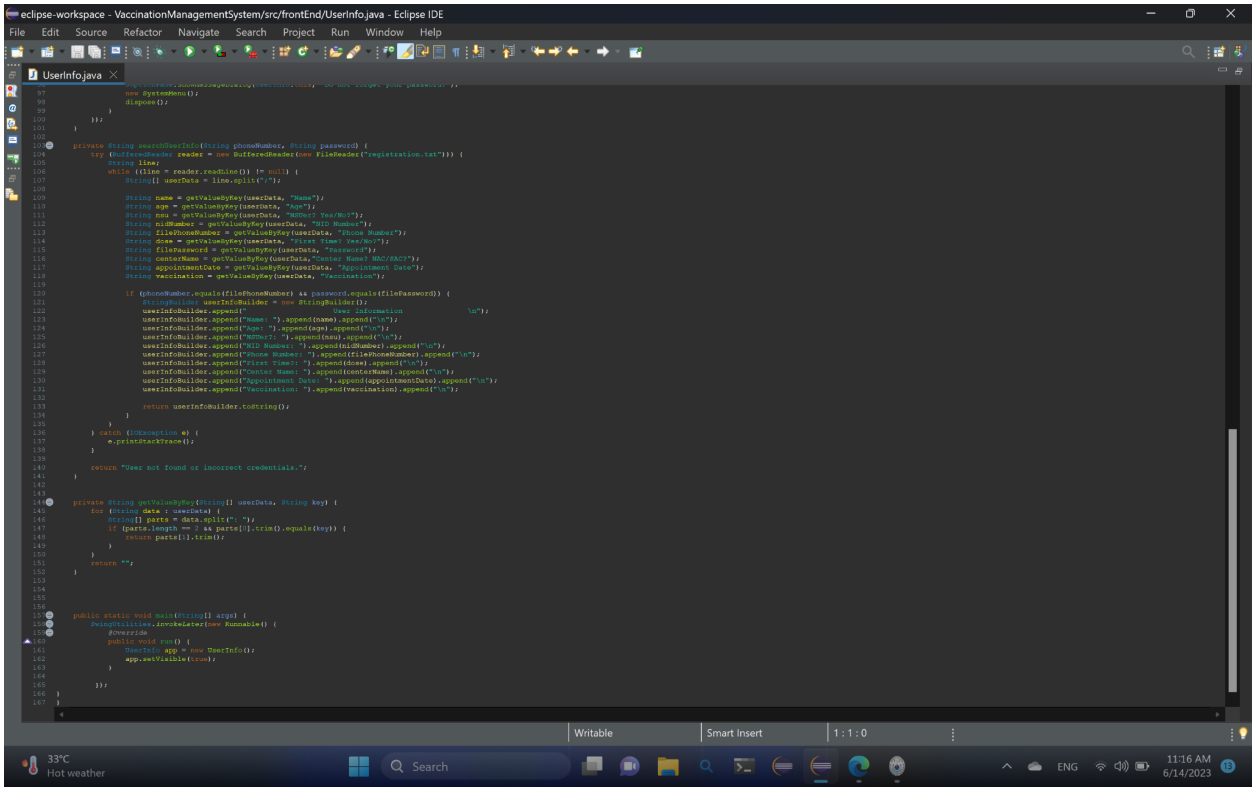


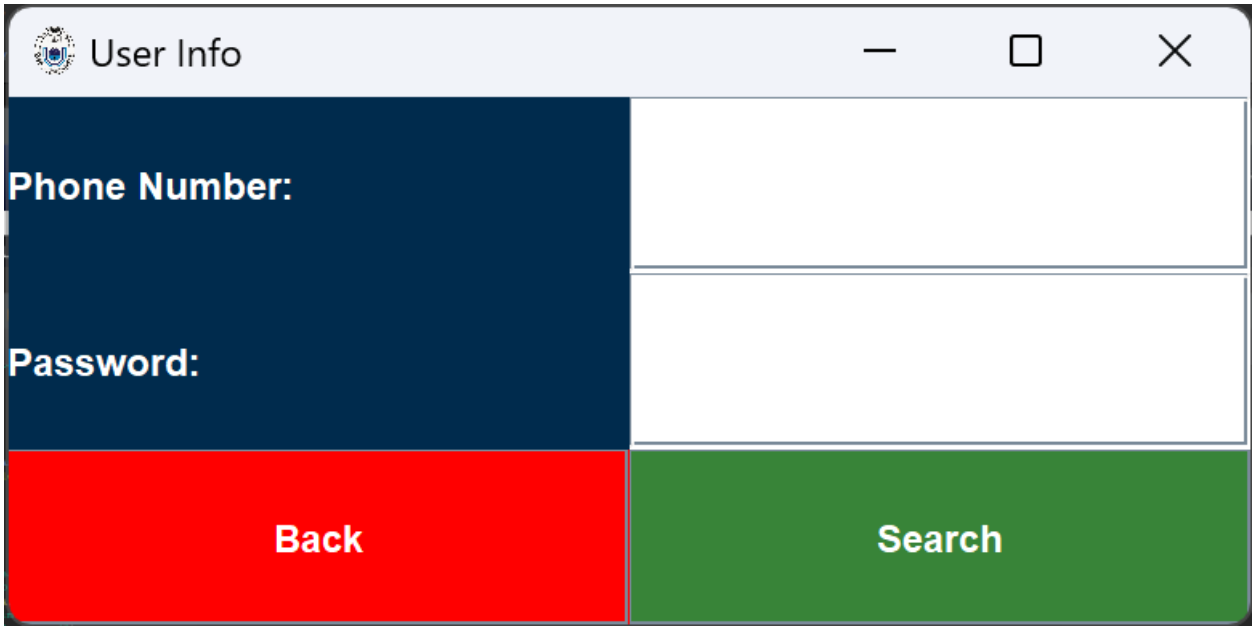
- This class extends the `UserRegistrationBase` class, which implies it inherits some functionality from the base class.
- The class initializes the user interface (UI) by calling the `initializeUI()` method.
- It adds an action listener to the `registerButton` to handle the button click event.
- The `actionPerformed` method is called when the button is clicked, and it retrieves the user input from various text fields such as name, age, NSU status, NID number, phone number, vaccine dose status, password, and center name.
- Based on certain conditions (age, NSU status, dose status, center name), the class generates an appointment date and checks if there are already 20 appointments. If so, it increments the date until it finds an available date with less than 20 appointments.
- The user registration data is then written to a file named "registration.txt" in a specific format.
- The appointment date and other information are displayed in the UI through the `emptyLabel`.
- If the user does not meet the eligibility criteria, a message is displayed in the `emptyLabel` stating that they are not eligible for the vaccine.
- The `main` method is the entry point of the program.
- It creates an instance of the `UserRegistration` class, initializing and displaying the user registration UI.

The UserInfo Class:



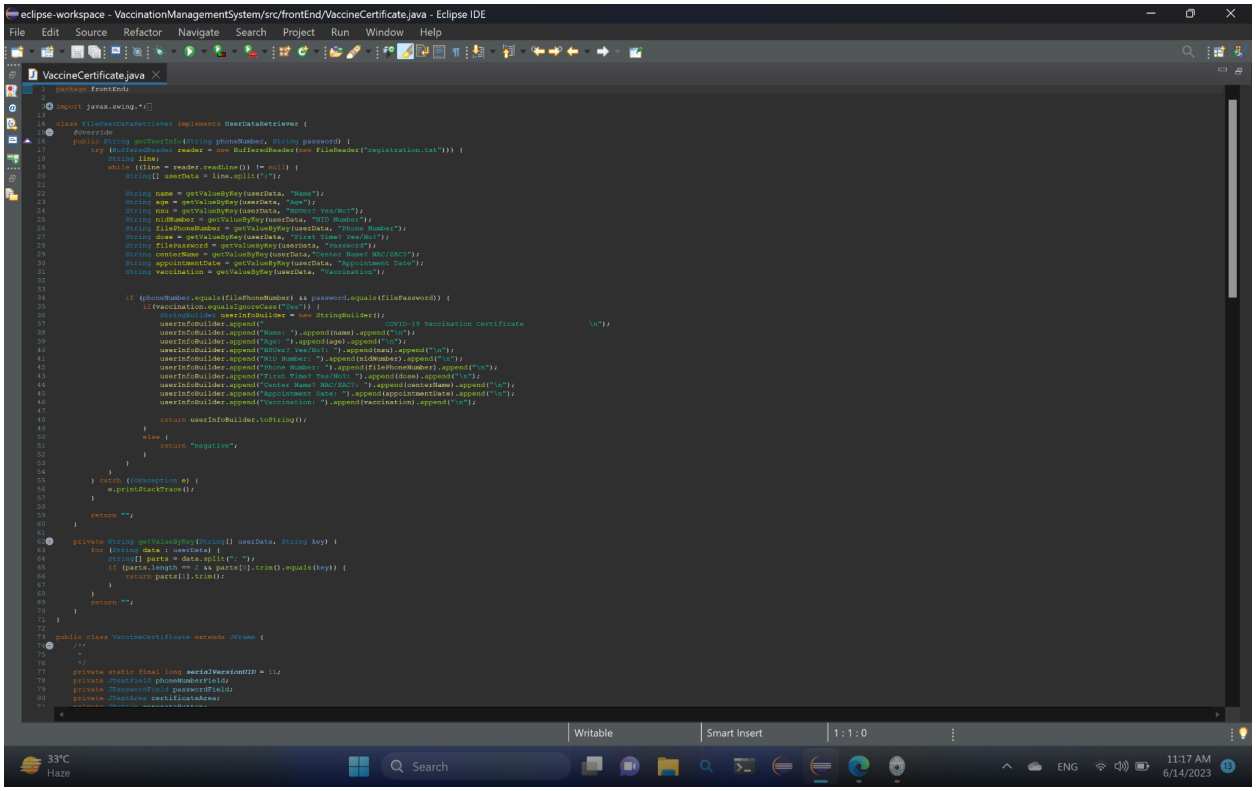

eclipse-workspace - VaccinationManagementSystem/src/frontEnd/UserInfo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
UserInfo.java
Writable
Smart Insert
1 : 1 : 0
33°C
Hot weather
Search
ENG
11:16 AM
6/14/2023


eclipse-workspace - VaccinationManagementSystem/src/frontEnd/UserInfo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
UserInfo.java
Writable
Smart Insert
1 : 1 : 0
33°C
Hot weather
Search
ENG
11:16 AM
6/14/2023



