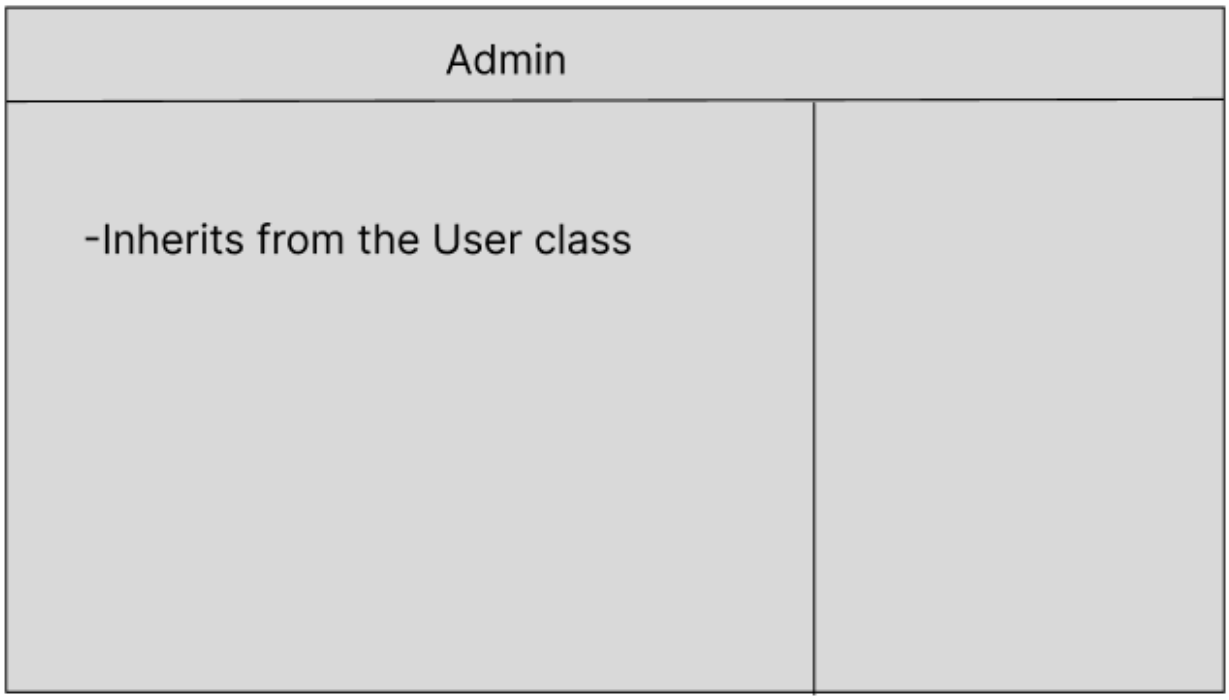
Functional Specification:

When the application begins, the user will see a login page. Depending on if the user is a nurse or an administrator, the page will be different.

For an administrator, when the site opens up, there will be a couple different options. First, the administrator should be able to put new hospital availability. They should be able to put the hospital that needs a nurse, shift type, and amount of nurses needed. The second thing the admin should do is make an account for new nurses. So when a nurse passes the interview process and is hired, the administrator uses the nurse's email and makes them an account with a temporary password. The final thing an administrator should be able to do is view all the nurses that are scheduled and what shifts are still available. This would have a filtering mechanism based on hospital, date and shift type.

If a nurse logs in, they will have the option to do a couple things. They will be able to look for needed shifts by filtering. The filtering mechanism will filter through date, hospital distance, hospital name and shift type. The nurse will then be able to schedule themselves. Nurses will not be able to schedule a shift that has time conflicts with an already scheduled shift. Another thing a nurse can do is see their upcoming schedule and cancel if need be. They can also see their past shifts. Finally, a nurse should be able to change their password.

CRC CARDS



Nurse	
- Inherits from the User class - holds the the list of shifts that are reserved - reserves and cancel the shifts - changes its password - Serializes itself	- Shift - AppSystem

User	
-super class for Nurse and Admin - stores and provides accessors for the users ID, name and password	

ViewManager	
- Stores and provides accessors to all the frames - sets the visibility for all frames - makes a new instance of each frame to be shown - attaches action listeners to buttons	- LoginFrame - DashboardFrame - NDashboardFrame - ADashboardFrame - NShiftsFrame - AShiftsFrame - NursesFrame - ScheduleFrame - AddNurseFrame - ShiftCreationFrame -ChangePasswordFrame

AppSystem	
- handles deserialization of nurses and shifts - checks login credentials - stores the current user - verifies old password before changing a nurse password - adds and deletes a nurse object - adds and deletes a nurse file	- Nurse - Shift

ShiftManager	
- Creates a shift and adds it to available shifts - Deletes a shift from available shifts - Reserves a shift for a nurse object - Cancels a shift for a nurse object	-Shift -AppSystem

Shift	
-Stores and provides accessors to the ID, type, hospital and date of a Shift - serializes itself	

ConcreteNurseIterator	
- Implements the NurseIterator which will be used for iterating through the Nurse objects	- Nurse

ConcreteShiftIterator	
- Implements the ShiftIterator which will be used for iterating through the Shift objects	- Shift

Sort	
- Class that extends the JPanel class and sorts an array list of shifts by date, hospital or type depending on the radio button that is selected - provides accessors to the radio buttons - provides new comparators objects depending on the type of sort needed	- Shift

Filter	
- Class that extends the JPanel class and filters an array list of shifts by hospital or type depending on the radio button that is selected - provides accessors to the radio buttons - provides new array list of the filtered list	- Shift

NurseCollection	
- implements a NurseIterator and implements the create iterator function by overriding the aggregator	-Nurse

ShiftCollection	
- implements a ShiftIterator and implements the create iterator function by overriding the aggregator	- Shift

NurseApp	
- runs main method and allows the view manager to make a new login frame	- AppSystem - ViewManager

DashboardFrame	
- abstract class for NDashboard and ADashboardFrame. - provides a logout button, a welcome message and a content panel that the menu buttons would be added to - provides an accessor method to the frame which	-Nurse -Admin -Shift -Filter -Sort -ShiftManager -VIewManager

ADashboardFrame	
- extends the DashboardFrame class - specifies and adds the menu buttons for the admin user - allows view manager to attach action listeners in its constructor	-ViewManager -AppSystem

NDashboardFrame	
- extends the DashboardFrame class - specifies and adds the menu buttons for the nurse user - allows view manager to attach action listeners in its constructor	-ViewManager -AppSystem

AShiftsFrame	
- holds the frame and the components that will be used to show the available shifts to the admin - enables the admin to delete any one of the shifts - allows view manager to attach action listeners in its constructor	-Shift -Filter -Sort -AppSystem -VlewManager

NShIftsFrame	
- holds the frame and the components that will be used to show the avaialble shifts to the nurse - enables a nurse to reserve any one of the shifts - allows view manager to attach action listeners in its constructor	-Shift -Filter -Sort -AppSystem -VlewManager

LoginFrame	
- allows a user to login - provides an accessor to the frame - allows view manager to attach action listeners in its constructor	- AppSystem - ViewManager

NursesFrame	
- Displays a list of all nurses to the admin - provides an accessor to the frame - allows view manager to attach action listeners in its constructor	-Nurse -AppSystem -VIewManager

AddNurseFrame	
- allows the admin to add a new nurse object by entering the ID, name and password - provides an accessor to the frame - allows view manager to attach action listeners in its constructor	-Nurse -AppSystem -VlewManager

ShiftCreationFrame	
- allows the admin to create a new shift by entering ID, hospital, type and date of the shift - provides an accessor to the frame - allows view manager to attach action listeners in its constructor	-Shift - AppSystem -ShiftManager -VlewManager

ChangePasswordFrame	
- allows a nurse to change their password by entering their old and new password - provides an accessor to the frame - allows view manager to attach action listeners in its constructor - set visibility for each frame	- AppSystem - ViewManager

ScheduleFrame	
- holds the frame and the components that will be used to show the shifts within the nurse's scedule - provides an accessor to the frame - allows view manager to attach action listeners in its constructor	-Shift - ShiftManager -Sort -AppSystem -VlewManager

DETAILED USE CASES

Login

ViewManager sets the LoginFrame visibility to true on start up

User enters ID and password in the JtextFields

User clicks the login button

The actionListener passes the texts from the JTextFields to the system

System verifies credentials.

If Credentials are correct:

User is assigned as the current user

Viewmanager is called to set the LoginFrame visibility to false

Viewmanager is called to set NDashboardFrame or ADashboardFrame visibility to true

If Credentials are incorrect:

Viewmanager is called to set an error message visibility to true

Change Password (pending)

Nurse selects the change password button on the dashboard

The actionListener notifies the ViewManager to make a new instance of the ChangePasswordFrame

The actionListener notifies the ViewManager to set the ChangePasswordFrame visibility to true

Nurse enters old and new password in the JtextFields

Nurse selects the changePassword Jbutton

The actionListener passes the texts from the JTextFields to the system which verifies the validity of the old password.

If Credentials are correct:

Password is changed in the Nurse object.

The viewmanager outputs a success message.

If Credentials are incorrect:

Viewmanager is called to set an error message visibility to true

Nurse view available shifts

Nurse selects the search button from dashboard

The actionListener notifies the ViewManager to set the AShiftsFrame visibility to true and calls the ShiftManager to viewAvailableShifts().

Iterator looks through each shift from array of shift objects and draws it on a Jlist element.

Lists will be redrawn depending on filters and sorting.

Admin view available shifts

Admin selects the search button from the dashboard. (NDashboardFrame).

The actionListener notifies the ViewManager to set the AShiftsFrame visibility to true and calls the ShiftManager to viewAvailableShifts().

-Iterator looks through each shift from array of shift objects and draws it on a Jlist element.

Lists will be redrawn depending on filters and sorting.

Reserve a shift

Nurse selects the reserve button on a specific shift from the available shifts page.

The actionListener calls the shiftmanager which calls the reserveShift() method.

The viewManager is then called to redraw the page.

Cancel reserved shift

Nurse selects the view schedule button from dashboard

The actionListener notifies the ViewManager to set the ScheduleFrame visibility to true

Iterator goes through each shift and draws a JList element for each with its contents

The nurse presses the cancel button beside the shift in which they require it to be canceled.

The actionListener notifies the ShiftManager which cancels reservation.

The viewManager is then called to redraw the frame.

Create a shift

Admin selects the add availability button from the dashboard

ViewManager sets the ShiftCreationFrame visibility to true

Admin enters date, shift type and hospital and pressed the create button.

The actionListener calls the ShiftManager createShift method.

ViewManager returns a successful message and resets the page.

Delete a shift

Admin selects the delete button on a specific shift from the available shifts page.

The actionListener calls the shift manager which calls the deleteAvaiShift() method.

The viewManager is then called to redraw the AShiftsFrame.

Admin creates new nurse account

Admin selects the add a new nurse button from the dashboard.

ViewManager sets the AddNurseFrame visibility to true

Admin inputs nurse information including: name, ID

The admin clicks the create button

The actionListener calls the addNurse() method from the system.

ViewManager returns a successful message and resets the page.

Admin view nurses

Admin selects the view nurses button from the dashboard.

The actionListener notifies the ViewManager to set the NursesFrame visibility to true

Iterator goes through each nurse and ViewManager draws their details onto NursesFrame

Admin deletes a nurse

Admin selects the delete button on a specific nurse on NursesFrame

The actionListener notifies the system to delete the nurse object

Logging out

User goes to the dashboard and picks to log out.

The system puts current user id to NULL.

ViewManager sets the LoginFrame visibility to true

Returning Home

The user presses the home icon on the page.

The viewManager puts the current frame visibility to false and the dashboard visibility to true.