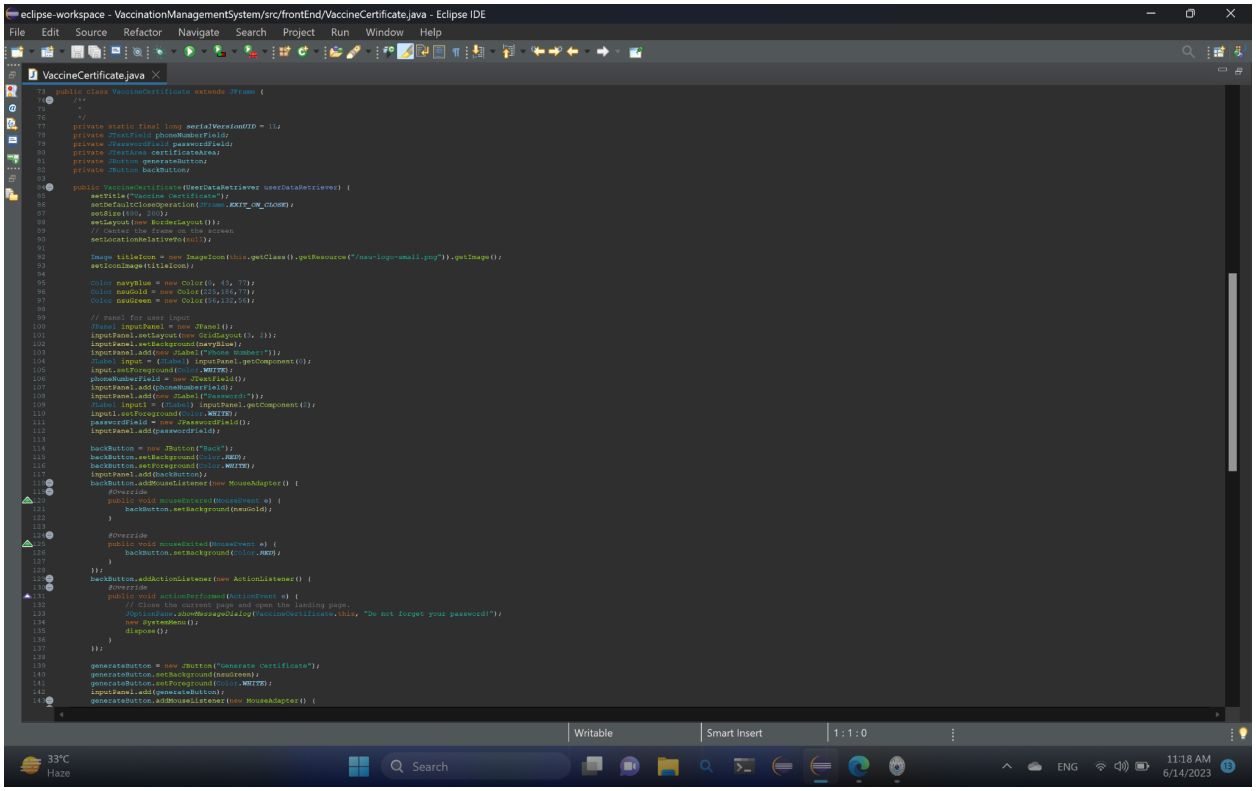
- This class extends the `JFrame` class, which allows it to create a window-based user interface.
- The class sets up the frame's title, size, layout, and close operation behavior.
- It creates and configures UI components such as text fields, labels, buttons, and panel.
- The `searchButton` and `backButton` have event listeners attached to them to handle mouse events.
- The `searchButton`'s `actionPerformed` method is called when clicking the button. It retrieves the user input (phone number and password) and calls the `searchUserInfo` method to search for the user information based on the provided credentials. The retrieved information is then displayed using a `JOptionPane`.
- The `backButton`'s `actionPerformed` method is called when clicking the button. It displays a message dialog and opens the `SystemMenu` class while closing the current page.
- The `searchUserInfo` method reads the "registration.txt" file line by line and searches for a matching phone number and password. If a match is found, it builds a `userInfoBuilder` string with the retrieved user information and returns it. Otherwise, it returns a message indicating that the user was not found or the credentials were incorrect.
- The `getValueByKey` method takes an array of user data and a key and searches for a corresponding value by splitting each data entry. If a match is found, it returns the corresponding value. If no match is found, it returns an empty string.
- The `main` method creates an instance of the `UserInfo` class and sets it visible on the event dispatch thread.

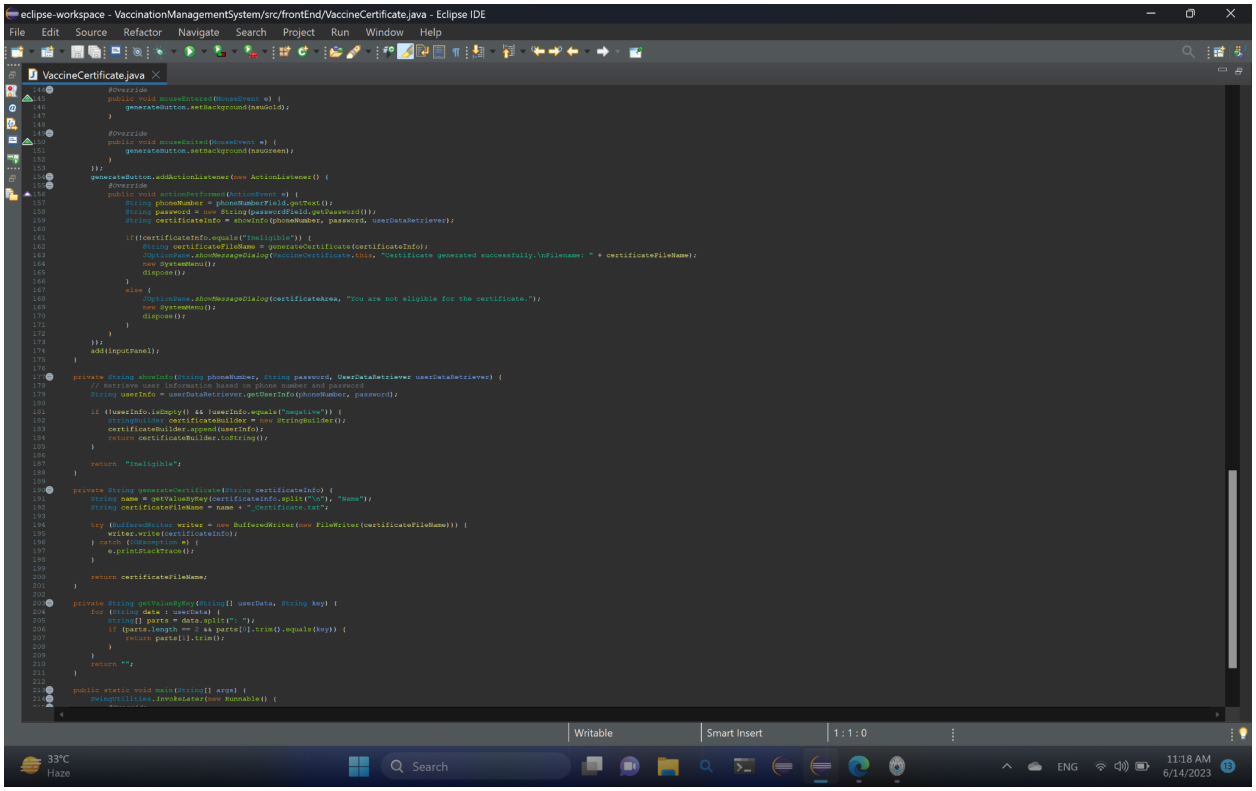
The FileUserDataRetriever Class:

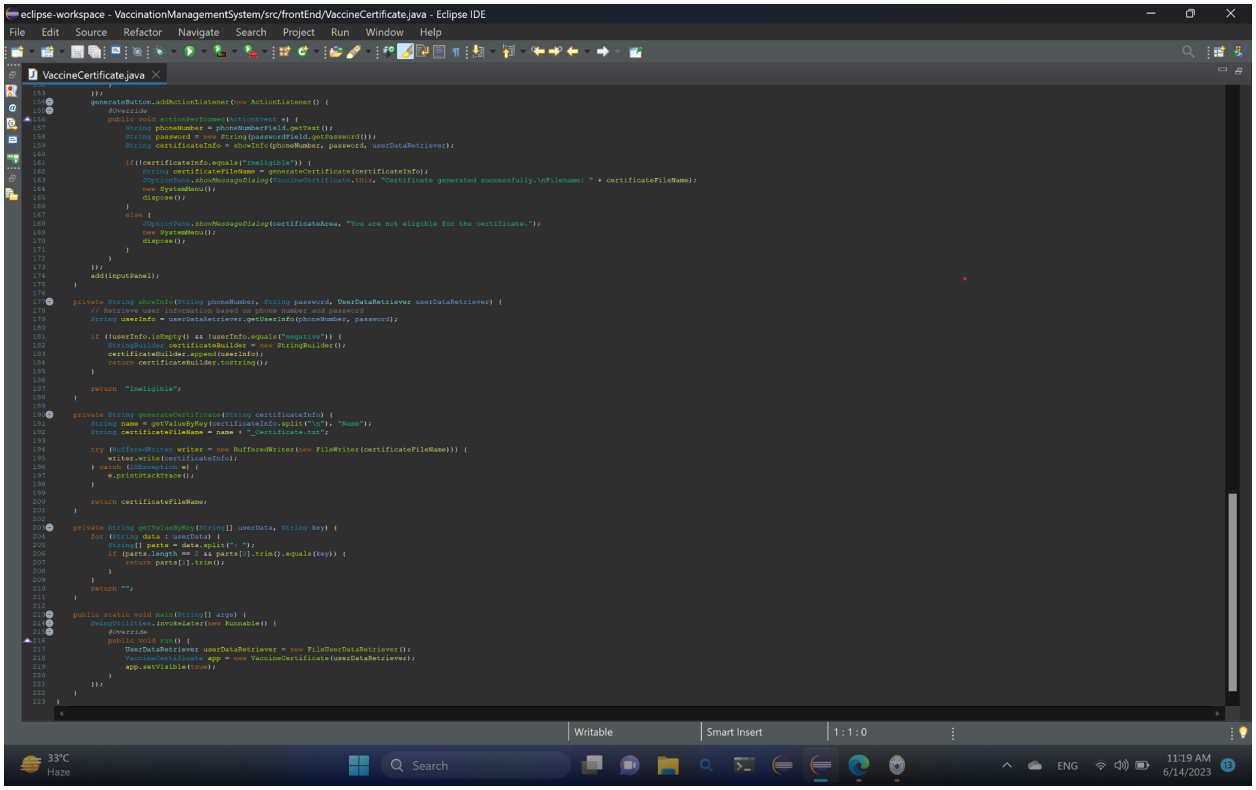


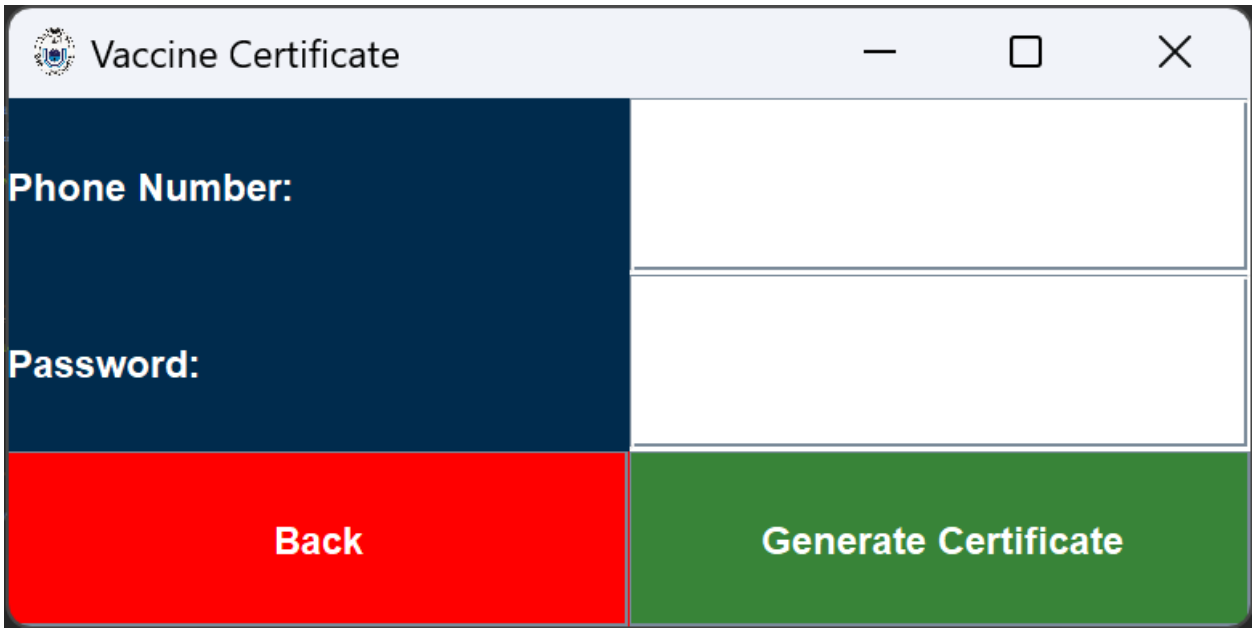
- This class implements the `UserDataRetriever` interface, which defines the contract for retrieving user information.
- It overrides the `getUserInfo` method from the interface to retrieve user information from a file (`registration.txt`).
- The `getUserInfo` method takes a phone number and password as parameters and returns the corresponding user information as a string.
- It reads each line from the file, splits the line into an array of data, and extracts the relevant information based on keys.
- It compares the provided phone number and password with the data from the file to find a matching user.
- If a match is found and the user has been vaccinated, it constructs a string with the user's information and returns it.
- If the user has not been vaccinated, it returns the string "negative".
- The `getValueByKey` method is a helper method that searches for a specific key in an array of user data and returns the corresponding value.

The VaccineCertificate Class:





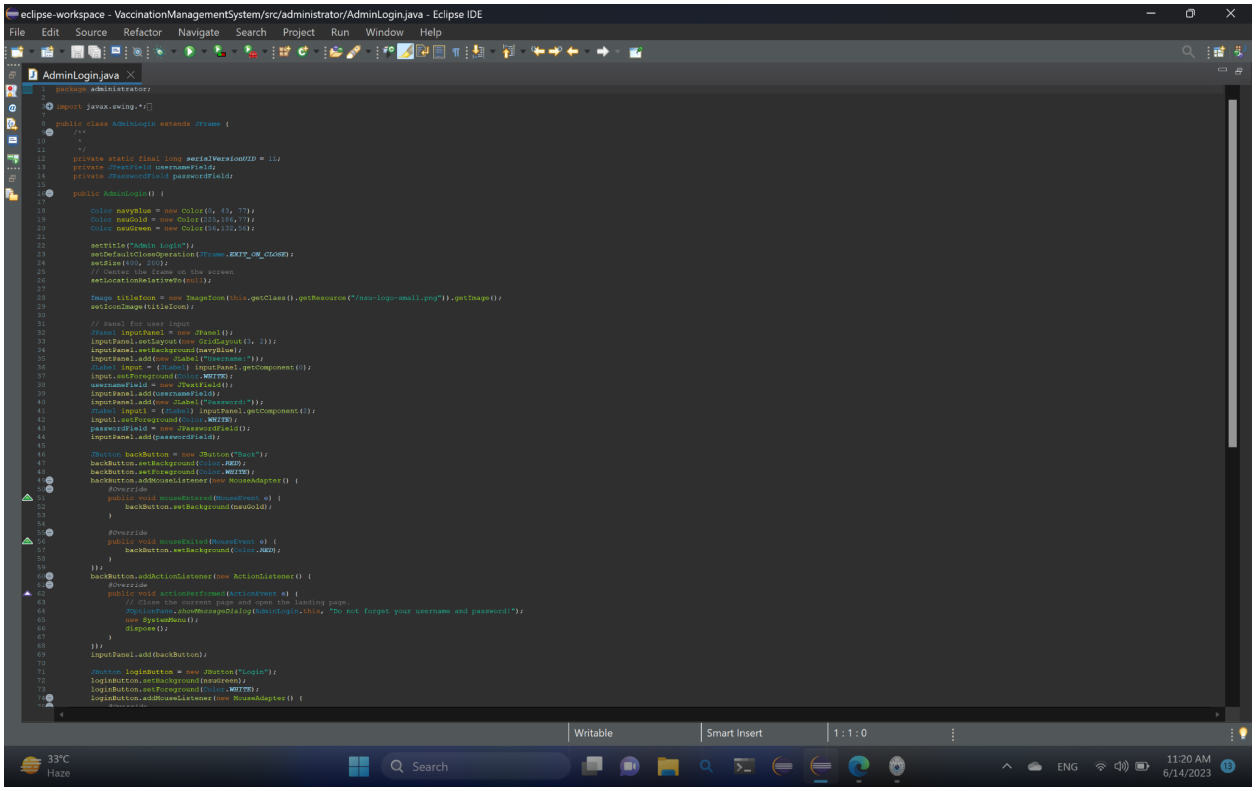


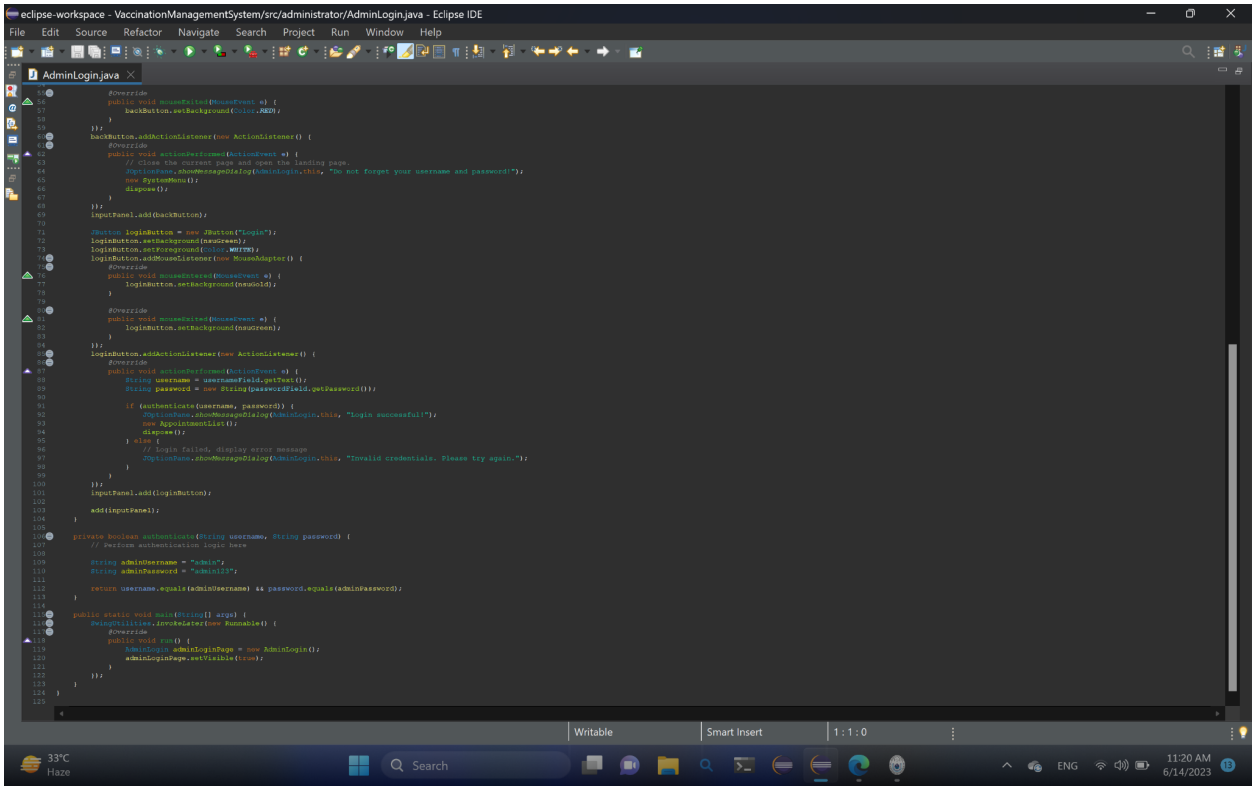


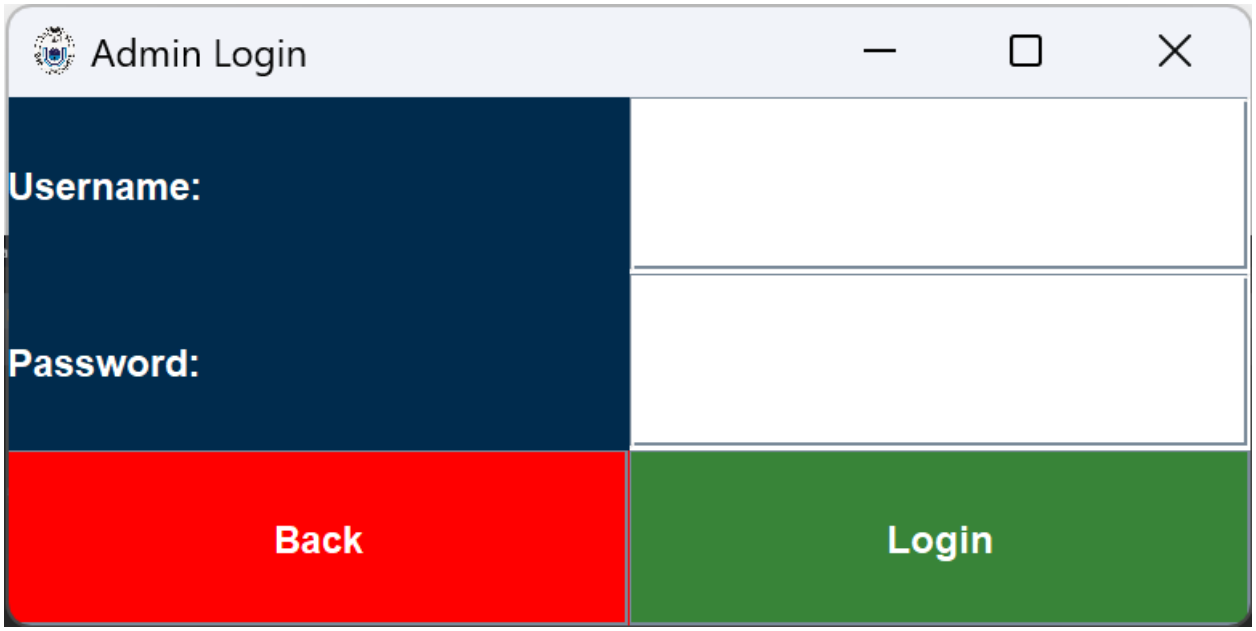
- This class extends the `JFrame` class to create a graphical user interface (GUI) for generating a vaccine certificate.
- It takes an instance of the `UserDataRetriever` interface as a parameter in the constructor.
- The class defines various GUI components such as text fields, buttons, and labels.
- It sets up the layout and appearance of the frame using `BorderLayout` and sets the frame's icon.

- The `showInfo` method retrieves user information by calling the `getUserInfo` method of the `UserDataRetriever` interface.
- It checks if the retrieved information is valid and not "negative".
- If valid, it appends the user's information to a `StringBuilder` and returns it as a string.
- If invalid, it returns the string "Ineligible".
- The `generateCertificate` method generates a certificate file by writing the certificate information to a file with a name based on the user's name.
- The `getValueByKey` method is a helper method that searches for a specific key in an array of user data and returns the corresponding value.
- The `main` method creates an instance of `FileUserDataRetriever`, passes it to the `VaccineCertificate` constructor, and displays the GUI.