Filter

User selects the required checkboxes (hospital, shiftType)

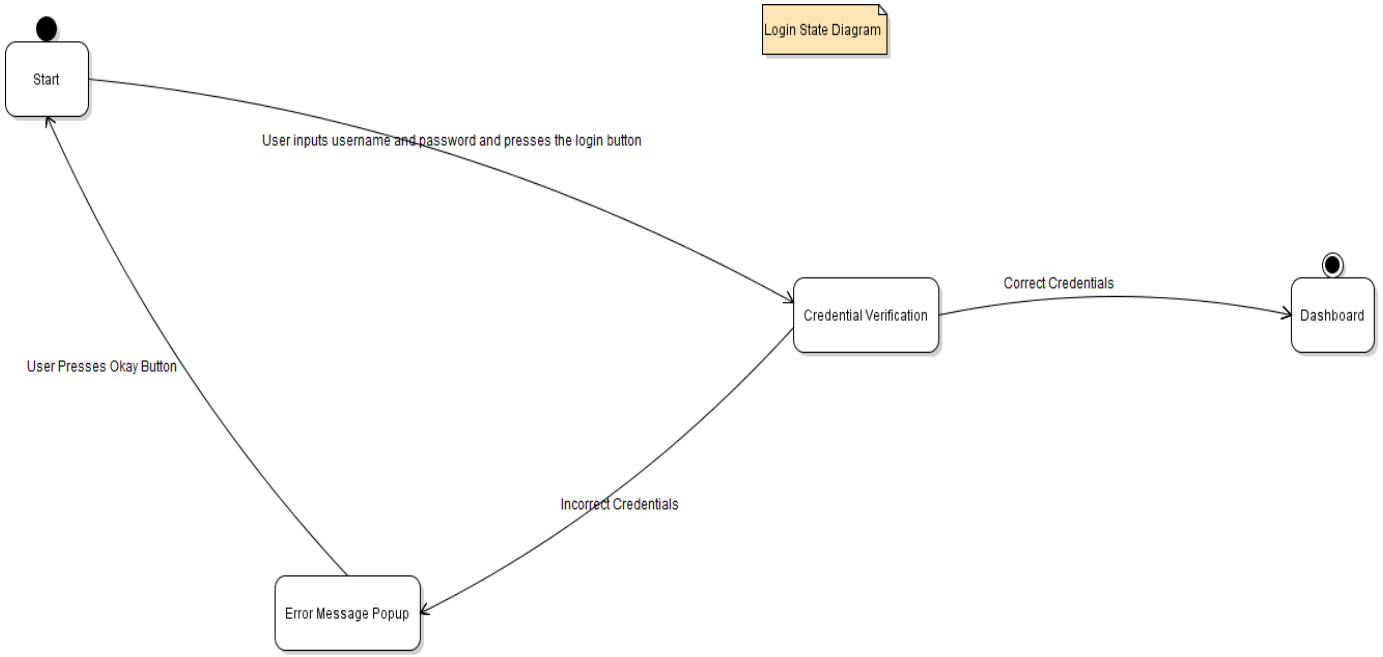
User selects the Filter button

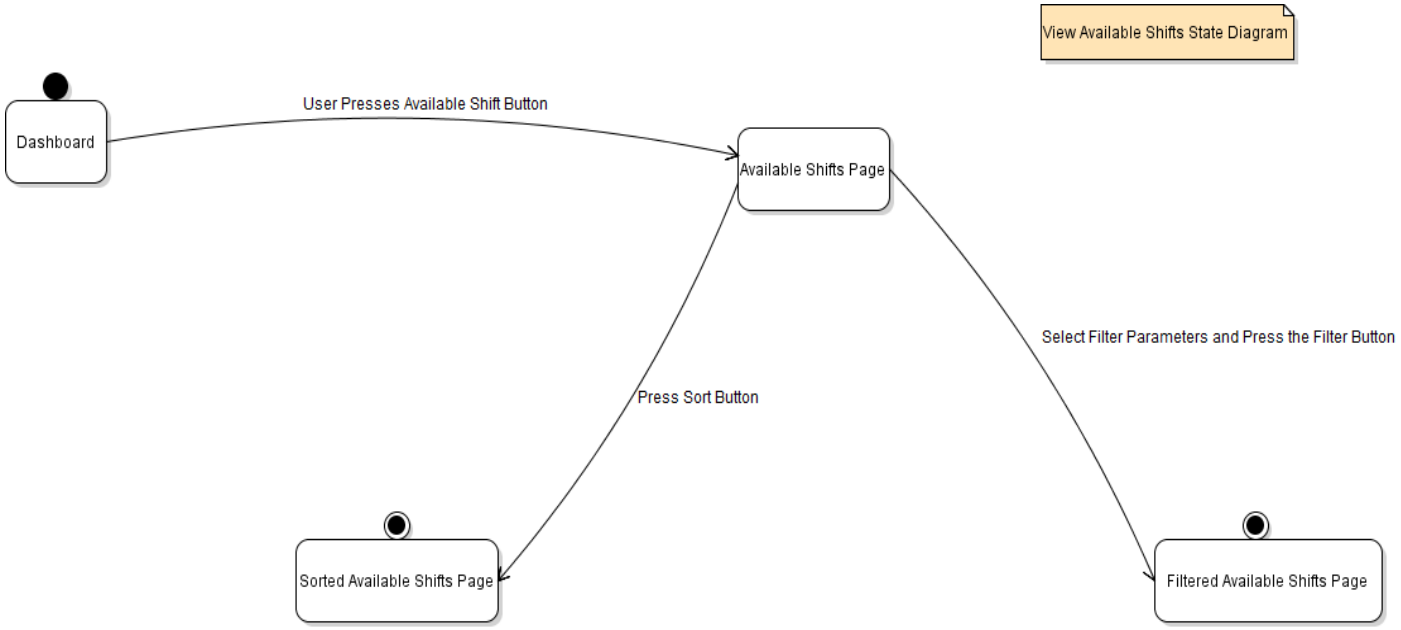
ActionListener notifies system to remove the shifts that are filtered out