The AdminLogin Class:







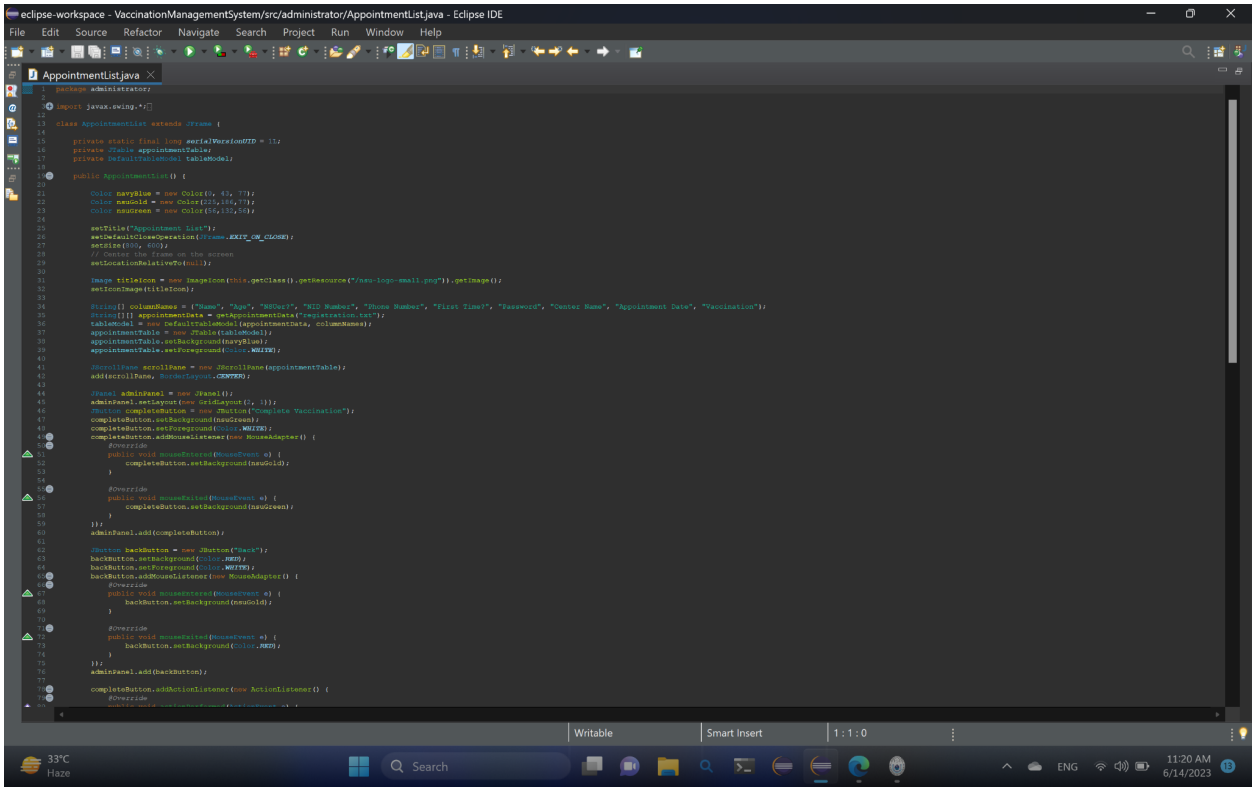
- This class extends the `JFrame` class to create a graphical user interface (GUI) for the admin login page.
- It provides a form for the admin to enter their username and password.
- The class sets the title, size, and close operation of the frame and centers it on the screen.
- It defines various GUI components, such as text fields, buttons, and labels.
- The `authenticate` method is a helper method that performs the authentication logic by comparing the entered username and password with predefined admin credentials.

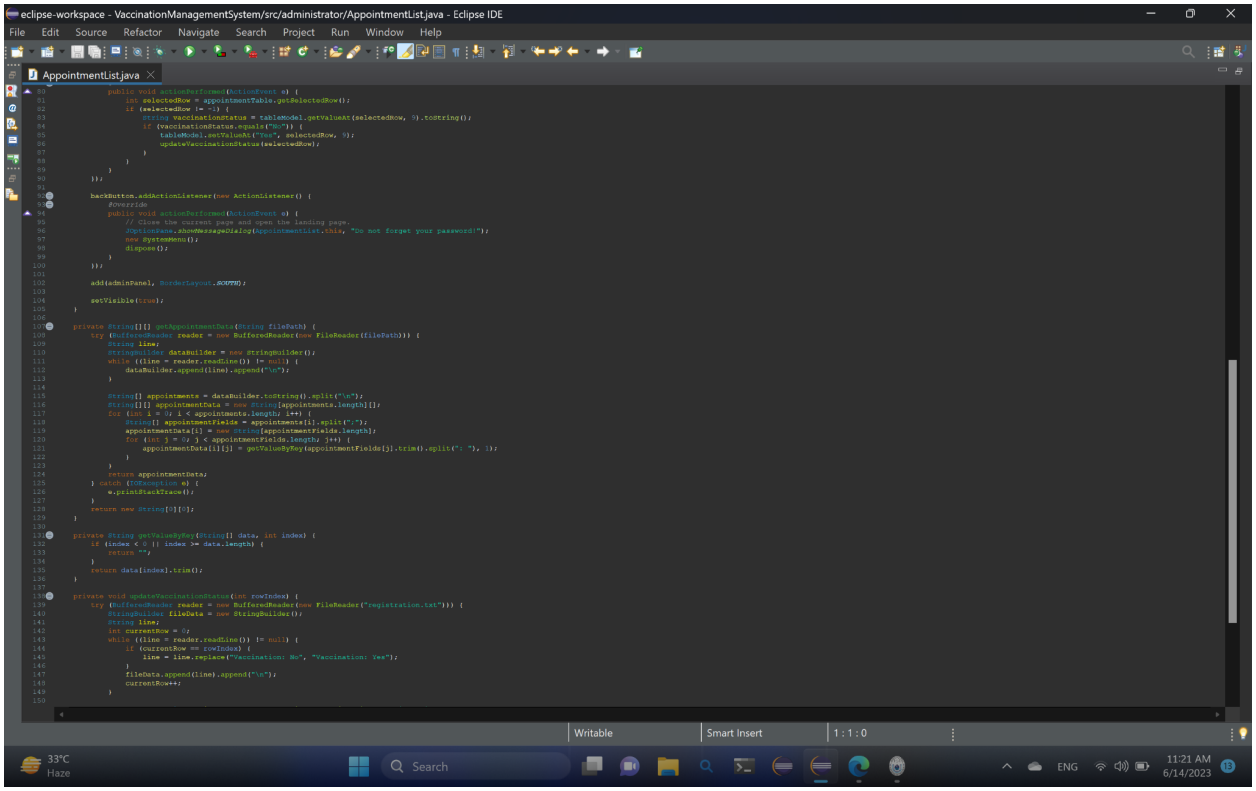
- The `authenticate` method returns `true` if the provided username and password match the admin credentials and `false` otherwise.
- The `main` method creates an instance of `AdminLogin` and displays the GUI.

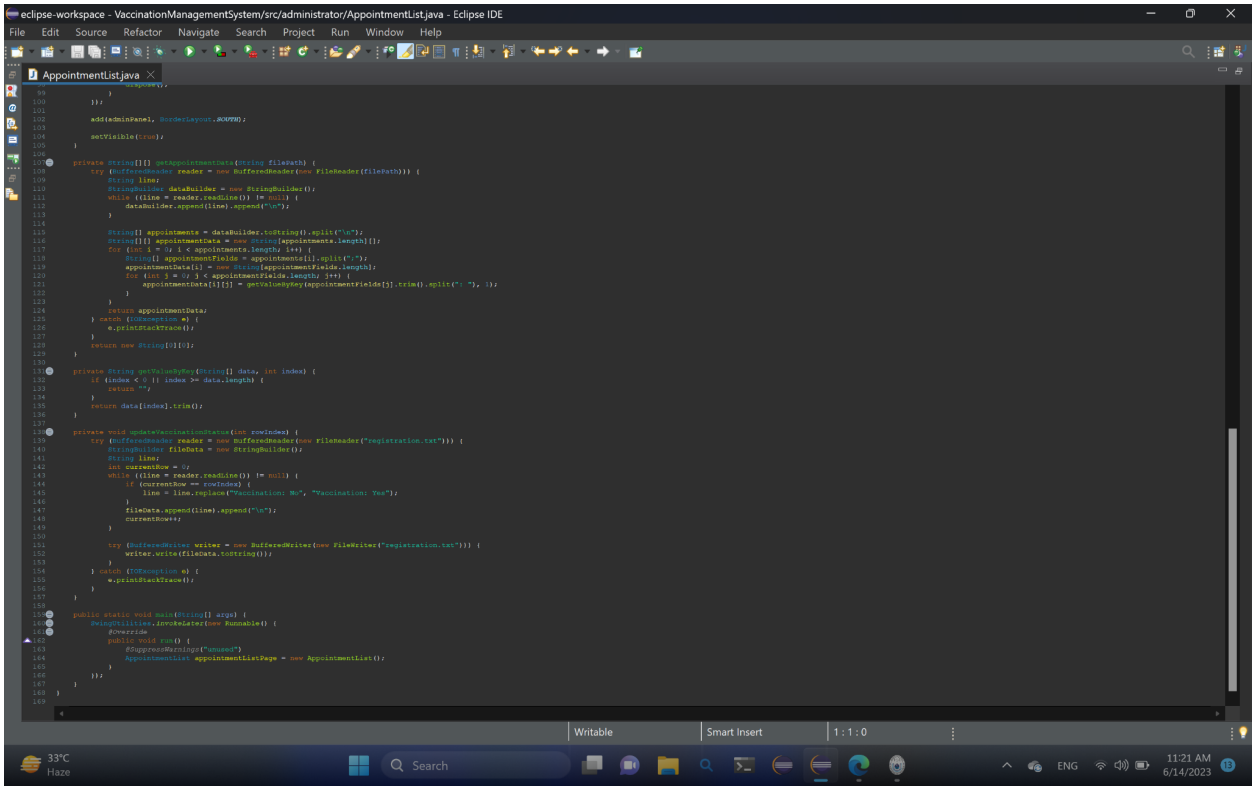
In the GUI:
- A panel (`inputPanel`) contains a grid layout to arrange the username, password fields, and buttons.
- The panel has a navy blue background color.
- The username and password fields are represented by `JTextField` and `JPasswordField`, respectively.
- The "Back" button allows the admin to return to the previous page (`SystemMenu`) and displays a message reminding them not to forget their username and password.
- The "Login" button triggers the login process when clicked.
  - It retrieves the entered username and password.
  - It calls the `authenticate` method to check if the provided credentials are valid.
  - A success message is displayed if the credentials are valid and a new `AppointmentList` page is opened.
  - If the credentials are invalid, an error message is displayed.