Shift manager returns a list of the new shifts

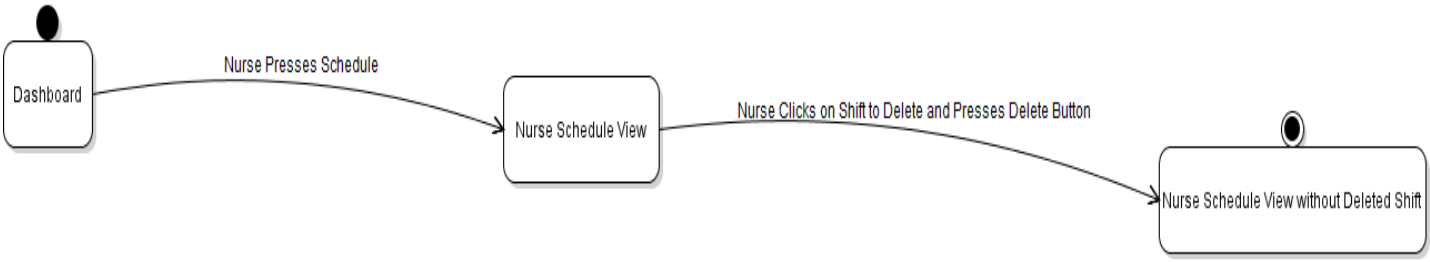
viewManager to redraw shifts

STATE DIAGRAMS

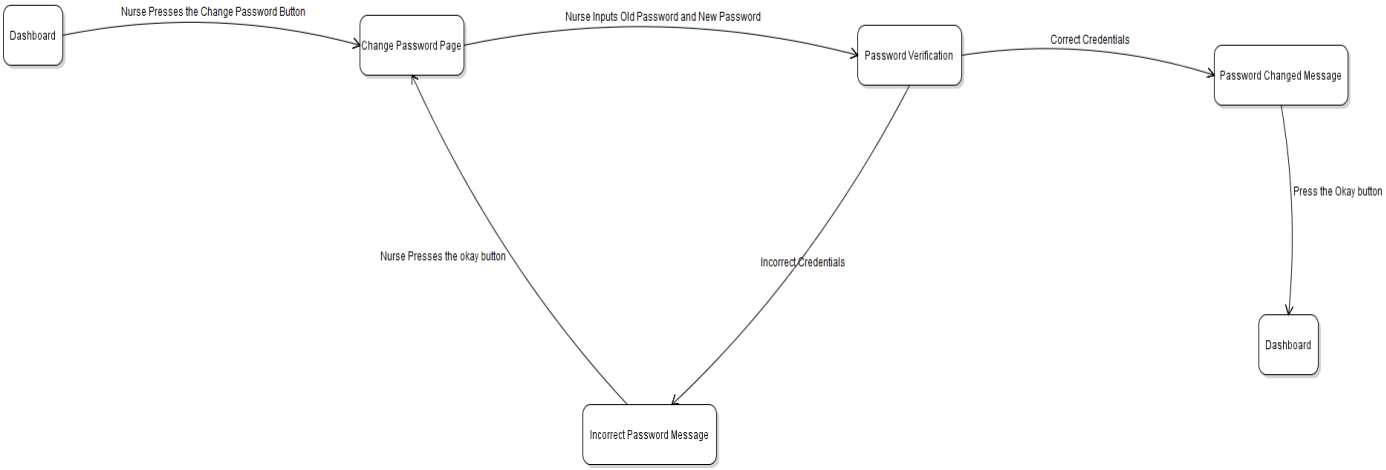




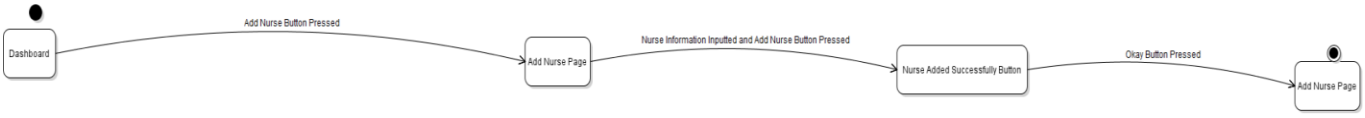
Cancel Shift State Diagram



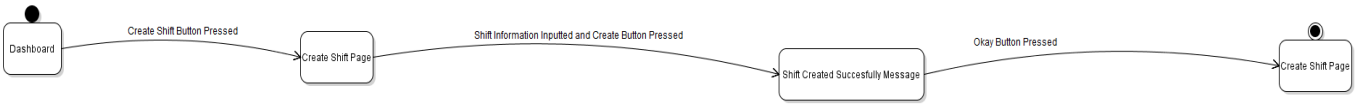
Change Password State Diagram

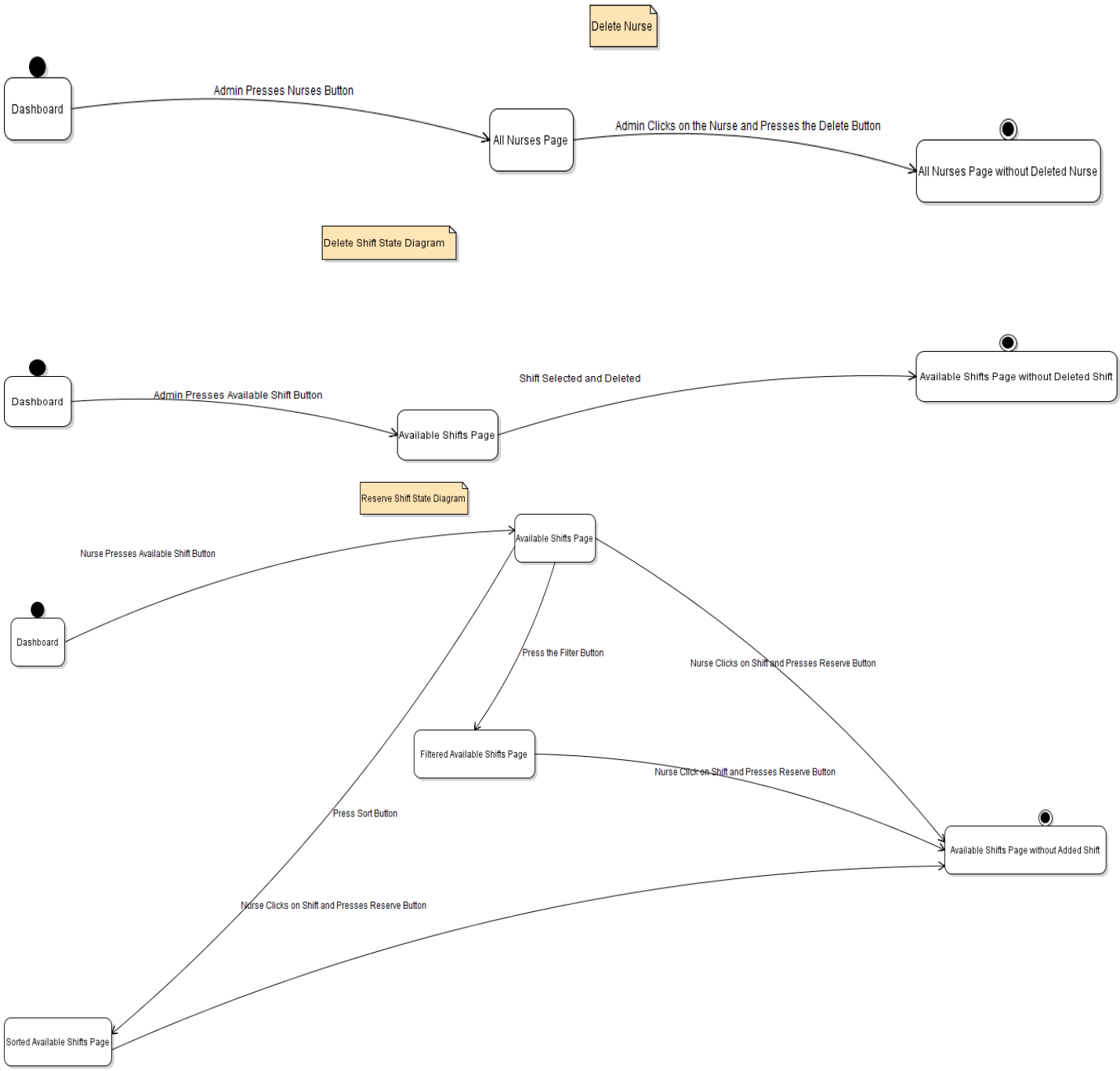


Create New Nurse State Diagram

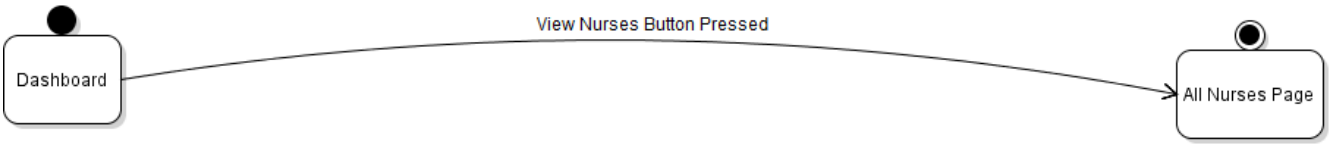


Create Shift State Diagram

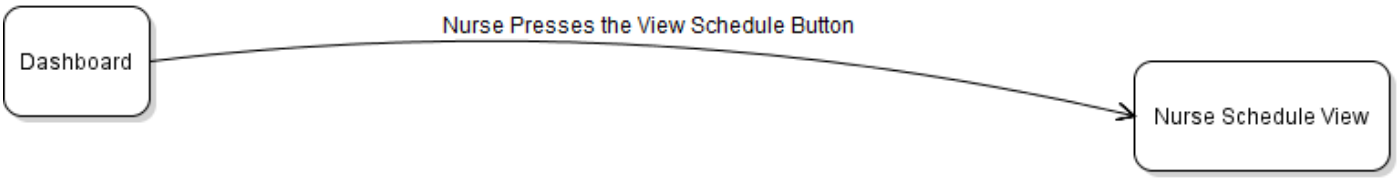



Delete Nurse
Dashboard
Admin Presses Nurses Button
All Nurses Page
Admin Clicks on the Nurse and Presses the Delete Button
All Nurses Page without Deleted Nurse
Delete Shift State Diagram
Dashboard
Admin Presses Available Shift Button
Available Shifts Page
Shift Selected and Deleted
Available Shifts Page without Deleted Shift
Reserve Shift State Diagram
Available Shifts Page
Nurse Presses Available Shift Button
Dashboard
Press the Filter Button
Nurse Clicks on Shift and Presses Reserve Button
Filtered Available Shifts Page
Nurse Click on Shift and Presses Reserve Button
Press Sort Button
Available Shifts Page without Added Shift
Nurse Clicks on Shift and Presses Reserve Button
Sorted Available Shifts Page

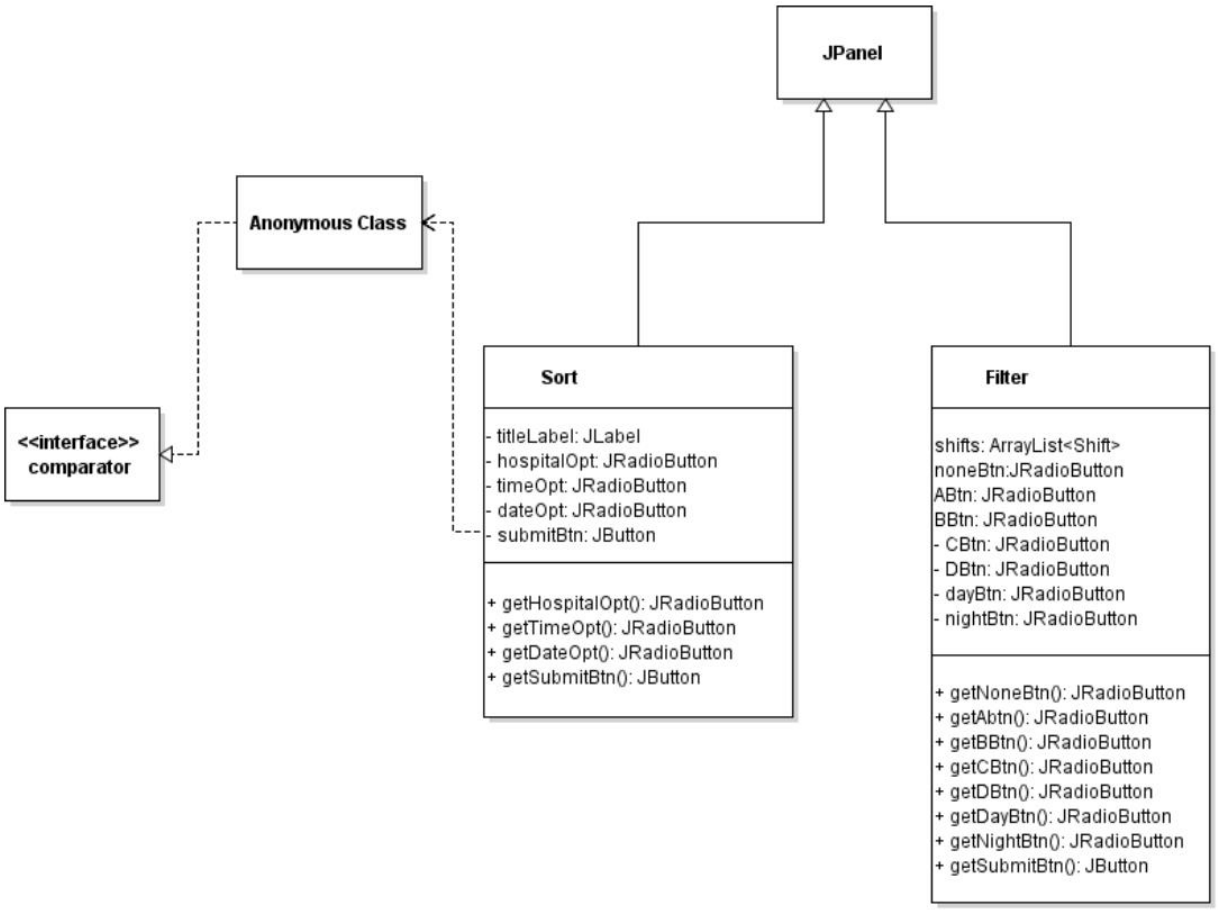
View All Nurses

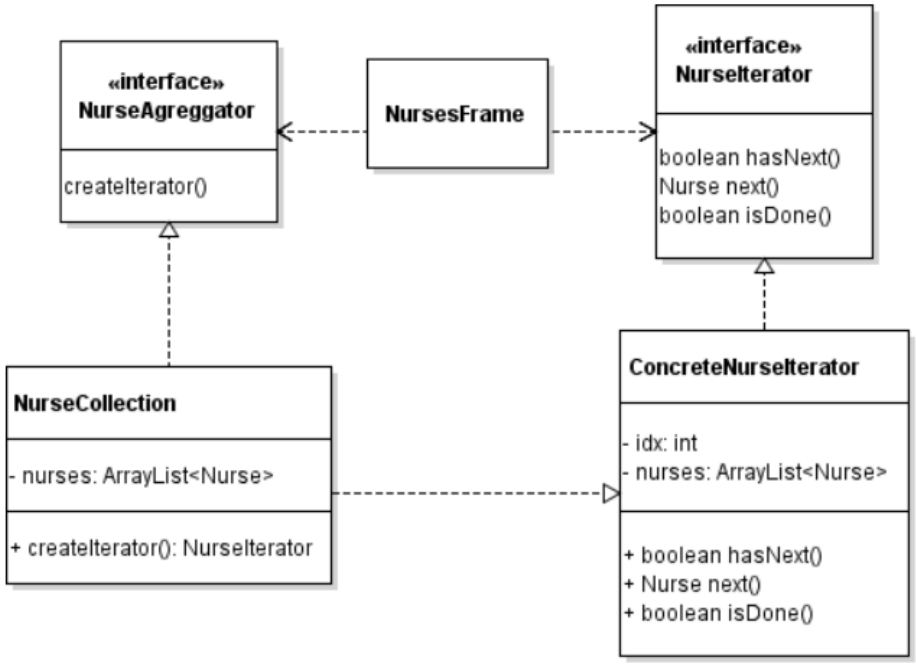


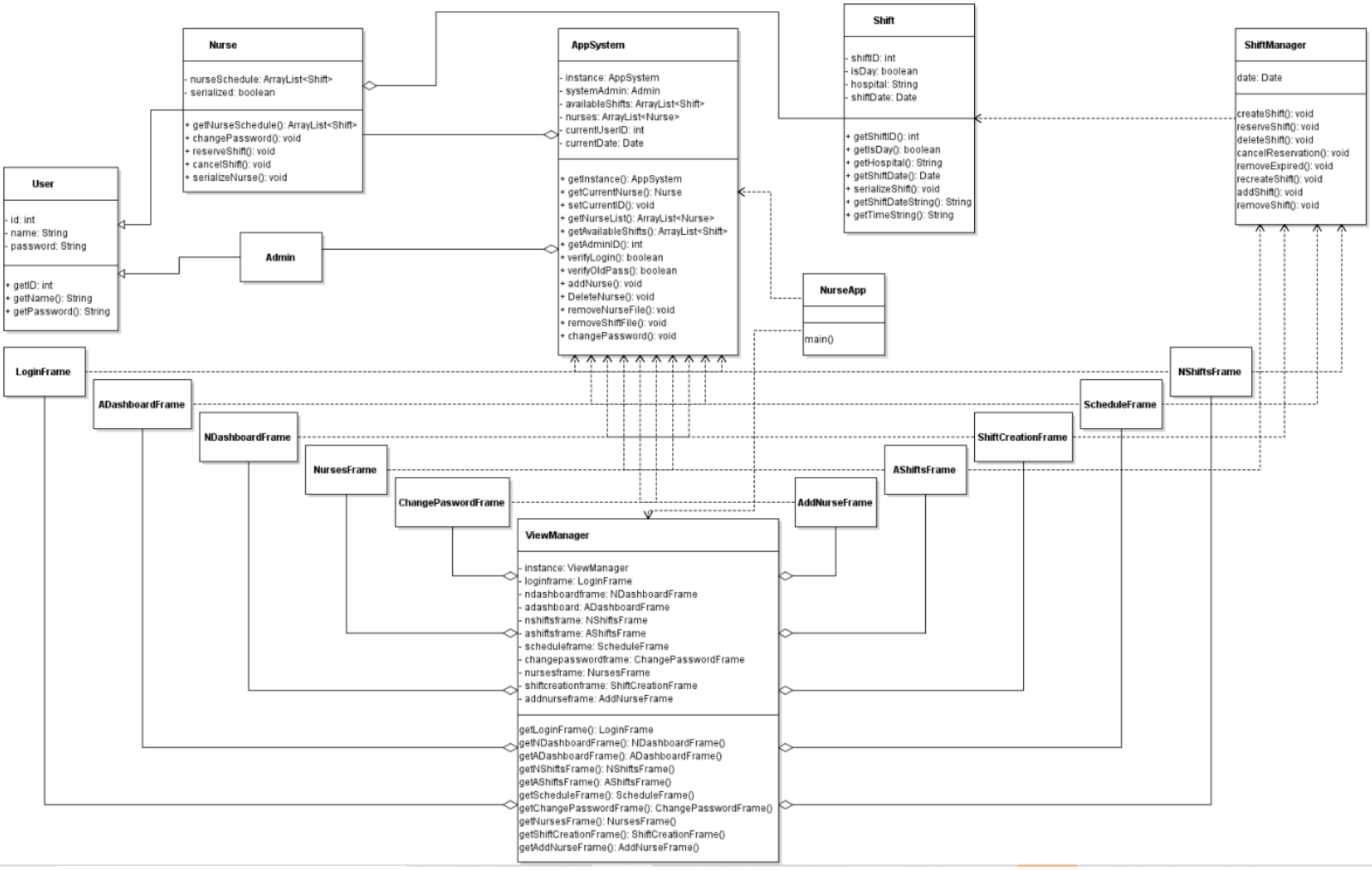
View Schedule Button

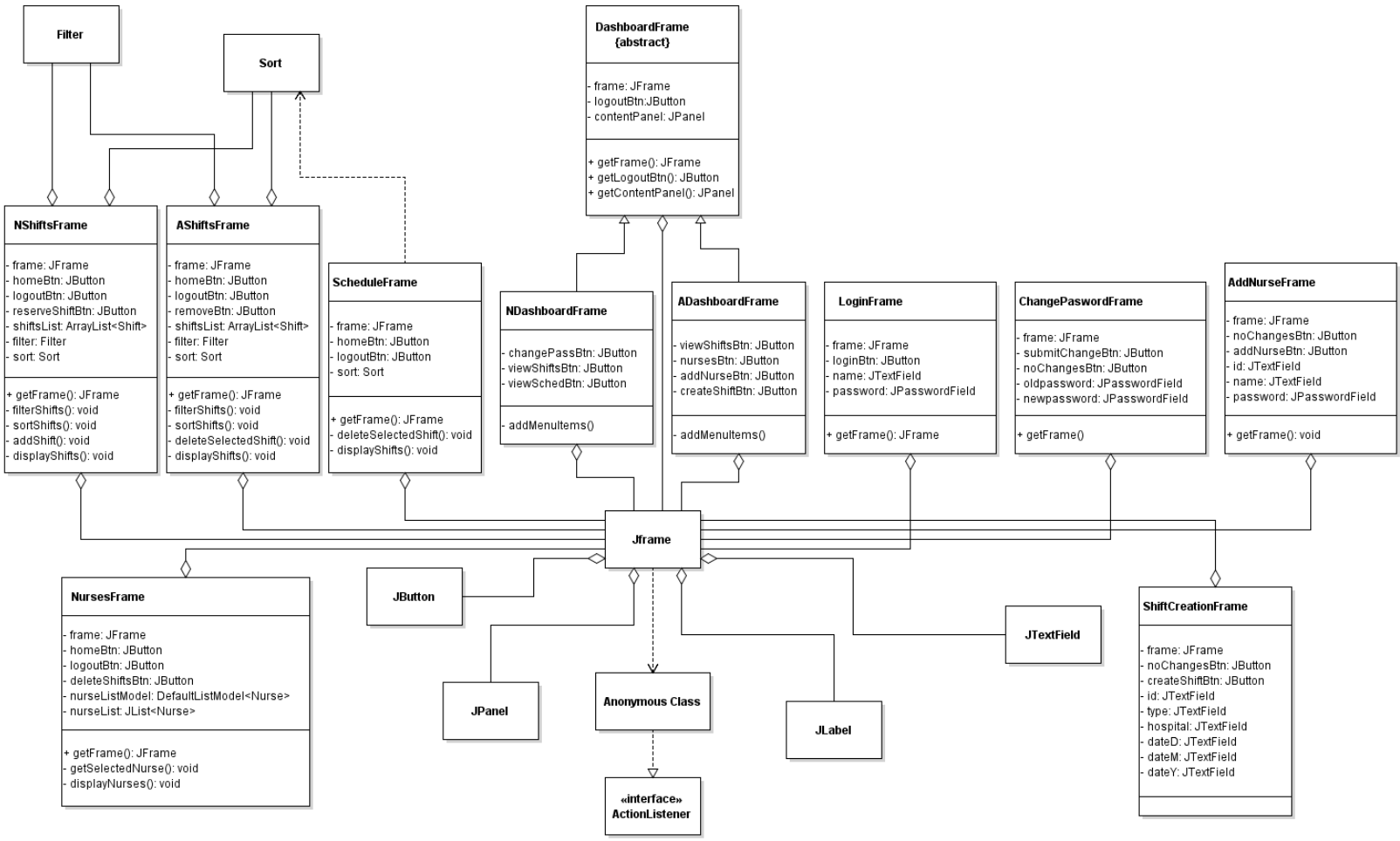


CLASS DIAGRAMS




«interface»
NurseAgreggator
createIterator()
NursesFrame
«interface»
NurseIterator
boolean hasNext()
Nurse next()
boolean isDone()
NurseCollection
- nurses: ArrayList<Nurse>
+ createIterator(): NurseIterator
ConcreteNurseIterator
- idx: int
- nurses: ArrayList<Nurse>
+ boolean hasNext()
+ Nurse next()
+ boolean isDone()


Nurse
- nurseSchedule: ArrayList<Shift>
- serialized: boolean
+ getNurseSchedule(): ArrayList<Shift>
+ changePassword(): void
+ reserveShift(): void
+ cancelShift(): void
+ serializeNurse(): void
AppSystem
- instance: AppSystem
- systemAdmin: Admin
- availableShifts: ArrayList<Shift>
- nurses: ArrayList<Nurse>
- currentUserID: int
- currentDate: Date
+ getInstance(): AppSystem
+ getCurrentNurse(): Nurse
+ setCurrentID(): void
+ getNurseList(): ArrayList<Nurse>
+ getAvailableShifts(): ArrayList<Shift>
+ getAdminID(): int
+ verifyLogin(): boolean
+ verifyOldPass(): boolean
+ addNurse(): void
+ DeleteNurse(): void
+ removeNurseFile(): void
+ removeShiftFile(): void
+ changePassword(): void
Shift
- shiftID: int
- isDay: boolean
- hospital: String
- shiftDate: Date
+ getShiftID(): int
+ getIsDay(): boolean
+ getHospital(): String
+ getShiftDate(): Date
+ serializeShift(): void
+ getShiftDateString(): String
+ getTimeString(): String
ShiftManager
date: Date
createShift(): void
reserveShift(): void
deleteShift(): void
cancelReservation(): void
removeExpired(): void
recreateShift(): void
addShift(): void
removeShift(): void
User
- id: int
- name: String
- password: String
+ getID: int
+ getName(): String
+ getPassword(): String
Admin
NurseApp
main()
LoginFrame
ADashboardFrame
NDashboardFrame
NursesFrame
ChangePaswordFrame
AddNurseFrame
AShiftsFrame
ShiftCreationFrame
ScheduleFrame
NShiftsFrame
ViewManager
- instance: ViewManager
- loginframe: LoginFrame
- ndashboardframe: NDashboardFrame
- adashboard: ADashboardFrame
- nshiftsframe: NShiftsFrame
- ashiftsframe: AShiftsFrame
- scheduleframe: ScheduleFrame
- changepasswordframe: ChangePasswordFrame
- nursesframe: NursesFrame
- shiftcreationframe: ShiftCreationFrame
- addnurseframe: AddNurseFrame
getLoginFrame(): LoginFrame
getNDashboardFrame(): NDashboardFrame()
getADashboardFrame(): ADashboardFrame()
getNShiftsFrame(): NShiftsFrame()
getAShiftsFrame(): AShiftsFrame()
getScheduleFrame(): ScheduleFrame()
getChangePasswordFrame(): ChangePasswordFrame()
getNursesFrame(): NursesFrame()
getShiftCreationFrame(): ShiftCreationFrame()
getAddNurseFrame(): AddNurseFrame()


Filter
Sort
DashboardFrame
{abstract}
- frame: JFrame
- logoutBtn:JButton
- contentPanel: JPanel
+ getFrame(): JFrame
+ getLogoutBtn(): JButton
+ getContentPanel(): JPanel
NShiftsFrame
- frame: JFrame
- homeBtn: JButton
- logoutBtn: JButton
- reserveShiftBtn: JButton
- shiftsList: ArrayList<Shift>
- filter: Filter
- sort: Sort
+ getFrame(): JFrame
- filterShifts(): void
- sortShifts(): void
- addShift(): void
- displayShifts(): void
AShiftsFrame
- frame: JFrame
- homeBtn: JButton
- logoutBtn: JButton
- removeBtn: JButton
- shiftsList: ArrayList<Shift>
- filter: Filter
- sort: Sort
+ getFrame(): JFrame
- filterShifts(): void
- sortShifts(): void
- deleteSelectedShift(): void
- displayShifts(): void
ScheduleFrame
- frame: JFrame
- homeBtn: JButton
- logoutBtn: JButton
- sort: Sort
+ getFrame(): JFrame
- deleteSelectedShift(): void
- displayShifts(): void
NDashboardFrame
- changePassBtn: JButton
- viewShiftsBtn: JButton
- viewSchedBtn: JButton
- addMenuItems()
ADashboardFrame
- viewShiftsBtn: JButton
- nursesBtn: JButton
- addNurseBtn: JButton
- createShiftBtn: JButton
- addMenuItems()
LoginFrame
- frame: JFrame
- loginBtn: JButton
- name: JTextField
- password: JPasswordField
+ getFrame(): JFrame
ChangePaswordFrame
- frame: JFrame
- submitChangeBtn: JButton
- noChangesBtn: JButton
- oldpassword: JPasswordField
- newpassword: JPasswordField
+ getFrame()
AddNurseFrame
- frame: JFrame
- noChangesBtn: JButton
- addNurseBtn: JButton
- id: JTextField
- name: JTextField
- password: JPasswordField
+ getFrame(): void
Jframe
NursesFrame
- frame: JFrame
- homeBtn: JButton
- logoutBtn: JButton
- deleteShiftsBtn: JButton
- nurseListModel: DefaultListModel<Nurse>
- nurseList: JList<Nurse>
+ getFrame(): JFrame
- getSelectedNurse(): void
- displayNurses(): void
JButton
JPanel
Anonymous Class
«interface»
ActionListener
JLabel
JTextField
ShiftCreationFrame
- frame: JFrame
- noChangesBtn: JButton
- createShiftBtn: JButton
- id: JTextField
- type: JTextField
- hospital: JTextField
- dateD: JTextField
- dateM: JTextField
- dateY: JTextField

SEQUENCE DIAGRAMS

SOURCE CODES

Login Frame

```
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 *Class represents the login frame. Sets up buttons and functionalities for login.
 */
public class LoginFrame {
    private JFrame frame = new JFrame("Login");
    private JButton loginBtn = new JButton("Login");
    private JTextField name = new JTextField(10);
    private JPasswordField password = new JPasswordField(10);

    /**
     * Constructor for the login frame.
     */
    public LoginFrame()
    {

        //Creating JLabel
        JLabel l1 = new JLabel("ID");
        JLabel l2 = new JLabel("Password");
```

```
l1.setLabelFor(name);

//Creating JPasswordField
l2.setLabelFor(password);
// Creating JLabel for the heading
JLabel heading = new JLabel("Welcome to eNurse");
heading.setFont(new Font("Poppins", Font.BOLD, 20));
heading.setBounds(100, 50, 300, 30);

JLabel subheading = new JLabel("please sign in");
subheading.setFont(new Font("Poppins", Font.ITALIC, 10));
subheading.setBounds(100, 70, 300, 30);



// Creating JButton
loginBtn.setFont(new Font("Poppins", Font.ITALIC, 10));
loginBtn.setBounds(100, 200, 100, 30);
frame.add(loginBtn);



/* This method specifies the location and size
 * of any component = > setBounds(x, y, width, height)
 * where x & y are cordinates from the top left
 * corner and remaining two parameters are the width
 * and height of the specific component.
```

```
 */
l1.setBounds(100, 100, 50, 30);
name.setBounds(200,100,150,30);

l2.setBounds(100, 150, 125, 30);
password.setBounds(200,150,150,30);


/* changing appearance of the label
 * Font,text color,background color
 */
l1.setFont(new Font("Poppins", Font.BOLD, 15));
name.setFont(new Font("Poppins", Font.ITALIC, 15));

l2.setFont(new Font("Poppins", Font.BOLD, 15));
name.setFont(new Font("Poppins", Font.ITALIC, 15));



//adds the labels to the frame
frame.add(l1);
frame.add(name);
frame.add(l2);
frame.add(password);
frame.add(heading);
frame.add(subheading);

frame.setLayout(null);
```

```
//sets the frame visibility to true
frame.setVisible(true);

//This method sets the width and height of the frame
frame.setSize(400, 500);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

ViewManager viewmanager = ViewManager.getInstance();
AppSystem appsys = AppSystem.getInstance();

viewmanager.attachListener(loginBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         String id = name.getText();
         String pass = password.getText();

         if (appsys.verifyLogin(Integer.parseInt(id), pass)==true)
         {

            viewmanager.setVisibility(frame, false);
            frame.dispose();

            if(Integer.parseInt(id) == appsys.getAdminID())
            {
               viewmanager.refreshADashboardFrame();
```

```
                    viewmanager.setVisibility(viewmanager.getADashboardFrame().getFrame(),
                    true);
                }
                else
                {
                   viewmanager.refreshNDashboardFrame();

                    viewmanager.setVisibility(viewmanager.getNDashboardFrame().getFrame(),
                    true);
                }

                name.setText("");
                password.setText("");

            }
            else {
                JOptionPane.showMessageDialog(frame, "Invalid ID or Password. Please try again.", "Login
Failed", JOptionPane.ERROR_MESSAGE);
                name.setText("");
                password.setText("");
            }
          }
        }
    );

  }

  /**
   * Returns the login frame.
```

```
     * @return
     */
    public JFrame getFrame()
    {
        return frame;
    }
}
```