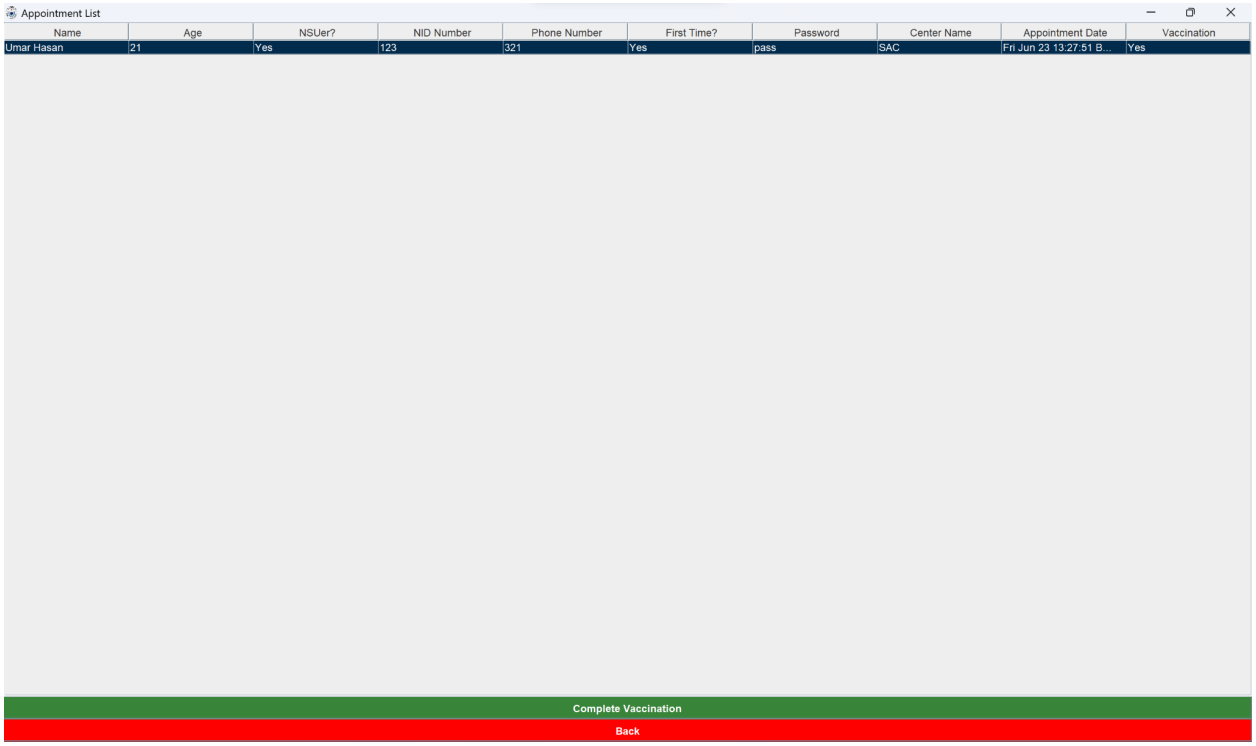
Overall, this code provides a GUI for the admin to enter their credentials and log in. It performs authentication using predefined admin credentials and allows access to the admin features upon successful login.

The AppointmentList Class:









- This class extends the `JFrame` class to create a graphical user interface (GUI) for displaying the appointment list.
- It provides a table that shows appointment data fetched from a file.
- The class sets the frame's title, size, and close operation and centers it on the screen.

- It defines a table model (`DefaultTableModel`) and a table (`JTable`) to display the appointment data.
- The appointment data is read from a file (`registration.txt`) and loaded into the table model.
- The class also includes buttons for completing vaccination and returning to the previous page.
- The `getAppointmentData` method reads the appointment data from the file and returns it as a two-dimensional array of strings.
- The `getValueByKey` method extracts a specific value from an array of data by providing the index.
- The `updateVaccinationStatus` method updates the vaccination status of an appointment in the file when the "Complete Vaccination" button is clicked.
- The `main` method creates an instance of `AppointmentList` and displays the GUI.

In the GUI:
- The appointment data is displayed in a table with columns such as "Name," "Age," "NSUer?," "NID Number," etc.
- The table has a navy blue background color and white foreground color.
- The table is added to a scroll pane to enable scrolling if the content exceeds the visible area.
- The GUI includes a panel (`adminPanel`) with a grid layout that contains the "Complete Vaccination" and "Back" buttons.
- The "Complete Vaccination" button allows marking an appointment as completed by changing the vaccination status from "No" to "Yes" in the table and updating the file.
- The "Back" button takes the admin back to the previous page (`SystemMenu`) and displays a message reminding them not to forget their password.

Overall, this code provides a GUI for displaying the appointment list and allows the admin to mark appointments as completed. It reads and updates data from a file to maintain the status of the appointments.

APPLICATION

Our vaccination management system application is designed for use within a university setting. And it primarily serves the needs of COVID-19 vaccinations. This application serves as a tool for managing and organizing the vaccination process for university members. Here are some potential use cases for this project:

1. **Appointment Management:** The application allows administrators to view a list of appointments, including details such as the name, age, NSUer status, NID number, phone number, first-time status, password, center name, appointment date, and vaccination status. This facilitates efficient appointment scheduling and tracking.

2. **Vaccination Status Tracking:** The application provides a way to track the vaccination status of each appointment. The "Complete Vaccination" button allows administrators to mark appointments as completed, updating the vaccination status accordingly.

3. **Data Persistence:** The application stores appointment data in a file (`registration.txt`). This ensures the data is persistent and can be accessed even if the application is restarted.

4. **User Authentication:** The application includes an admin login functionality to ensure only authorized personnel can access and manage the appointment data. The `AdminLogin` class handles the authentication process.

5. **User-friendly Interface:** The application utilizes Swing, a Java GUI toolkit, to create an intuitive and visually appealing interface. It uses colors, icons, buttons, and tables to enhance the user experience and provide a convenient way to interact with the system.

6. **Navigation:** The application offers a "Back" button, which allows administrators to return to the previous page (in this case, the system menu). This simplifies navigation and ensures a smooth user experience.

Overall, this vaccination management system aims to streamline the process of scheduling and tracking appointments, managing vaccination statuses, and providing a user-friendly interface for administrators within a university environment. It assists in efficiently managing the vaccination process and maintaining accurate records for effective communication and decision-making.

LIMITATIONS

While our project code is a foundation for a vaccination management system, it also has certain limitations. Here are some potential limitations of this project:

1. **Limited Functionality:** The code provided focuses primarily on appointment management and vaccination status tracking of the COVID-19 vaccination program. However, there may be other aspects to consider in a comprehensive vaccination management system, such as inventory management, reporting and analytics, user roles and permissions, and integration with external systems.

2. **Lack of Error Handling:** The code does not include robust error handling mechanisms. For example, exceptions may occur if there are issues with file operations (e.g., reading from or writing to `registration.txt`), but they are not handled gracefully. Proper error handling and user-friendly error messages should be implemented to enhance the reliability of the application.

3. **Security Considerations:** The current implementation lacks comprehensive security measures. While the `AdminLogin` class provides a basic authentication mechanism, it uses hard-coded credentials, which is not secure. In a production environment, stronger authentication methods (e.g., encrypted passwords) and authorization mechanisms should be implemented to protect sensitive data and prevent unauthorized access.

4. **Limited Data Validation:** The code does not perform extensive validation on the input data. For example, it does not check for data integrity, format validation, or handle potential input errors. Implementing proper data validation and error handling is essential to ensure the accuracy and integrity of the data stored and processed by the system.

5. **User Experience Improvements:** While the code includes basic user interface elements, there is room for enhancing the user experience. For example, providing feedback during long-running operations, implementing pagination or filtering options for the appointment list, and improving the visual design can make the application more user-friendly and efficient.

6. **Scalability & Performance:** The current implementation reads all appointment data from a text file (`registration.txt`) into memory at once. This approach may not scale well with a large number of appointments. A more efficient data storage and retrieval mechanism (such as a database) could improve performance and scalability.

7. **Maintenance & Extensibility:** The code lacks proper code organization and modularization, making it difficult to maintain and extend the application in the long run. Applying software design principles, such as encapsulation, separation of concerns, and modular architecture, can enhance code maintainability and extensibility.

These limitations highlight areas that require further attention and improvement to create a more robust, secure, and scalable vaccination management system.

FUTURE WORK

Our vaccination management system has potential future scopes for further development and enhancement. Here are some future scopes of work:

1. **Expanded Functionality:** The system can include additional features and modules to make it more comprehensive. Some possible enhancements could include adding new vaccination programs, inventory management to track vaccine availability, scheduling and reminders for follow-up doses, integration with external systems (e.g., electronic health records), and reporting and analytics to generate insights on vaccination trends and coverage.

2. **User Roles and Permissions:** Implementing a role-based access control system can allow different levels of access and permissions for various user types, such as administrators, healthcare professionals, and users. This would ensure that sensitive information is protected and users have appropriate access to system functionalities based on their roles.

3. **Automated Notifications:** Integrating with messaging services (e.g., email, SMS) can enable automated notifications and reminders for appointments, vaccination updates, and important announcements. This would help improve communication with users and ensure they stay informed about their vaccination status.

4. **Enhanced Security Measures:** Strengthening the system's security is crucial. Implementing secure password storage techniques (e.g., hashing and salting), using secure protocols for data transmission, and employing measures like two-factor authentication can enhance the overall security posture of the application.

5. **Data Validation and Error Handling:** Implementing robust data validation mechanisms can help ensure the integrity and consistency of the data stored in the system. This includes validating user input, handling errors gracefully, and implementing appropriate error messages and feedback to users.

6. **Improved User Experience:** Enhancements to the user interface and overall user experience can make the system more intuitive and user-friendly. This may involve refining the design, improving navigation, providing progress indicators during lengthy operations, and optimizing the application's overall performance.

7. **Scalability and Performance Optimization:** As the system grows and handles larger amounts of data, it becomes crucial to optimize its scalability and performance. This could involve implementing caching mechanisms, optimizing database queries, and considering distributed architectures to ensure the system can handle increased user loads effectively.

8. **Localization and Internationalization:** Adapting the system to support multiple languages and regional preferences can help cater to a broader user base. This includes providing language options, date, and time formats, and adhering to cultural and regional norms.

9. **Integration with Vaccine APIs:** Integrating the system with vaccine-related APIs, such as government databases or third-party vaccine providers, can enable real-time updates on vaccine availability, eligibility criteria, and other relevant information. This would help streamline appointment scheduling and provide users with up-to-date information.

10. **Continuous Maintenance and Bug Fixes:** Regular maintenance and bug fixing are essential to ensure the smooth functioning of the system. This includes addressing reported issues, monitoring system performance, and applying software updates and security patches.

These future scopes of work can enhance the vaccination management system's functionality, usability, security, and scalability, making it more comprehensive and capable of meeting evolving requirements and user needs.

THE SOURCE CODE

AbstractSystemMenu.java:

```
package backEnd;

import javax.swing.*;
import java.awt.*;

public abstract class AbstractSystemMenu {

        protected JFrame frame;

    public AbstractSystemMenu(String title) {
        frame = new JFrame(title);
        frame.setLayout(new GridLayout(3, 2));
        frame.setSize(1200, 800);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    protected void addButton(JButton button) {
        frame.add(button);
    }

    protected void showFrame() {
        frame.setVisible(true);
    }

    protected void disposeFrame() {
        frame.dispose();
    }
}
```

UserDataRetriever.java:

```
package backEnd;

public interface UserDataRetriever {
    String getUserInfo(String phoneNumber, String password);
}
```