ADashboardFrame.java

```
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Class that extends the DashboardFrame class, specifies and adds the menu
 * buttons for the admin user by overriding the abstract method from the
 * superclass. Class also provides an accessor method to the frame which
 * will enable the viewmanager class to attach the correct action listeners
 */
public class ADashboardFrame extends DashboardFrame{
    //private instance variables
    private JButton viewShiftsBtn = new JButton("View Shifts");
    private JButton nursesBtn = new JButton("View Nurses");
    private JButton addNurseBtn = new JButton("Add Nurse");
    private JButton createShiftBtn = new JButton("Create Shift");
```

```
/**
 * Constructor for ADashboardFrame. Sets up the menu buttons and their
 * dimensions, and adds action listeners to handle user interactions.
 */
public ADashboardFrame()
{
   //setting the dimensions for the menu buttons
   Dimension buttonSize = new Dimension(250, 200); // Adjust the size as needed
   viewShiftsBtn.setPreferredSize(buttonSize);
   nursesBtn.setPreferredSize(buttonSize);
   addNurseBtn.setPreferredSize(buttonSize);
   createShiftBtn.setPreferredSize(buttonSize);

   //adding menu panel to the content panel that was inherited from the super class
   getContentPanel().add( this.menuSetup(),BorderLayout.SOUTH);

   //retrieving the single instance of viewmanager and appsytem
   ViewManager viewmanager = ViewManager.getInstance();
   AppSystem appsys = AppSystem.getInstance();

   //logout button action listener
   viewmanager.attachListener(getLogoutBtn(),
      new ActionListener()
      {
         @Override
         public void actionPerformed(ActionEvent event)
         {
            viewmanager.setVisibility(getFrame(), false);
```

```
            getFrame().dispose();
            viewmanager.refreshLoginFrame();
            viewmanager.setVisibility(viewmanager.getLoginFrame().getFrame(), true);
            appsys.setCurrentID(0);

        }
    }
);

//add nurse button action listener
viewmanager.attachListener(addNurseBtn,
    new ActionListener()
    {
        @Override
        public void actionPerformed(ActionEvent event)
        {
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshAddNurseFrame();
            viewmanager.setVisibility(viewmanager.getAddNurseFrame().getFrame(), true);

        }
    }
);

//view shifts button action listener
 viewmanager.attachListener(viewShiftsBtn,
    new ActionListener()
    {
```

```
        @Override
        public void actionPerformed(ActionEvent event)
        {
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshAShiftsFrame();
            viewmanager.setVisibility(viewmanager.getAShiftsFrame().getFrame(), true);

        }
    }
);

//view nurses button action listener
viewmanager.attachListener(nursesBtn,
    new ActionListener()
    {
        @Override
        public void actionPerformed(ActionEvent event)
        {
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshNursesFrame();
            viewmanager.setVisibility(viewmanager.getNursesFrame().getFrame(), true);

        }
    }
);

//create new shift button action listener
```

```
    viewmanager.attachListener(createShiftBtn,
        new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent event)
            {
                viewmanager.setVisibility(getFrame(), false);
                getFrame().dispose();
                viewmanager.refreshShiftCreationFrame();
                viewmanager.setVisibility(viewmanager.getShiftCreationFrame().getFrame(), true);

            }
        }
    );
}

/**
 * method that overrides the abstract class method and adds the correct
 * buttons for the admin dashboard options
 * @param menu the panel that contains the menu buttons
 */
@Override
protected void addMenuItems(JPanel menu)
{

    menu.add(viewShiftsBtn);
    menu.add(nursesBtn);
    menu.add(addNurseBtn);
    menu.add(createShiftBtn);
```

```
    }

}
```

NDashboardFrame.java

```
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Class that extends the DashboardFrame class, specifies and adds the menu
 * buttons for the nurse user by overriding the abstract method from the
 * superclass. Class also provides an accessor method to the frame which
 * will enable the viewmanager class to attach the correct action listeners
 * @author Romari
 */
public class NDashboardFrame extends DashboardFrame{
    //private instance variables
    private JButton changePassBtn = new JButton("Change Password");
    private JButton viewShiftsBtn = new JButton("View Shifts");
    private JButton viewSchedBtn = new JButton("View Schedule");

    /**
```

```
 * constructor method that sets the size of the menu buttons, adds the menu
 * button panel to the content panel and attaches action listeners to the
 * buttons
 */
public NDashboardFrame()
{
   //setting the dimensions for the menu buttons
   Dimension buttonSize = new Dimension(250, 200); // Adjust the size as needed
   viewShiftsBtn.setPreferredSize(buttonSize);
   viewSchedBtn.setPreferredSize(buttonSize);
   changePassBtn.setPreferredSize(buttonSize);

   //adding menu panel to the content panel that was inherited from the super class
   getContentPanel().add( this.menuSetup(),BorderLayout.SOUTH);

   //retrieving the single instance of viewmanager and appysytem
   ViewManager viewmanager = ViewManager.getInstance();
   AppSystem appsys = AppSystem.getInstance();

   //logout button action listener
   viewmanager.attachListener(getLogoutBtn(),
      new ActionListener()
      {
         @Override
         public void actionPerformed(ActionEvent event)
         {
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshLoginFrame();
```

```
            viewmanager.setVisibility(viewmanager.getLoginFrame().getFrame(), true);
            appsys.setCurrentID(0);

        }
    }
);

//change password button action listener
viewmanager.attachListener(changePassBtn,
    new ActionListener()
    {
        @Override
        public void actionPerformed(ActionEvent event)
        {
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshChangePasswordFrame();
            viewmanager.setVisibility(viewmanager.getChangePasswordFrame().getFrame(), true);
        }
    }
);

//view shifts button action listener
 viewmanager.attachListener(viewShiftsBtn,
    new ActionListener()
    {
        @Override
        public void actionPerformed(ActionEvent event)
        {
```

```
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshNShiftsFrame();
            viewmanager.setVisibility(viewmanager.getNShiftsFrame().getFrame(), true);

        }
      }
    );

     //view schedule button action listener
    viewmanager.attachListener(viewSchedBtn,
      new ActionListener()
      {
        @Override
        public void actionPerformed(ActionEvent event)
        {
          viewmanager.setVisibility(getFrame(), false);
          getFrame().dispose();
          viewmanager.refreshScheduleFrame();
          viewmanager.setVisibility(viewmanager.getScheduleFrame().getFrame(), true);
        }
      }
    );
}

/**
 * method that overrides the abstract class method and adds the correct
 * buttons for the admin dashboard options
 * @param menu the panel that contains the menu buttons
```

```
     */
    @Override
    protected void addMenuItems(JPanel menu)
    {
        menu.add(viewShiftsBtn);
        menu.add(viewSchedBtn);
        menu.add(changePassBtn);

    }
}
```

SceduleFrame.java

```
    package project;

    import javax.swing.*;
    import java.awt.*;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import java.util.ArrayList;
    import java.util.Date;

    /**
     *Makes the schedule frame where the nurses sees their schedule.
     * Implements the GUI functionality.
     */
    public class ScheduleFrame {
        AppSystem appsys = AppSystem.getInstance();
        private JFrame frame = new JFrame("Schedule");
        private JButton homeBtn = new JButton("Home");
```

```
private JButton logoutBtn = new JButton("Logout");
private Sort sort = new Sort();

/**
 * Constructor for the schedule frame.
 */
public ScheduleFrame()
{
   JLabel heading = new JLabel("eNurse");
   heading.setFont(new Font("Poppins", Font.BOLD, 20));
   JLabel subheading = new JLabel("Schedule Frame");
   subheading.setFont(new Font("Poppins", Font.ITALIC, 10));




   // Creating a subpanel for horizontal buttons on the right
   JPanel rightPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
   rightPanel.add(homeBtn);
   rightPanel.add(logoutBtn);



   JPanel leftPanel = new JPanel();
   leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y_AXIS));
   leftPanel.add(heading);
   leftPanel.add(subheading);

   // Set layout manager for the frame
   frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));
```

```
// Creating a panel for the entire content with BorderLayout
JPanel headerPanel = new JPanel(new BorderLayout());
headerPanel.add(leftPanel, BorderLayout.WEST);
headerPanel.add(rightPanel, BorderLayout.EAST);
headerPanel.add(new JSeparator(), BorderLayout.SOUTH); // Horizontal Line

// Add the content panel to the frame
frame.add(headerPanel);



   // Second Layer
JPanel meepPanel = new JPanel(new FlowLayout(FlowLayout.CENTER)); // Use FlowLayout.CENTER for center alignment

JLabel subheading2 = new JLabel("Welcome");
subheading2.setFont(new Font("Poppins", Font.ITALIC, 30));
meepPanel.add(subheading2);



// Set layout manager for the frame
frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));
frame.add(meepPanel);



////frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
Schedule schedule = new Schedule();
schedule.createScheduleFrame(frame.getContentPane());


frame.setSize(1000, 800); // Adjust the size as needed
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


ViewManager viewmanager = ViewManager.getInstance();
AppSystem appsys = AppSystem.getInstance();

//logout button action listener
viewmanager.attachListener(logoutBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(frame, false);
         frame.dispose();
         viewmanager.refreshLoginFrame();
         viewmanager.setVisibility(viewmanager.getLoginFrame().getFrame(), true);
         appsys.setCurrentID(0);

      }
   }
);

//home button action listener
```

```
    viewmanager.attachListener(homeBtn,
        new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent event)
            {
                viewmanager.setVisibility(frame, false);
                frame.dispose();
                viewmanager.refreshNDashboardFrame();
                viewmanager.setVisibility(viewmanager.getNDashboardFrame().getFrame(), true);
            }
        }
    );
}
/**
 * Returns the schedule frame.
 * @return schedule frame
 */
public JFrame getFrame()
{
    return frame;
}

/**
* Inner class Schedule manages the display and interactions with the shift schedule.
* It handles the creation of UI components for displaying, filtering, sorting, and
* removing shifts.
*/
public class Schedule
```

```
{
    DefaultListModel<Shift> listModel = new DefaultListModel<>();
    JList<Shift> jList = new JList<>(listModel);

    public void createScheduleFrame(Container container) {
        // Create an ArrayList to store data
        //ArrayList<Shift> dataList = new ArrayList<>();

        displayShifts();



        jList.setCellRenderer(new MiniGrid());

        // Create a JScrollPane to allow scrolling if the list is too long
        JScrollPane scrollPane = new JScrollPane(jList);



        JButton removeButton = new JButton("Remove Shift");
        removeButton.addActionListener(e -> deleteSelectedShift());

        ArrayList<Shift> shiftsList = (ArrayList<Shift>) appsys.getAvailableShifts();



        JPanel westPanel = new JPanel();
        westPanel.setLayout(new BoxLayout(westPanel, BoxLayout.Y_AXIS));
        //westPanel.add(sortPanel);

        JPanel eastPanel = new JPanel();
```

```
        eastPanel.setLayout(new BoxLayout(eastPanel, BoxLayout.Y_AXIS));
        eastPanel.add(removeButton);

        // Create a panel to hold the JList and the Remove Shift button
        JPanel mainPanel = new JPanel(new BorderLayout());
        mainPanel.add(westPanel,BorderLayout.WEST);
        mainPanel.add(scrollPane, BorderLayout.CENTER); // Place JList at the top
        mainPanel.add(eastPanel, BorderLayout.EAST); // Remove button on the bottom of th
        container.add(mainPanel);



    }
    /**
     * Inner class MiniGrid extends DefaultListCellRenderer to customize the rendering of list items.
    */
    private class MiniGrid extends DefaultListCellRenderer
    {

        private static final int CELL_PADDING = 5;
        private static final int BORDER_THICKNESS = 1;
        private static final Color BORDER_COLOR = Color.BLACK;



        @Override
        public Component getListCellRendererComponent(JList<?> list, Object value, int index, boolean
isSelected, boolean cellHasFocus)
        {
            Shift shift = (Shift) value;
```

```
            JPanel panel = new JPanel(new GridLayout(4, 2, CELL_PADDING, CELL_PADDING));
            panel.setBackground(isSelected ? list.getSelectionBackground() : list.getBackground());
             //The Details of the panel

            panel.add(createLabel("Shift ID:"));
            panel.add(createLabel(Integer.toString(shift.getShiftID())));
            panel.add(createLabel("Date:"));
            panel.add(createLabel(shift.getShiftDateString()));
            panel.add(createLabel("Time:"));
            panel.add(createLabel(shift.getTimeString()));
            panel.add(createLabel("Hospital:"));
            panel.add(createLabel(shift.getHospital()));



            //The Border around the Panel
            panel.setBorder(BorderFactory.createCompoundBorder(
                    BorderFactory.createMatteBorder(0, 0, BORDER_THICKNESS, BORDER_THICKNESS,
BORDER_COLOR),
                    BorderFactory.createEmptyBorder(CELL_PADDING, CELL_PADDING, CELL_PADDING,
CELL_PADDING)
            ));
            return panel;
        }


        private JLabel createLabel(String text)
        {
            JLabel label = new JLabel(text);
            label.setHorizontalAlignment(SwingConstants.LEFT);
            return label;
```

```
    }
}
/**
 * Populates the JList with available shifts.
 * Displays the shifts using the iterator.
 */
public void displayShifts()
{

    var shifts = appsys.getCurrentNurse().getNurseSchedule();


    sort.sortCollection(shifts, sort.getCompByShiftDate());



    ShiftCollection shiftCollection = new ShiftCollection(shifts);

    ShiftIterator shiftIterator =shiftCollection.createIterator();
    while (shiftIterator.hasNext())
    {
        Shift shift = shiftIterator.next();
        listModel.addElement(shift);
        System.out.print(shift.toString() + " kool");
    }



    for(Shift shift: shifts)
    {
        System.out.println(shift.toString());
    }
```

```
    }
    /**
    * Deletes the selected shift from the list and system.
    */
    private void deleteSelectedShift()
    {

       int selectedIndex = jList.getSelectedIndex();
       if (selectedIndex != -1)
       {
          Shift selectedShift = listModel.get(selectedIndex);
          ShiftManager shiftManager = new ShiftManager();
          shiftManager.recreateShift(appsys.getAvailableShifts(), selectedShift); // Implement this method
in NurseManager
          listModel.remove(selectedIndex);
          shiftManager.cancelReservation(selectedShift);

       }
    }



  }
}
```

ADashboardFrame.java

```
package project;

import javax.swing.*;
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Class that extends the DashboardFrame class, specifies and adds the menu
 * buttons for the admin user by overriding the abstract method from the
 * superclass. Class also provides an accessor method to the frame which
 * will enable the viewmanager class to attach the correct action listeners
 */
public class ADashboardFrame extends DashboardFrame{
   //private instance variables
   private JButton viewShiftsBtn = new JButton("View Shifts");
   private JButton nursesBtn = new JButton("View Nurses");
   private JButton addNurseBtn = new JButton("Add Nurse");
   private JButton createShiftBtn = new JButton("Create Shift");

   /**
    * Constructor for ADashboardFrame. Sets up the menu buttons and their
    * dimensions, and adds action listeners to handle user interactions.
    */
   public ADashboardFrame()
   {
      //setting the dimensions for the menu buttons
      Dimension buttonSize = new Dimension(250, 200); // Adjust the size as needed
      viewShiftsBtn.setPreferredSize(buttonSize);
      nursesBtn.setPreferredSize(buttonSize);
      addNurseBtn.setPreferredSize(buttonSize);
      createShiftBtn.setPreferredSize(buttonSize);
```

```
//adding menu panel to the content panel that was inherited from the super class
getContentPanel().add( this.menuSetup(),BorderLayout.SOUTH);

//retrieving the single instance of viewmanager and appsytem
ViewManager viewmanager = ViewManager.getInstance();
AppSystem appsys = AppSystem.getInstance();

//logout button action listener
viewmanager.attachListener(getLogoutBtn(),
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(getFrame(), false);
         getFrame().dispose();
         viewmanager.refreshLoginFrame();
         viewmanager.setVisibility(viewmanager.getLoginFrame().getFrame(), true);
         appsys.setCurrentID(0);

      }
   }
);

//add nurse button action listener
viewmanager.attachListener(addNurseBtn,
   new ActionListener()
   {
```

```
        @Override
        public void actionPerformed(ActionEvent event)
        {
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshAddNurseFrame();
            viewmanager.setVisibility(viewmanager.getAddNurseFrame().getFrame(), true);

        }
    }
);

//view shifts button action listener
 viewmanager.attachListener(viewShiftsBtn,
    new ActionListener()
    {
        @Override
        public void actionPerformed(ActionEvent event)
        {
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshAShiftsFrame();
            viewmanager.setVisibility(viewmanager.getAShiftsFrame().getFrame(), true);

        }
    }
);

//view nurses button action listener
```

```
viewmanager.attachListener(nursesBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(getFrame(), false);
         getFrame().dispose();
         viewmanager.refreshNursesFrame();
         viewmanager.setVisibility(viewmanager.getNursesFrame().getFrame(), true);

      }
   }
);

//create new shift button action listener
viewmanager.attachListener(createShiftBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(getFrame(), false);
         getFrame().dispose();
         viewmanager.refreshShiftCreationFrame();
         viewmanager.setVisibility(viewmanager.getShiftCreationFrame().getFrame(), true);

      }
   }
```

```
        );
    }

    /**
     * method that overrides the abstract class method and adds the correct
     * buttons for the admin dashboard options
     * @param menu the panel that contains the menu buttons
     */
    @Override
    protected void addMenuItems(JPanel menu)
    {

        menu.add(viewShiftsBtn);
        menu.add(nursesBtn);
        menu.add(addNurseBtn);
        menu.add(createShiftBtn);

    }

}
```

AShiftsFrame.java

```
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
```

```
/**
 * Class that organizes all the components of the admin shifts frame and contains
 * the filter and sort panels. Class also provides an accessor method to the frame
 * which will enable the viewmanager class to attach the correct action listeners
 * @author Romari
 */
public class AShiftsFrame {
    AppSystem appsys = AppSystem.getInstance();
    private JFrame frame = new JFrame("Available Shifts");
    private JButton homeBtn = new JButton("Home");
    private JButton logoutBtn = new JButton("Logout");
    private JButton removeBtn = new JButton("Remove Shift");
    private ArrayList<Shift> shiftsList = appsys.getAvailableShifts();
    private Filter filter = new Filter(shiftsList);
    private Sort sort = new Sort();

    /**
     * constructor method that adds the panels and buttons and populates the frame.
     * The class attaches action listeners to the buttons using the viewmanager class
     */
    public AShiftsFrame()
    {
        JLabel heading = new JLabel("eNurse");
        heading.setFont(new Font("Poppins", Font.BOLD, 20));
        JLabel subheading = new JLabel("Admin Shifts Frame");
        subheading.setFont(new Font("Poppins", Font.ITALIC, 10));


        // Creating a subpanel for horizontal buttons on the right
```

```
JPanel rightPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
rightPanel.add(homeBtn);
rightPanel.add(logoutBtn);



JPanel leftPanel = new JPanel();
leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y_AXIS));
leftPanel.add(heading);
leftPanel.add(subheading);

// Set layout manager for the frame
frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));



// Creating a panel for the entire content with BorderLayout
JPanel headerPanel = new JPanel(new BorderLayout());
headerPanel.add(leftPanel, BorderLayout.WEST);
headerPanel.add(rightPanel, BorderLayout.EAST);
headerPanel.add(new JSeparator(), BorderLayout.SOUTH); // Horizontal Line

// Add the content panel to the frame
frame.add(headerPanel);



   // Second Layer
JPanel meepPanel = new JPanel(new FlowLayout(FlowLayout.CENTER)); // Use FlowLayout.CENTER for center alignment


JLabel subheading2 = new JLabel("Welcome");
```

```
subheading2.setFont(new Font("Poppins", Font.ITALIC, 30));
meepPanel.add(subheading2);


// Set layout manager for the frame
frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));
frame.add(meepPanel);


////frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

Schedule schedule = new Schedule();
schedule.createScheduleFrame(frame.getContentPane());


frame.setSize(1000, 800); // Adjust the size as needed
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
//////frame.setVisible(true);


ViewManager viewmanager = ViewManager.getInstance();
AppSystem appsys = AppSystem.getInstance();

//logout button action listener
viewmanager.attachListener(logoutBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
```

```
        {
            viewmanager.setVisibility(frame, false);
            frame.dispose();
            viewmanager.refreshLoginFrame();
            viewmanager.setVisibility(viewmanager.getLoginFrame().getFrame(), true);
            appsys.setCurrentID(0);

        }
      }
    );

    //home button action listener
    viewmanager.attachListener(homeBtn,
      new ActionListener()
      {
        @Override
        public void actionPerformed(ActionEvent event)
        {
            viewmanager.setVisibility(frame, false);
            viewmanager.setVisibility(viewmanager.getADashboardFrame().getFrame(), true);

        }
      }
    );
}

/**
 * method that returns AShiftsFrame frame
 *  @return The JFrame of this AShiftsFrame.
```

```
 */
public JFrame getFrame()
{
   return frame;
}



/**
* Inner class Schedule manages the display and interactions with the shift schedule.
* It handles the creation of UI components for displaying, filtering, sorting, and
* removing shifts.
*/
private class Schedule
{
   DefaultListModel<Shift> listModel = new DefaultListModel<>();
   JList<Shift> jList = new JList<>(listModel);

   /**
    * Sets up the schedule frame with necessary UI components.
    * @param container The container to add the schedule components to.
    */

   public void createScheduleFrame(Container container) {

      displayShifts();

      // These are the details that goes inside.
      jList.setCellRenderer(new MiniGrid());
```

```
// Create a JScrollPane to allow scrolling if the list is too long
JScrollPane scrollPane = new JScrollPane(jList);

// Create a Remove Shift button with ActionEvent.

removeBtn.addActionListener(e -> deleteSelectedShift());

filter.getSubmitBtn().addActionListener(e -> filterShifts());

sort.getSubmitBtn().addActionListener(e -> sortShifts());


JPanel westPanel = new JPanel();
westPanel.setLayout(new BoxLayout(westPanel, BoxLayout.Y_AXIS));
westPanel.add(filter);
westPanel.add(sort);

JPanel eastPanel = new JPanel();
eastPanel.setLayout(new BoxLayout(eastPanel, BoxLayout.Y_AXIS));
eastPanel.add(removeBtn);

// Create a panel to hold the JList and the Remove Shift button
JPanel mainPanel = new JPanel(new BorderLayout());
mainPanel.add(westPanel,BorderLayout.WEST);
mainPanel.add(scrollPane, BorderLayout.CENTER); // Place JList at the top
mainPanel.add(eastPanel, BorderLayout.EAST); // Remove button on the bottom of th
container.add(mainPanel);
```

```
    }
    /**
     * Inner class MiniGrid extends DefaultListCellRenderer to customize the rendering of list items.
    */
    private class MiniGrid extends DefaultListCellRenderer
    {

        private static final int CELL_PADDING = 5;
        private static final int BORDER_THICKNESS = 1;
        private static final Color BORDER_COLOR = Color.BLACK;



        @Override
        public Component getListCellRendererComponent(JList<?> list, Object value, int index, boolean
isSelected, boolean cellHasFocus)
        {
            Shift shift = (Shift) value;

            JPanel panel = new JPanel(new GridLayout(4, 2, CELL_PADDING, CELL_PADDING));
            panel.setBackground(isSelected ? list.getSelectionBackground() : list.getBackground());
             //The Details of the panel

            panel.add(createLabel("Shift ID:"));
            panel.add(createLabel(Integer.toString(shift.getShiftID())));
            panel.add(createLabel("Date:"));
            panel.add(createLabel(shift.getShiftDateString()));
            panel.add(createLabel("Time:"));
            panel.add(createLabel(shift.getTimeString()));
            panel.add(createLabel("Hospital:"));
```

```
            panel.add(createLabel(shift.getHospital()));



            //The Border around the Panel
            panel.setBorder(BorderFactory.createCompoundBorder(
                BorderFactory.createMatteBorder(0, 0, BORDER_THICKNESS, BORDER_THICKNESS,
BORDER_COLOR),
                BorderFactory.createEmptyBorder(CELL_PADDING, CELL_PADDING, CELL_PADDING,
CELL_PADDING)
            ));
            return panel;
        }
        /**
        * Method to create a JLabel with specified text.
        * @param text Text for the label.
        * @return A JLabel with the given text.
        */
        private JLabel createLabel(String text)
        {
            JLabel label = new JLabel(text);
            label.setHorizontalAlignment(SwingConstants.LEFT);
            return label;
        }
    }
     /**
     * Populates the JList with available shifts.
     * Displays the shifts using the iterator.
     */
    private void displayShifts()
```

```
{
   var shifts = appsys.getAvailableShifts();
   System.out.println("here");
   ShiftCollection shiftCollection = new ShiftCollection(shifts);

   ShiftIterator shiftIterator =shiftCollection.createIterator();
   while (shiftIterator.hasNext())
   {
      Shift shift = shiftIterator.next();
      listModel.addElement(shift);
   }
}
/**
* Deletes the selected shift from the list and system.
*/
private void deleteSelectedShift()
{
   int selectedIndex = jList.getSelectedIndex();
   if (selectedIndex != -1)
   {
      Shift selectedShift = listModel.get(selectedIndex);
      ShiftManager shiftManager = new ShiftManager();
      shiftManager.removeShift(selectedShift); // Implement this method in NurseManager
      listModel.remove(selectedIndex);
   }
}


/**
```

```
 * Applies filter to the shift list based on the selected filter options.
*/
private void filterShifts()
{
   listModel.clear();

   var shifts = appsys.getAvailableShifts();



   if(filter.getNoneBtn().isSelected())
   {
      System.out.println("radio button none");
      shifts = appsys.getAvailableShifts();
   }
   else if(filter.getABtn().isSelected())
   {
      System.out.println("radio button A");
      shifts = filter.filterShiftsByHospital(1);
   }
   else if(filter.getBBtn().isSelected())
   {
      System.out.println("radio button B");
      shifts = filter.filterShiftsByHospital(2);
   }
   else if(filter.getCBtn().isSelected())
   {
      System.out.println("radio button C");
      shifts = filter.filterShiftsByHospital(3);
   }
```

```
    else if(filter.getDBtn().isSelected())
    {
        System.out.println("radio button D");
        shifts = filter.filterShiftsByHospital(4);
    }
    else if(filter.getDayBtn().isSelected())
    {
        System.out.println("radio button day");
        shifts = filter.filterShiftsByType(1);
    }
    else if(filter.getNightBtn().isSelected())
    {
        System.out.println("radio button night");
        shifts = filter.filterShiftsByType(2);
    }


    ShiftCollection shiftCollection = new ShiftCollection(shifts);

    ShiftIterator shiftIterator =shiftCollection.createIterator();
    while (shiftIterator.hasNext())
    {
        Shift shift = shiftIterator.next();
        listModel.addElement(shift);
    }

}
 /**
* Sorts the shift list based on the selections.
```

```
*/
private void sortShifts()
{
   listModel.clear();

   var shifts = appsys.getAvailableShifts();

   if(sort.getHospitalOpt().isSelected())
   {
      sort.sortCollection(shifts, sort.getCompByShiftHospital());
   }
   else if(sort.getTimeOpt().isSelected())
   {
      sort.sortCollection(shifts, sort.getCompByShiftType());
   }
   else if(sort.getDateOpt().isSelected())
   {
      sort.sortCollection(shifts, sort.getCompByShiftDate());
   }

   ShiftCollection shiftCollection = new ShiftCollection(shifts);

   ShiftIterator shiftIterator =shiftCollection.createIterator();
   while (shiftIterator.hasNext())
   {
      Shift shift = shiftIterator.next();
      listModel.addElement(shift);
   }
```

```
        }


    }
}
```