UserRegistrationBase.java:

```
package backEnd;

import javax.swing.*;

import frontEnd.SystemMenu;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.File;
import java.io.IOException;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class UserRegistrationBase {
	// Declare components
    protected JFrame frame;
    protected JTextField nameTextField;
    protected JButton backButton;
    protected JTextField ageTextField;
    protected JTextField nsuTextField;
    protected JTextField nidTextField;
    protected JTextField phoneTextField;
    protected JTextField doseTextField;
    protected JPasswordField passwordTextField;
    protected JButton registerButton;
    protected JLabel appointmentDateLabel;
    protected JTextArea textArea1;
    protected JTextField centerTextField;
    protected JLabel emptyLabel;

    @SuppressWarnings("deprecation")
	protected int getNumberOfAppointments(Date date) {
        int count = 0;
        try {
            File file = new File("registration.txt");
            Scanner scanner = new Scanner(file);
```

```
            DateFormat dateFormat = new SimpleDateFormat("EEE MMM dd HH:mm:ss zzz yyyy");

            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                if (line.startsWith("Appointment Date: ")) {
                    String dateString = line.substring("Appointment Date: ".length());
                    Date appointmentDate = dateFormat.parse(dateString);
                    if (appointmentDate.getDate() == date.getDate() &&
                            appointmentDate.getMonth() == date.getMonth() &&
                            appointmentDate.getYear() == date.getYear()) {
                        count++;
                    }
                }
            }
            scanner.close();
        } catch (IOException | ParseException e) {
            e.printStackTrace();
        }
        return count;
    }

    protected void initializeUI() {
        frame = new JFrame("Registration Form");
        frame.setLayout(new GridLayout(0, 2, 0, 1));

        // Set the frame icon
        Image titleIcon = new
ImageIcon(this.getClass().getResource("/nsu-logo-small.png")).getImage();
        frame.setIconImage(titleIcon);

        // Define colors
        Color navyBlue = new Color(0, 43, 77);
        Color nsuGold = new Color(225,186,77);
        Color nsuGreen = new Color(56,132,56);

        // Initialize and configure text fields
        nameTextField = new JTextField();
        nameTextField.setPreferredSize(new Dimension(200,25));
        nameTextField.setAlignmentX(Component.CENTER_ALIGNMENT);

        // Create a panel for the "Name" field with a navy blue background
        JPanel name = new JPanel(new FlowLayout());
        name.add(new JLabel("Name:"));
        name.setBackground(navyBlue);
```

```
frame.add(name);
frame.add(nameTextField);

// Set the foreground (text) color of the "Name" label to white
JLabel nameLabel = (JLabel) name.getComponent(0);
nameLabel.setForeground(Color.WHITE);

// Initialize and configure text fields
ageTextField = new JTextField();
ageTextField.setPreferredSize(new Dimension(200,25));
ageTextField.setAlignmentX(Component.CENTER_ALIGNMENT);

// Create a panel for the "Age" field with a navy blue background
JPanel age = new JPanel(new FlowLayout());
age.add(new JLabel("Age:"));
age.setBackground(navyBlue);
frame.add(age);
frame.add(ageTextField);

// Set the foreground (text) color of the "Age" label to white
JLabel ageLabel = (JLabel) age.getComponent(0);
ageLabel.setForeground(Color.WHITE);

// Initialize and configure text fields
nsuTextField = new JTextField();
nsuTextField.setPreferredSize(new Dimension(200,25));
nsuTextField.setAlignmentX(Component.CENTER_ALIGNMENT);

// Create a panel for the "NSUer? Yes/No?" field with a navy blue background
JPanel nsu = new JPanel(new FlowLayout());
nsu.add(new JLabel("NSUer? Yes/No?:"));
nsu.setBackground(navyBlue);
frame.add(nsu);
frame.add(nsuTextField);

// Set the foreground (text) color of the "NSUer? Yes/No?" label to white
JLabel nsuLabel = (JLabel) nsu.getComponent(0);
nsuLabel.setForeground(Color.WHITE);

// Initialize and configure text fields
nidTextField = new JTextField();
nidTextField.setPreferredSize(new Dimension(200,25));
nidTextField.setAlignmentX(Component.CENTER_ALIGNMENT);
```

```
// Create a panel for the "NID Number" field with a navy blue background
JPanel nid = new JPanel(new FlowLayout());
nid.add(new JLabel("NID Number:"));
nid.setBackground(navyBlue);
frame.add(nid);
frame.add(nidTextField);

// Set the foreground (text) color of the "NID Number" label to white
JLabel nidLabel = (JLabel) nid.getComponent(0);
nidLabel.setForeground(Color.WHITE);

// Initialize and configure text fields
phoneTextField = new JTextField();
phoneTextField.setPreferredSize(new Dimension(200,25));
phoneTextField.setAlignmentX(Component.CENTER_ALIGNMENT);

// Create a panel for the "Phone Number" field with a navy blue background
JPanel phone = new JPanel(new FlowLayout());
phone.add(new JLabel("Phone Number:"));
phone.setBackground(navyBlue);
frame.add(phone);
frame.add(phoneTextField);

// Set the foreground (text) color of the "Phone Number" label to white
JLabel phoneLabel = (JLabel) phone.getComponent(0);
phoneLabel.setForeground(Color.WHITE);

// Initialize and configure text fields
doseTextField = new JTextField();
doseTextField.setPreferredSize(new Dimension(200,25));
doseTextField.setAlignmentX(Component.CENTER_ALIGNMENT);

// Create a panel for the "First Time? Yes/No?" field with a navy blue background
JPanel dose = new JPanel(new FlowLayout());
dose.add(new JLabel("First Time? Yes/No?:"));
dose.setBackground(navyBlue);
frame.add(dose);
frame.add(doseTextField);

// Set the foreground (text) color of the "First Time? Yes/No?" label to white
JLabel doseLabel = (JLabel) dose.getComponent(0);
doseLabel.setForeground(Color.WHITE);

// Initialize and configure text fields
```

```
        centerTextField = new JTextField();
        centerTextField.setPreferredSize(new Dimension(200,25));
        centerTextField.setAlignmentX(Component.CENTER_ALIGNMENT);

        // Create a panel for the "Center Name? NAC/SAC?" field with a navy blue background
        JPanel center = new JPanel(new FlowLayout());
        center.add(new JLabel("Center Name? NAC/SAC?:"));
        center.setBackground(navyBlue);
        frame.add(center);
        frame.add(centerTextField);

        // Set the foreground (text) color of the "Center Name? NAC/SAC?" label to white
        JLabel centerLabel = (JLabel) center.getComponent(0);
        centerLabel.setForeground(Color.WHITE);

        // Initialize and configure text fields
        passwordTextField = new JPasswordField();
        passwordTextField.setPreferredSize(new Dimension(200,25));
        passwordTextField.setAlignmentX(Component.CENTER_ALIGNMENT);

        // Create a panel for the "Password" field with a navy blue background
        JPanel password = new JPanel(new FlowLayout());
        password.add(new JLabel("Password:"));
        password.setBackground(navyBlue);
        frame.add(password);
        frame.add(passwordTextField);

        // Set the foreground (text) color of the "Password" label to white
        JLabel passwordLabel = (JLabel) password.getComponent(0);
        passwordLabel.setForeground(Color.WHITE);

        // Create a panel for the "Appointment Date" field with a navy blue background
        JPanel appointmentDatePanel = new JPanel(new FlowLayout());
        appointmentDatePanel.add(new JLabel("Appointment Date:"));
        appointmentDatePanel.setBackground(navyBlue);
        frame.add(appointmentDatePanel);

        // Set the foreground (text) color of the "Appointment Date" label to white
        JLabel appointmentDateLabel = (JLabel) appointmentDatePanel.getComponent(0);
        appointmentDateLabel.setForeground(Color.WHITE);

        // Create an empty label for spacing purposes and to display the generated appointment
date
        emptyLabel = new JLabel();
```

```
emptyLabel.setPreferredSize(new Dimension(200, 25));
emptyLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
emptyLabel.setBackground(Color.WHITE);
emptyLabel.setForeground(Color.BLACK);
frame.add(emptyLabel);

// Configure the back button
backButton = new JButton("Back");
backButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Close the current page and open the landing page.
        frame.setVisible(false);
        new SystemMenu();
    }
});
backButton.setPreferredSize(new Dimension(200, 25));
backButton.setBackground(Color.RED);
backButton.setForeground(Color.WHITE);
backButton.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseEntered(MouseEvent e) {
        backButton.setBackground(nsuGold);
    }

    @Override
    public void mouseExited(MouseEvent e) {
        backButton.setBackground(Color.RED);
    }
});
frame.add(backButton);

// Configure the register button
registerButton = new JButton("Register");
registerButton.setPreferredSize(new Dimension(200, 25));
registerButton.setBackground(nsuGreen);
registerButton.setForeground(Color.WHITE);
registerButton.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseEntered(MouseEvent e) {
        registerButton.setBackground(nsuGold);
    }

    @Override
```

```
            public void mouseExited(MouseEvent e) {
                registerButton.setBackground(nsuGreen);
            }
        });
        registerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Implementation in child class
            }
        });

        frame.add(registerButton);

        // Configure frame properties
        frame.setSize(1200, 800);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        // Center the frame on the screen
        frame.setLocationRelativeTo(null);

        // Align the text fields and buttons in the middle of the frame
        nameTextField.setAlignmentX(Component.CENTER_ALIGNMENT);
    }
}
```

SystemMenu.java:

```
package frontEnd;

import backEnd.AbstractSystemMenu;
import javax.swing.*;
import administrator.AdminLogin;
import java.awt.*;
import java.awt.event.*;

public class SystemMenu extends AbstractSystemMenu {

    // Define buttons
    private JButton registerButton;
    private JButton vaccineCardButton;
    private JButton vaccineCertificateButton;
    private JButton adminLoginButton;
```

```
    public SystemMenu() {
        super("NSU COVID-19 Vaccination Management System"); // Call the constructor of the
superclass

        // Set the frame icon
        Image titleIcon = new
ImageIcon(this.getClass().getResource("/nsu-logo-small.png")).getImage();
        frame.setIconImage(titleIcon);

        // Set the layout manager for the frame
        frame.setLayout(new GridLayout(3, 0, 0, 1));

        // Create a panel for the first row with a navy blue background
        JPanel firstRowPanel = new JPanel(new FlowLayout());
        Color navyBlue = new Color(0, 43, 77);
        firstRowPanel.setBackground(navyBlue);
        frame.add(firstRowPanel);

        // Create and add an image component to the first row panel
        JLabel imageLabel1 = new JLabel(new
ImageIcon(this.getClass().getResource("/phr-logo.png")));
        firstRowPanel.add(imageLabel1);

        // Create a panel for the second row with a navy blue background
        JPanel firstRowPanel1 = new JPanel(new FlowLayout());
        firstRowPanel1.setBackground(navyBlue);
        frame.add(firstRowPanel1);

        // Create and add an image component to the second row panel
        JLabel imageLabel2 = new JLabel(new
ImageIcon(this.getClass().getResource("/shls-logo.png")));
        firstRowPanel1.add(imageLabel2);

        Color nsuCyan = new Color(51,143,190);

        // Create and configure the "User Registration" button
        registerButton = new JButton("User Registration");
        registerButton.setBackground(navyBlue);
        registerButton.setForeground(Color.WHITE);
        registerButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                registerButton.setBackground(nsuCyan);
```

```
    }

    @Override
    public void mouseExited(MouseEvent e) {
        registerButton.setBackground(navyBlue);
    }
});
frame.add(registerButton);

// Create and configure the "User Info" button
vaccineCardButton = new JButton("User Info");
vaccineCardButton.setBackground(navyBlue);
vaccineCardButton.setForeground(Color.WHITE);
vaccineCardButton.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseEntered(MouseEvent e) {
        vaccineCardButton.setBackground(nsuCyan);
    }

    @Override
    public void mouseExited(MouseEvent e) {
        vaccineCardButton.setBackground(navyBlue);
    }
});
frame.add(vaccineCardButton);

// Create and configure the "Vaccine Certificate" button
vaccineCertificateButton = new JButton("Vaccine Certificate");
vaccineCertificateButton.setBackground(navyBlue);
vaccineCertificateButton.setForeground(Color.WHITE);
vaccineCertificateButton.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseEntered(MouseEvent e) {
        vaccineCertificateButton.setBackground(nsuCyan);
    }

    @Override
    public void mouseExited(MouseEvent e) {
        vaccineCertificateButton.setBackground(navyBlue);
    }
});
frame.add(vaccineCertificateButton);

Color nsuGold = new Color(225,186,77);
```

```
        Color nsuGreen = new Color(56,132,56);

        // Create and configure the "Admin Login" button
        adminLoginButton = new JButton("Admin Login");
        adminLoginButton.setBackground(nsuGold);
        adminLoginButton.setForeground(navyBlue);
        adminLoginButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                adminLoginButton.setBackground(nsuGreen);
            }

            @Override
            public void mouseExited(MouseEvent e) {
                adminLoginButton.setBackground(nsuGold);
            }
        });
        frame.add(adminLoginButton);

        // Add action listeners to the buttons

        registerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UserRegistration();
                disposeFrame();
            }
        });

        vaccineCardButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                UserInfo registrantInfo = new UserInfo();
                registrantInfo.setVisible(true);
                disposeFrame();
            }
        });

        vaccineCertificateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                VaccineCertificate vaccineCertificate = new VaccineCertificate(new
FileUserDataRetriever());
                vaccineCertificate.setVisible(true);
```

```
            disposeFrame();
        }
    });

    adminLoginButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            AdminLogin adminLoginPage = new AdminLogin();
            adminLoginPage.setVisible(true);
            disposeFrame();
        }
    });

    // Set the background color of the frame to the color of the first row panel
    frame.setBackground(navyBlue);

    showFrame(); // Display the frame
}

public static void main(String[] args) {
    new SystemMenu(); // Create an instance of the SystemMenu class
}
}
```