AddNurseFrame.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Frame for adding a new nurse into the system.
 */
public class AddNurseFrame {
    private JFrame frame = new JFrame("Add Nurse");

    private JButton noChangesBtn = new JButton("Go Back/No Changes");
    private JButton addNurseBtn = new JButton("Add Nurse");
    private JTextField id = new JTextField(10);
    private JTextField name = new JTextField(10);
    private JPasswordField password = new JPasswordField(10);
```

```
 /**
 * Constructor for AddNurseFrame.
 * Initializes the UI components and sets up action listeners for the buttons.
 * When the add nurse button is clicked, a new nurse is added to the system.
 */
public AddNurseFrame()
{
   //Creating JLabel
   JLabel l1 = new JLabel("ID");
   JLabel l2 = new JLabel("Name");
   JLabel l3 = new JLabel("Password");

   //Creating JTextField
   l1.setLabelFor(id);

   l2.setLabelFor(name);

   l3.setLabelFor(password);

   // Creating JLabel for the heading
   JLabel heading = new JLabel("Add New Nurse");
   heading.setFont(new Font("Poppins", Font.BOLD, 20));
   heading.setBounds(75, 50, 300, 30);

   // Creating JButton
   addNurseBtn.setFont(new Font("Poppins", Font.ITALIC, 10));
   addNurseBtn.setBounds(50, 250, 200, 30);
   frame.add(addNurseBtn);
```

```
noChangesBtn.setFont(new Font("Poppins", Font.ITALIC, 10));
noChangesBtn.setBounds(50, 300, 200, 30);
frame.add(noChangesBtn);



/* This method specifies the location and size
 * of any component = > setBounds(x, y, width, height)
 * where x & y are cordinates from the top left
 * corner and remaining two parameters are the width
 * and height of the specific component.
 */
l1.setBounds(50, 100, 150, 30);
id.setBounds(175,100,150,30);

l2.setBounds(50, 150, 150, 30);
name.setBounds(175,150,150,30);

l3.setBounds(50, 200, 150, 30);
password.setBounds(175,200,150,30);



/* changing appearance of the label
 * Font,text color,background color
 */
l1.setFont(new Font("Poppins", Font.BOLD, 15));
id.setFont(new Font("Poppins", Font.ITALIC, 15));

l2.setFont(new Font("Poppins", Font.BOLD, 15));
```

```
name.setFont(new Font("Poppins", Font.ITALIC, 15));

l3.setFont(new Font("Poppins", Font.BOLD, 15));
password.setFont(new Font("Poppins", Font.ITALIC, 15));



//adds the labels to the frame
frame.add(l1);
frame.add(id);
frame.add(l2);
frame.add(name);
frame.add(l3);
frame.add(password);
frame.add(heading);



/* layout set to null ->
 * as no layout managers used
 * in this example (like FlowLayout,BoxLayout.etc)
 */
frame.setLayout(null);



//This method sets the width and height of the frame
frame.setSize(400, 500);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

ViewManager viewmanager = ViewManager.getInstance();
```

```
        AppSystem appsys = AppSystem.getInstance();

        //add nurse button action listener
        viewmanager.attachListener(addNurseBtn,
            new ActionListener()
            {
                @Override
                public void actionPerformed(ActionEvent event)
                {
                    String ID = id.getText();
                    String Name = name.getText();
                    String Password = password.getText();
                    appsys.addNurse(Integer.parseInt(ID), Name, Password);
                    JOptionPane.showMessageDialog(frame, "Nurse successfully added.", "Success",
JOptionPane.INFORMATION_MESSAGE);
                    viewmanager.setVisibility(frame, false);
                    frame.dispose();
                    viewmanager.refreshADashboardFrame();
                    viewmanager.setVisibility(viewmanager.getADashboardFrame().getFrame(), true);
                }
            }
        );

        //go back button action listener
        viewmanager.attachListener(noChangesBtn,
            new ActionListener()
            {
                @Override
                public void actionPerformed(ActionEvent event)
```

```
            {
                viewmanager.setVisibility(frame, false);
                frame.dispose();
                viewmanager.refreshADashboardFrame();
                viewmanager.setVisibility(viewmanager.getADashboardFrame().getFrame(), true);
                appsys.setCurrentID(0);

            }
          }
        );
    }
    /**
     * Returns the Add Nurse Frame
     * @return The JFrame associated with this class.
     */
    public JFrame getFrame()
    {
        return frame;
    }

}
```

Admin.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;
```

```
/**
 *Class for administrator in the system.
 */
public class Admin extends User{
    /**
     * Constructs a new Admin object with specified ID, name, and password.
     *
     * @param adminID The ID for for the admin.
     * @param name The name of the admin.
     * @param password The password for admin account access.
     */
   public Admin(int adminID, String name, String password)
   {
      super(adminID, name, password);
   }

}
```

AppSystem.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import java.util.ArrayList;
import java.util.Date;
import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.FilenameFilter;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.util.logging.Level;
import java.util.logging.Logger;
//import java.util.List;

/**
 * The AppSystem class is a singleton that manages the functionality of the application.
 * It handles the operations in the system.
 * This class ensures that there is only one instance of the system at any given time.
 */
public class AppSystem {

   private static AppSystem instance = null;

   private Admin systemAdmin;
   private static ArrayList<Shift> availableShifts = new ArrayList<>();
   private static ArrayList<Nurse> nurses = new ArrayList<>();
   private static int currentUserID;

   /**
    * Private constructor for AppSystem.
    * Initializes the system admin, loads nurses and shifts from serialized files.
    */
   private AppSystem()
   {
      currentUserID = 0 ;
      systemAdmin = new Admin(1000, "admin", "admin");
```

```
//serialize nurses
File nurseFolder = new File("nurses");
File[] nurseFiles = nurseFolder.listFiles();

for (File nurseFile : nurseFiles)
{
   if (nurseFile.isFile()) {
      try (ObjectInputStream nurseOis = new ObjectInputStream(new FileInputStream(nurseFile)))
      {
         Nurse nurseObject = (Nurse) nurseOis.readObject();
         nurses.add(nurseObject);
         System.out.println(nurseObject.getID());
         ArrayList<Shift> nurseSchedule = nurseObject.getNurseSchedule();
         for(Shift shift1 : nurseSchedule) {
          System.out.println(shift1);
          }
      } catch (IOException | ClassNotFoundException e) {
         e.printStackTrace();
      }
   }
}


//serialize available shifts
File shiftFolder = new File("shifts");
File[] shiftFiles = shiftFolder.listFiles();
```

```
  for (File shiftFile : shiftFiles)
  {
     if (shiftFile.isFile()) {
        try (ObjectInputStream shiftOis = new ObjectInputStream(new FileInputStream(shiftFile)))
        {
           Shift shiftObject = (Shift) shiftOis.readObject();
           availableShifts.add(shiftObject);
           System.out.println(shiftObject.getShiftID());
        } catch (IOException | ClassNotFoundException e) {
           e.printStackTrace();
        }
     }
  }



}
 /**
 * Returns the singleton instance of AppSystem.
 * @return The singleton instance of AppSystem.
 */
public static AppSystem getInstance()
{

   if (instance == null)
   {
      instance = new AppSystem();
   }

   return instance;
```

```
}
/**
 * Returns the currently logged-in nurse.
 * @return The current Nurse object, or null if no nurse is logged in.
 */
public Nurse getCurrentNurse()
{
   System.out.println(currentUserID+ "From getCurrentNurse");

   for(Nurse nurse: nurses)
   { System.out.println(nurse.getID()+ "Get ID From getCurrentNurse");
      if(currentUserID == nurse.getID())
      {

         return nurse;
      }
   }
   return null;
}
/**
 * Sets the current user ID.
 * @param ID The user ID to set to the current userID
 */
public static void setCurrentID(int ID)
{
   currentUserID = ID;
}
/**
 * Returns a list of the nurse objects.
```

```
 * @return An ArrayList of Nurse objects.
 */
public ArrayList<Nurse> getNurseList() {
   return nurses;
}
/**
 * Returns a list of the shift objects.
 * @return An ArrayList of shift objects.
 */
public ArrayList<Shift> getAvailableShifts()
{
   return availableShifts;
}

/**
 * Returns the admin ID.
 * @return An integer representing the admin ID.
 */
public int getAdminID()
{
   return systemAdmin.getID();
}

/**
 * Verifies login credentials for users.
 * @param ID The user ID.
 * @param password The password.
 * @return true if credentials are valid, false otherwise.
 */
```

```
public boolean verifyLogin(int ID, String password)
{
   if (ID == systemAdmin.getID())
   {
      if(password.equals(systemAdmin.getPassword()))
      {
         setCurrentID(ID);
         System.out.println("Admin Login successful");
         return true;
      }
   }
   else
   {
      for(Nurse nurse: nurses)
      {
         if(ID == nurse.getID())
         {
            if(password.equals(nurse.getPassword()))
            {
               setCurrentID(ID);
               System.out.println("Nurse Login successful");
               return true;
            }

         }
      }
   }
   return false;
}
```

```
/**
 * Verifies the old password is correct when changing password and changes the password.
 * @param oldPassword The old password.
 * @param newPassword The new password.
 * @return true if password is changed, false otherwise
 */
public boolean verifyOldPass(String oldPassword, String newPassword)
{
   for(Nurse nurse: nurses)
   {
      if(currentUserID == nurse.getID())
      {
         if (oldPassword.equals(nurse.getPassword()))
         {
            nurse.changePassword(newPassword);
            return true;
         }
      }
   }
   return false;
}
/**
 * Adds nurse object to the nurses ArrayList.
 * @param nurseID The nurseID to add.
 * @param name The nurse name to add.
 * @param password The nurse password to add.
 */
public void addNurse(int nurseID, String name, String password)
{
```

```
    nurses.add(new Nurse(nurseID, name, password));
}
/**
 * Deletes nurse based on the nurseID.
 * @param nurseID The nurseID to delete.
 */
public void deleteNurse(int nurseID)
{
    for(Nurse nurse: nurses)
    {
        if(nurseID == nurse.getID())
        {
            nurses.remove(nurse);
        }
    }
}

/**
 * Removes the nurse file when deleting nurse.
 * @param nurse Nurse to be deleted.
 */

public static void removeNurseFile(Nurse nurse){
    //nurses.remove(nurse);
    //System.out.println("removed");
    String directoryPath = "nurses";
    String filename = "nurses" + File.separator + String.valueOf(nurse.getID());
    File fileToDelete = new File(filename);
    if (fileToDelete.exists()) {
```