UserRegistration.java:

```
package frontEnd;

import java.awt.event.*;
import java.io.*;
import java.util.Date;

import backEnd.UserRegistrationBase;

class UserRegistration extends UserRegistrationBase {
    public UserRegistration() {
        initializeUI();

        registerButton.addActionListener(new ActionListener() {
            @SuppressWarnings("deprecation")
                            @Override
            public void actionPerformed(ActionEvent e) {
                String name = nameTextField.getText();
```

```
            int age = Integer.parseInt(ageTextField.getText());
            String nsu = String.valueOf(nsuTextField.getText());
            String nidNumber = nidTextField.getText();
            String phoneNumber = phoneTextField.getText();
            String dose = String.valueOf(doseTextField.getText());
            String password = String.valueOf(passwordTextField.getPassword());
            String centerName = String.valueOf(centerTextField.getText());

            // Generate the vaccine appointment date.

            if(age >= 18 && nsu.equalsIgnoreCase("Yes") && dose.equalsIgnoreCase("Yes") &&
(centerName.equalsIgnoreCase("NAC") || centerName.equalsIgnoreCase("SAC"))) {
                Date appointmentDate = new Date();
                appointmentDate.setDate(appointmentDate.getDate() + 10);

                // Check if there are already 20 appointments on the appointment date
                int numberOfAppointments = getNumberOfAppointments(appointmentDate);
                while (numberOfAppointments >= 20) {
                    appointmentDate.setDate(appointmentDate.getDate() + 1);
                    numberOfAppointments = getNumberOfAppointments(appointmentDate);
                }

                try {
                    File file = new File("registration.txt");
                    FileWriter fileWriter = new FileWriter(file, true);
                    fileWriter.write("Name: " + name + ";");
                    fileWriter.write("Age: " + age + ";");
                    fileWriter.write("NSUer? Yes/No?: " + nsu + ";");
                    fileWriter.write("NID Number: " + nidNumber + ";");
                    fileWriter.write("Phone Number: " + phoneNumber + ";");
                    fileWriter.write("First Time? Yes/No?: " + dose + ";");
                    fileWriter.write("Password: " + password + ";");
                    fileWriter.write("Center Name? NAC/SAC?: " + centerName + ";");
                    fileWriter.write("Appointment Date: " + appointmentDate +  ";");
                    fileWriter.write("Vaccination: " + "No" +  ";");
                    fileWriter.write("\n");
                    fileWriter.close();
                } catch (IOException ex) {
                    ex.printStackTrace();
                }

                // Update the appointment date label.
                emptyLabel.setText(" " + appointmentDate);
            }
```

```
            else {
                @SuppressWarnings("unused")
                                    Date appointmentDate = new Date();
                // Update the appointment date label.
                emptyLabel.setText("Sorry! You Are Not Eligible For This Vaccine.");
            }


            }
        });
    }

    public static void main(String[] args) {
        new UserRegistration();
    }
}
```

UserInfo.java:

```
package frontEnd;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.*;


public class UserInfo extends JFrame {
    /**
             *
             */
            private static final long serialVersionUID = 1L;
            private JTextField phoneNumberField;
    private JPasswordField passwordField;
    private JButton backButton;
    public UserInfo() {
        setTitle("User Info");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 200);
        setLayout(new BorderLayout());
```

```
        // Center the frame on the screen
        setLocationRelativeTo(null);

        Image titleIcon = new
ImageIcon(this.getClass().getResource("/nsu-logo-small.png")).getImage();
        setIconImage(titleIcon);

        Color navyBlue = new Color(0, 43, 77);
        Color nsuGold = new Color(225,186,77);
        Color nsuGreen = new Color(56,132,56);

        // Panel for user input
        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new GridLayout(3, 2));
        inputPanel.setBackground(navyBlue);
        inputPanel.add(new JLabel("Phone Number:"));
        JLabel input = (JLabel) inputPanel.getComponent(0);
        input.setForeground(Color.WHITE);
        phoneNumberField = new JTextField();
        inputPanel.add(phoneNumberField);
        inputPanel.add(new JLabel("Password:"));
        JLabel input1 = (JLabel) inputPanel.getComponent(2);
        input1.setForeground(Color.WHITE);
        passwordField = new JPasswordField();
        inputPanel.add(passwordField);
        backButton = new JButton("Back");
        backButton.setBackground(Color.RED);
        backButton.setForeground(Color.WHITE);
        backButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                backButton.setBackground(nsuGold);
            }

            @Override
            public void mouseExited(MouseEvent e) {
                backButton.setBackground(Color.RED);
            }
        });
        inputPanel.add(backButton);
        JButton searchButton = new JButton("Search");
        searchButton.setBackground(nsuGreen);
        searchButton.setForeground(Color.WHITE);
        searchButton.addMouseListener(new MouseAdapter() {
```

```
        @Override
        public void mouseEntered(MouseEvent e) {
            searchButton.setBackground(nsuGold);
        }

        @Override
        public void mouseExited(MouseEvent e) {
            searchButton.setBackground(nsuGreen);
        }
    });
    inputPanel.add(searchButton);

    add(inputPanel);

    searchButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String phoneNumber = phoneNumberField.getText();
            String password = new String(passwordField.getPassword());

            String userInfo = searchUserInfo(phoneNumber, password);
            JOptionPane.showMessageDialog(UserInfo.this, userInfo);
        }
    });
    backButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Close the current page and open the landing page.
                JOptionPane.showMessageDialog(UserInfo.this, "Do not forget your password!");
            new SystemMenu();
            dispose();
        }
    });
}

private String searchUserInfo(String phoneNumber, String password) {
    try (BufferedReader reader = new BufferedReader(new FileReader("registration.txt"))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] userData = line.split(";");

            String name = getValueByKey(userData, "Name");
            String age = getValueByKey(userData, "Age");
            String nsu = getValueByKey(userData, "NSUer? Yes/No?");
```

```
            String nidNumber = getValueByKey(userData, "NID Number");
            String filePhoneNumber = getValueByKey(userData, "Phone Number");
            String dose = getValueByKey(userData, "First Time? Yes/No?");
            String filePassword = getValueByKey(userData, "Password");
            String centerName = getValueByKey(userData,"Center Name? NAC/SAC?");
            String appointmentDate = getValueByKey(userData, "Appointment Date");
            String vaccination = getValueByKey(userData, "Vaccination");

            if (phoneNumber.equals(filePhoneNumber) && password.equals(filePassword)) {
                StringBuilder userInfoBuilder = new StringBuilder();
                userInfoBuilder.append("                User Information            \n");
                userInfoBuilder.append("Name: ").append(name).append("\n");
                userInfoBuilder.append("Age: ").append(age).append("\n");
                userInfoBuilder.append("NSUer?: ").append(nsu).append("\n");
                userInfoBuilder.append("NID Number: ").append(nidNumber).append("\n");
                userInfoBuilder.append("Phone Number:
").append(filePhoneNumber).append("\n");
                userInfoBuilder.append("First Time?: ").append(dose).append("\n");
                userInfoBuilder.append("Center Name: ").append(centerName).append("\n");
                userInfoBuilder.append("Appointment Date:
").append(appointmentDate).append("\n");
                userInfoBuilder.append("Vaccination: ").append(vaccination).append("\n");

                return userInfoBuilder.toString();
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    return "User not found or incorrect credentials.";
  }


  private String getValueByKey(String[] userData, String key) {
    for (String data : userData) {
        String[] parts = data.split(": ");
        if (parts.length == 2 && parts[0].trim().equals(key)) {
            return parts[1].trim();
        }
    }
    return "";
  }
```

```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                UserInfo app = new UserInfo();
                app.setVisible(true);
            }

        });
}
}
```

VaccineCertificate.java:

```
package frontEnd;

import javax.swing.*;

import backEnd.UserDataRetriever;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.*;

class FileUserDataRetriever implements UserDataRetriever {
    @Override
    public String getUserInfo(String phoneNumber, String password) {
        try (BufferedReader reader = new BufferedReader(new FileReader("registration.txt"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] userData = line.split(";");

                String name = getValueByKey(userData, "Name");
                String age = getValueByKey(userData, "Age");
                String nsu = getValueByKey(userData, "NSUer? Yes/No?");
                String nidNumber = getValueByKey(userData, "NID Number");
                String filePhoneNumber = getValueByKey(userData, "Phone Number");
```

```
            String dose = getValueByKey(userData, "First Time? Yes/No?");
            String filePassword = getValueByKey(userData, "Password");
            String centerName = getValueByKey(userData,"Center Name? NAC/SAC?");
            String appointmentDate = getValueByKey(userData, "Appointment Date");
            String vaccination = getValueByKey(userData, "Vaccination");


            if (phoneNumber.equals(filePhoneNumber) && password.equals(filePassword)) {
                if(vaccination.equalsIgnoreCase("Yes")) {
                    StringBuilder userInfoBuilder = new StringBuilder();
                    userInfoBuilder.append("                    COVID-19 Vaccination Certificate
\n");
                    userInfoBuilder.append("Name: ").append(name).append("\n");
                    userInfoBuilder.append("Age: ").append(age).append("\n");
                    userInfoBuilder.append("NSUer? Yes/No?: ").append(nsu).append("\n");
                    userInfoBuilder.append("NID Number: ").append(nidNumber).append("\n");
                    userInfoBuilder.append("Phone Number:
").append(filePhoneNumber).append("\n");
                    userInfoBuilder.append("First Time? Yes/No?: ").append(dose).append("\n");
                    userInfoBuilder.append("Center Name? NAC/SAC?:
").append(centerName).append("\n");
                    userInfoBuilder.append("Appointment Date:
").append(appointmentDate).append("\n");
                    userInfoBuilder.append("Vaccination: ").append(vaccination).append("\n");

                    return userInfoBuilder.toString();
                }
                else {
                    return "negative";
                }
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    return "";
}

private String getValueByKey(String[] userData, String key) {
    for (String data : userData) {
        String[] parts = data.split(": ");
        if (parts.length == 2 && parts[0].trim().equals(key)) {
            return parts[1].trim();
```

```
            }
        }
        return "";
    }
}

public class VaccineCertificate extends JFrame {
    /**
             *
             */
            private static final long serialVersionUID = 1L;
            private JTextField phoneNumberField;
    private JPasswordField passwordField;
    private JTextArea certificateArea;
    private JButton generateButton;
    private JButton backButton;

    public VaccineCertificate(UserDataRetriever userDataRetriever) {
        setTitle("Vaccine Certificate");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 200);
        setLayout(new BorderLayout());
        // Center the frame on the screen
        setLocationRelativeTo(null);

        Image titleIcon = new
ImageIcon(this.getClass().getResource("/nsu-logo-small.png")).getImage();
        setIconImage(titleIcon);

        Color navyBlue = new Color(0, 43, 77);
        Color nsuGold = new Color(225,186,77);
        Color nsuGreen = new Color(56,132,56);

        // Panel for user input
        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new GridLayout(3, 2));
        inputPanel.setBackground(navyBlue);
        inputPanel.add(new JLabel("Phone Number:"));
        JLabel input = (JLabel) inputPanel.getComponent(0);
        input.setForeground(Color.WHITE);
        phoneNumberField = new JTextField();
        inputPanel.add(phoneNumberField);
        inputPanel.add(new JLabel("Password:"));
        JLabel input1 = (JLabel) inputPanel.getComponent(2);
```

```
        input1.setForeground(Color.WHITE);
        passwordField = new JPasswordField();
        inputPanel.add(passwordField);

        backButton = new JButton("Back");
        backButton.setBackground(Color.RED);
        backButton.setForeground(Color.WHITE);
        inputPanel.add(backButton);
        backButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                backButton.setBackground(nsuGold);
            }

            @Override
            public void mouseExited(MouseEvent e) {
                backButton.setBackground(Color.RED);
            }
        });
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Close the current page and open the landing page.
                    JOptionPane.showMessageDialog(VaccineCertificate.this, "Do not forget your
password!");
                new SystemMenu();
                dispose();
            }
        });

        generateButton = new JButton("Generate Certificate");
        generateButton.setBackground(nsuGreen);
        generateButton.setForeground(Color.WHITE);
        inputPanel.add(generateButton);
        generateButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                generateButton.setBackground(nsuGold);
            }

            @Override
            public void mouseExited(MouseEvent e) {
                generateButton.setBackground(nsuGreen);
            }
```

```
        });
        generateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                    String phoneNumber = phoneNumberField.getText();
                String password = new String(passwordField.getPassword());
                    String certificateInfo = showInfo(phoneNumber, password, userDataRetriever);

                    if(!certificateInfo.equals("Ineligible")) {
                            String certificateFileName = generateCertificate(certificateInfo);
                    JOptionPane.showMessageDialog(VaccineCertificate.this, "Certificate generated
successfully.\nFilename: " + certificateFileName);
                    new SystemMenu();
                    dispose();
                    }
                else {
                    JOptionPane.showMessageDialog(certificateArea, "You are not eligible for the
certificate.");
                    new SystemMenu();
                    dispose();
                }
            }
        });
        add(inputPanel);
    }

    private String showInfo(String phoneNumber, String password, UserDataRetriever
userDataRetriever) {
        // Retrieve user information based on phone number and password
        String userInfo = userDataRetriever.getUserInfo(phoneNumber, password);

        if (!userInfo.isEmpty() && !userInfo.equals("negative")) {
            StringBuilder certificateBuilder = new StringBuilder();
            certificateBuilder.append(userInfo);
            return certificateBuilder.toString();
        }

        return  "Ineligible";
    }

    private String generateCertificate(String certificateInfo) {
        String name = getValueByKey(certificateInfo.split("\n"), "Name");
        String certificateFileName = name + "_Certificate.txt";
```

```
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(certificateFileName))) {
            writer.write(certificateInfo);
        } catch (IOException e) {
            e.printStackTrace();
        }

        return certificateFileName;
    }

    private String getValueByKey(String[] userData, String key) {
        for (String data : userData) {
            String[] parts = data.split(": ");
            if (parts.length == 2 && parts[0].trim().equals(key)) {
                return parts[1].trim();
            }
        }
        return "";
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                UserDataRetriever userDataRetriever = new FileUserDataRetriever();
                VaccineCertificate app = new VaccineCertificate(userDataRetriever);
                app.setVisible(true);
            }
        });
    }
}
```

AdminLogin.java:

```
package administrator;

import javax.swing.*;
import frontEnd.SystemMenu;
import java.awt.*;
import java.awt.event.*;

public class AdminLogin extends JFrame {
    /**
          *
```

```
         */
        private static final long serialVersionUID = 1L;
        private JTextField usernameField;
    private JPasswordField passwordField;

    public AdminLogin() {

        Color navyBlue = new Color(0, 43, 77);
      Color nsuGold = new Color(225,186,77);
      Color nsuGreen = new Color(56,132,56);

      setTitle("Admin Login");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setSize(400, 200);
      // Center the frame on the screen
      setLocationRelativeTo(null);

      Image titleIcon = new
ImageIcon(this.getClass().getResource("/nsu-logo-small.png")).getImage();
      setIconImage(titleIcon);

      // Panel for user input
      JPanel inputPanel = new JPanel();
      inputPanel.setLayout(new GridLayout(3, 2));
      inputPanel.setBackground(navyBlue);
      inputPanel.add(new JLabel("Username:"));
      JLabel input = (JLabel) inputPanel.getComponent(0);
      input.setForeground(Color.WHITE);
      usernameField = new JTextField();
      inputPanel.add(usernameField);
      inputPanel.add(new JLabel("Password:"));
      JLabel input1 = (JLabel) inputPanel.getComponent(2);
      input1.setForeground(Color.WHITE);
      passwordField = new JPasswordField();
      inputPanel.add(passwordField);

      JButton backButton = new JButton("Back");
      backButton.setBackground(Color.RED);
      backButton.setForeground(Color.WHITE);
      backButton.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseEntered(MouseEvent e) {
          backButton.setBackground(nsuGold);
        }
```

```
            @Override
            public void mouseExited(MouseEvent e) {
                backButton.setBackground(Color.RED);
            }
        });
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Close the current page and open the landing page.
                    JOptionPane.showMessageDialog(AdminLogin.this, "Do not forget your
username and password!");
                new SystemMenu();
                dispose();
            }
        });
        inputPanel.add(backButton);

        JButton loginButton = new JButton("Login");
        loginButton.setBackground(nsuGreen);
        loginButton.setForeground(Color.WHITE);
        loginButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                loginButton.setBackground(nsuGold);
            }

            @Override
            public void mouseExited(MouseEvent e) {
                loginButton.setBackground(nsuGreen);
            }
        });
        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());

                if (authenticate(username, password)) {
                    JOptionPane.showMessageDialog(AdminLogin.this, "Login successful!");
                    new AppointmentList();
                    dispose();
                } else {
                    // Login failed, display error message
```

```
                JOptionPane.showMessageDialog(AdminLogin.this, "Invalid credentials. Please try
again.");
            }
        }
    });
    inputPanel.add(loginButton);

    add(inputPanel);
  }

  private boolean authenticate(String username, String password) {
    // Perform authentication logic here

    String adminUsername = "admin";
    String adminPassword = "admin123";

    return username.equals(adminUsername) && password.equals(adminPassword);
  }

  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
      @Override
      public void run() {
        AdminLogin adminLoginPage = new AdminLogin();
        adminLoginPage.setVisible(true);
      }
    });
  }
}
```

AppointmentList.java:

```
package administrator;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import frontEnd.SystemMenu;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.*;
```

```
class AppointmentList extends JFrame {

        private static final long serialVersionUID = 1L;
        private JTable appointmentTable;
    private DefaultTableModel tableModel;

    public AppointmentList() {

        Color navyBlue = new Color(0, 43, 77);
       Color nsuGold = new Color(225,186,77);
       Color nsuGreen = new Color(56,132,56);

       setTitle("Appointment List");
       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       setSize(800, 600);
       // Center the frame on the screen
       setLocationRelativeTo(null);

       Image titleIcon = new
ImageIcon(this.getClass().getResource("/nsu-logo-small.png")).getImage();
       setIconImage(titleIcon);

       String[] columnNames = {"Name", "Age", "NSUer?", "NID Number", "Phone Number", "First
Time?", "Password", "Center Name", "Appointment Date", "Vaccination"};
       String[][] appointmentData = getAppointmentData("registration.txt");
       tableModel = new DefaultTableModel(appointmentData, columnNames);
       appointmentTable = new JTable(tableModel);
       appointmentTable.setBackground(navyBlue);
       appointmentTable.setForeground(Color.WHITE);

       JScrollPane scrollPane = new JScrollPane(appointmentTable);
       add(scrollPane, BorderLayout.CENTER);

       JPanel adminPanel = new JPanel();
       adminPanel.setLayout(new GridLayout(2, 1));
       JButton completeButton = new JButton("Complete Vaccination");
       completeButton.setBackground(nsuGreen);
       completeButton.setForeground(Color.WHITE);
       completeButton.addMouseListener(new MouseAdapter() {
          @Override
          public void mouseEntered(MouseEvent e) {
             completeButton.setBackground(nsuGold);
          }
```

```
        @Override
        public void mouseExited(MouseEvent e) {
            completeButton.setBackground(nsuGreen);
        }
    });
    adminPanel.add(completeButton);

    JButton backButton = new JButton("Back");
    backButton.setBackground(Color.RED);
    backButton.setForeground(Color.WHITE);
    backButton.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseEntered(MouseEvent e) {
            backButton.setBackground(nsuGold);
        }

        @Override
        public void mouseExited(MouseEvent e) {
            backButton.setBackground(Color.RED);
        }
    });
    adminPanel.add(backButton);

    completeButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            int selectedRow = appointmentTable.getSelectedRow();
            if (selectedRow != -1) {
                String vaccinationStatus = tableModel.getValueAt(selectedRow, 9).toString();
                if (vaccinationStatus.equals("No")) {
                    tableModel.setValueAt("Yes", selectedRow, 9);
                    updateVaccinationStatus(selectedRow);
                }
            }
        }
    });

    backButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Close the current page and open the landing page.
                JOptionPane.showMessageDialog(AppointmentList.this, "Do not forget your
password!");
```

```
            new SystemMenu();
            dispose();
        }
    });

    add(adminPanel, BorderLayout.SOUTH);

    setVisible(true);
}

private String[][] getAppointmentData(String filePath) {
    try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
        String line;
        StringBuilder dataBuilder = new StringBuilder();
        while ((line = reader.readLine()) != null) {
            dataBuilder.append(line).append("\n");
        }

        String[] appointments = dataBuilder.toString().split("\n");
        String[][] appointmentData = new String[appointments.length][];
        for (int i = 0; i < appointments.length; i++) {
            String[] appointmentFields = appointments[i].split(";");
            appointmentData[i] = new String[appointmentFields.length];
            for (int j = 0; j < appointmentFields.length; j++) {
                appointmentData[i][j] = getValueByKey(appointmentFields[j].trim().split(": "), 1);
            }
        }
        return appointmentData;
    } catch (IOException e) {
        e.printStackTrace();
    }
    return new String[0][0];
}

private String getValueByKey(String[] data, int index) {
    if (index < 0 || index >= data.length) {
        return "";
    }
    return data[index].trim();
}

private void updateVaccinationStatus(int rowIndex) {
    try (BufferedReader reader = new BufferedReader(new FileReader("registration.txt"))) {
        StringBuilder fileData = new StringBuilder();
```

```
            String line;
            int currentRow = 0;
            while ((line = reader.readLine()) != null) {
                if (currentRow == rowIndex) {
                    line = line.replace("Vaccination: No", "Vaccination: Yes");
                }
                fileData.append(line).append("\n");
                currentRow++;
            }

            try (BufferedWriter writer = new BufferedWriter(new FileWriter("registration.txt"))) {
                writer.write(fileData.toString());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                @SuppressWarnings("unused")
                                    AppointmentList appointmentListPage = new AppointmentList();
            }
        });
    }
}
```