```
        boolean isDeleted = fileToDelete.delete();
        if (isDeleted) {
            System.out.println("Nurse file deleted successfully.");
        } else {
            System.out.println("Failed to delete the nurse file.");
        }
    }
}
/**
 *Removes shift file for deleted shifts.
 * @param shift Shift file to be deleted.
 */
public static void removeShiftFile(Shift shift){
    availableShifts.remove(shift);
    System.out.println("removed shift");
    String directoryPath = "shifts";
    String filename = "shifts" + File.separator + String.valueOf(shift.getShiftID());
    File fileToDelete = new File(filename);
    if (fileToDelete.exists()) {
        boolean isDeleted = fileToDelete.delete();
        if (isDeleted) {
            System.out.println("Shift file deleted successfully.");
        } else {
            System.out.println("Failed to delete the shift file.");
        }
    }
}
/**
 * Adds nurse objects to nurses ArrayList.
```

```
     * @param nurseObject
     */


    public void addNurse(Nurse nurseObject)
    {
        nurses.add(nurseObject);
    }
    /**
     * Reserializes the nurse when password is changed.
     */
    public void changePassword()
    {
        Nurse nurse = getCurrentNurse();
        removeNurseFile(nurse);
        try {
            nurse.serializeNurse(Integer.toString(nurse.getID()));
        } catch (IOException ex) {
            Logger.getLogger(AppSystem.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

ChangePasswordFrame.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 *Frame to change a password.
 */
public class ChangePasswordFrame {
   private JFrame frame = new JFrame("Change Password");

   private JButton submitChangeBtn = new JButton("Save New Password");
   private JButton noChangesBtn = new JButton("Go Back/No Changes");
   private JPasswordField oldpassword = new JPasswordField(10);
   private JPasswordField newpassword = new JPasswordField(10);
   /**
    * Change password frame constructor, initializes GUI components.
    */
   public ChangePasswordFrame()
   {


      //Creating JLabel
      JLabel l1 = new JLabel("Old Password");
      JLabel l2 = new JLabel("New Password");

      l1.setLabelFor(oldpassword);
```

```
l2.setLabelFor(newpassword);

// Creating JLabel for the heading
JLabel heading = new JLabel("Change Your Password");
heading.setFont(new Font("Poppins", Font.BOLD, 20));
heading.setBounds(50, 50, 300, 30);

JLabel subheading = new JLabel("please enter old password");
subheading.setFont(new Font("Poppins", Font.ITALIC, 10));
subheading.setBounds(175, 120, 300, 30);
frame.add(subheading);

JLabel subheading2 = new JLabel("please enter your new password");
subheading2.setFont(new Font("Poppins", Font.ITALIC, 10));
subheading2.setBounds(175, 170, 300, 30);
frame.add(subheading2);

// Creating JButton
submitChangeBtn.setFont(new Font("Poppins", Font.ITALIC, 10));
submitChangeBtn.setBounds(50, 225, 200, 30);
frame.add(submitChangeBtn);


// Creating JButton
noChangesBtn.setFont(new Font("Poppins", Font.ITALIC, 10));
noChangesBtn.setBounds(50, 275, 200, 30);
frame.add(noChangesBtn);
```

```
/* This method specifies the location and size
 * of any component = > setBounds(x, y, width, height)
 * where x & y are cordinates from the top left
 * corner and remaining two parameters are the width
 * and height of the specific component.
 */
l1.setBounds(50, 100, 200, 30);
oldpassword.setBounds(175,100,150,30);

l2.setBounds(50, 150, 200, 30);
newpassword.setBounds(175,150,150,30);


/* changing appearance of the label
 * Font,text color,background color
 */
l1.setFont(new Font("Poppins", Font.BOLD, 15));
oldpassword.setFont(new Font("Poppins", Font.ITALIC, 15));



l2.setFont(new Font("Poppins", Font.BOLD, 15));
newpassword.setFont(new Font("Poppins", Font.ITALIC, 15));



//adds the labels to the frame
```

```
frame.add(l1);
frame.add(oldpassword);
frame.add(l2);
frame.add(newpassword);
frame.add(heading);



/* layout set to null ->
 * as no layout managers used
 * in this example (like FlowLayout,BoxLayout.etc)
 */
frame.setLayout(null);

//This method sets the width and height of the frame
frame.setSize(400, 500);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

ViewManager viewmanager = ViewManager.getInstance();
AppSystem appsys = AppSystem.getInstance();

//submit change button action listener
viewmanager.attachListener(submitChangeBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         String oldPassword = oldpassword.getText();
         String newPassword = newpassword.getText();
```

```
            if (appsys.verifyOldPass(oldPassword, newPassword)==true)
            {
                JOptionPane.showMessageDialog(frame, "Password successfully changed.", "Success",
JOptionPane.INFORMATION_MESSAGE);
                viewmanager.setVisibility(frame, false);
                frame.dispose();
                viewmanager.refreshNDashboardFrame();
                viewmanager.setVisibility(viewmanager.getNDashboardFrame().getFrame(), true);
                appsys.changePassword();

            }
            else{
                JOptionPane.showMessageDialog(frame, "Invalid Password. Please try again.", "Password
Change Failed", JOptionPane.ERROR_MESSAGE);
                oldpassword.setText("");
                newpassword.setText("");
            }
        }
    }
);

    //go back button action listener
    viewmanager.attachListener(noChangesBtn,
        new ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent event)
            {
```

```
                viewmanager.setVisibility(frame, false);
                viewmanager.setVisibility(viewmanager.getNDashboardFrame().getFrame(), true);
                appsys.setCurrentID(0);

            }
        }
    );
  }
  /**
   * Returns the Change Password Frame.
   * @return The change password frame.
   */

  public JFrame getFrame()
  {
     return frame;
  }

}
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import java.util.ArrayList;

/**
 * A concrete iterator for iterating over a collection of Nurse objects.
```

```
 * Implements the NurseIterator interface and provides functionality
 * to iterate through the nurses.
 */
public class ConcreteNurseIterator  implements NurseIterator{
    private int idx;
    private ArrayList<Nurse> nurses;
     /**
     * Constructor to initialize the ConcreteNurseIterator with a list of nurses.
     * @param nurses The list of Nurse objects to iterate over.
     */
    public ConcreteNurseIterator(ArrayList<Nurse> nurses) {
        this.nurses = nurses;
        idx = 0;
    }
    /**
     * Checks if there is a next nurse to iterate over.
     * @return Returns a boolean if there is another nurse next.
     */

    @Override
    public boolean hasNext() {
        return idx < nurses.size();
    }
    /**
     * Returns the next nurse in the iteration and advances.
     * @return Returns the next nurse object in the list.
     */

    @Override
```

```
    public Nurse next() {
        return nurses.get(idx++);
    }
    /**
     * Checks if the end of the list has been reached.
     * @return Returns a boolean stating whether the end of the list has been reached.
     */
    @Override
    public boolean isDone()
    {
        return idx == nurses.size();
    }
}
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import java.util.ArrayList;

/**
 * A concrete iterator for iterating over a collection of Shift objects.
 * Implements the ShiftIterator interface and provides functionality
 * to iterate through the shifts.
 */
public class ConcreteShiftIterator implements ShiftIterator{
    private int idx;
    private ArrayList<Shift> shifts;
```

```
/**
 * Constructor to initialize the ConcreteShiftIterator with a list of shifts.
 * @param shifts The list of Shifts objects to iterate over.
 */
public ConcreteShiftIterator(ArrayList<Shift> shifts) {
    this.shifts = shifts;
    idx = 0;
}
/**
 * Checks if there is a next shift to iterate over.
 * @return Returns a boolean if there is another shift next.
 */
@Override
public boolean hasNext() {
    return idx < shifts.size();
}
/**
 * Returns the next shift in the iteration and advances.
 * @return Returns the next shift object in the list.
 */
@Override
public Shift next() {
    return shifts.get(idx++);
}
/**
 * Checks if the end of the list has been reached.
 * @return Returns a boolean stating whether the end of the list has been reached.
 */
```

```
    @Override
    public boolean isDone()
    {
        return idx == shifts.size();
    }
}
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

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

/**
 * An abstract class representing a dashboard frame. This class provides a common structure
 * for Nurse Dashboard and Admin Dashboard.
 * It includes a logout button, content panel, and a method for setting up the menu.
 */
public abstract class DashboardFrame {
    private JFrame frame = new JFrame("dashboard");
    private JButton logoutBtn = new JButton("Logout");
    private JPanel contentPanel = new JPanel(new BorderLayout());
    /**
     * Constructor for DashboardFrame.
     */
    public DashboardFrame()
    {
```

```
JLabel heading = new JLabel("eNurse");
heading.setFont(new Font("Poppins", Font.BOLD, 20));
JLabel subheading = new JLabel("Nurse Dashboard");
subheading.setFont(new Font("Poppins", Font.ITALIC, 10));

JPanel leftPanel = new JPanel();
leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y_AXIS));
leftPanel.add(heading);
leftPanel.add(subheading);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
buttonPanel.add(logoutBtn);



//The Header BorderLayout
contentPanel.add(leftPanel, BorderLayout.WEST);
contentPanel.add(buttonPanel, BorderLayout.NORTH);
contentPanel.add(new JSeparator(), BorderLayout.CENTER); // Horizontal Line

JLabel welcomeLabel = new JLabel("Welcome");
welcomeLabel.setFont(new Font("Poppins", Font.BOLD, 20));
JPanel welcomePanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
welcomePanel.add(welcomeLabel, BorderLayout.CENTER);
contentPanel.add(welcomeLabel);

// Set layout manager for the frame
frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));
```

```
    // Add the content panel to the frame
    frame.add(contentPanel);

    // This method sets the width and height of the frame
    frame.setSize(800, 800); // Adjust the size as needed
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
/**
 * Method to be implemented by subclasses for adding menu items to the dashboard.
 * @param menu The JPanel to which the menu items will be added.
 */
public JPanel menuSetup()
{
    JPanel menu = new JPanel(new FlowLayout());
    addMenuItems(menu);
    int bottomPadding = 350;
    menu.setBorder(BorderFactory.createEmptyBorder(0, 0, bottomPadding, 0));
    return menu;
}
 /**
 * Sets up the menu panel to be added to the dashboard.
 * @return A JPanel containing the menu items.
 */
protected abstract void addMenuItems(JPanel menu);
 /**
 * Gets the frame of the dashboard.
 * @return The JFrame of the dashboard.
 */
public JFrame getFrame()
```

```
    {
        return frame;
    }
    /**
     * Gets the logout button of the dashboard.
     * @return The logout JButton on the dashboard.
     */
    public JButton getLogoutBtn()
    {
        return logoutBtn;
    }
    /**
     * Gets the content panel of the dashboard.
     * @return The content JPanel of the dashboard.
     */
    public JPanel getContentPanel()
    {
        return contentPanel;
    }



}
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import javax.swing.*;
```

```
import java.util.ArrayList;

/**
 * This class represents a filter panel used to filter shifts based on various criteria.
 */
public class Filter extends JPanel{

    private ArrayList<Shift> shifts; // Assuming you have a list of shifts
    private JRadioButton noneBtn = new JRadioButton("none");
    private JRadioButton ABtn = new JRadioButton("Hospital A");
    private JRadioButton BBtn = new JRadioButton("Hospital B");
    private JRadioButton CBtn = new JRadioButton("Hospital C");
    private JRadioButton DBtn = new JRadioButton("Hospital D");
    private JRadioButton dayBtn = new JRadioButton("Day");
    private JRadioButton nightBtn = new JRadioButton("Night");

    private JLabel titleLabel = new JLabel("Filter Shifts");
    private JButton submitButton = new JButton("Submit");

     /**
     * Constructor for the Filter class.
     * @param shifts An ArrayList of Shift objects to be filtered.
     */
    public Filter(ArrayList<Shift> shifts) {
        this.shifts = shifts;
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));

        ButtonGroup hospitalGroup = new ButtonGroup();
```

```
    hospitalGroup.add(noneBtn);
    hospitalGroup.add(ABtn);
    hospitalGroup.add(BBtn);
    hospitalGroup.add(CBtn);
    hospitalGroup.add(DBtn);
    hospitalGroup.add(dayBtn);
    hospitalGroup.add(nightBtn);

    this.add(titleLabel);
    this.add(noneBtn);
    this.add(ABtn);
    this.add(BBtn);
    this.add(CBtn);
    this.add(DBtn);
    this.add(dayBtn);
    this.add(nightBtn);
    this.add(submitButton);
}

/**
 * Filters the provided list of shifts based on the specified hospital.
 * @param shifts The list of Shift objects to filter.
 * @param hospital The hospital criteria for filtering.
 * @return A filtered ArrayList of Shift objects based on the hospital criteria.
 */
private ArrayList<Shift> filterShiftsByHospital(ArrayList<Shift> shifts, int hospital)
{
    //filer by hospital, type, date
```

```
    ArrayList<Shift> filteredShifts = new ArrayList<>();

    for(Shift shift: shifts)
    {
        if( (hospital == 1 && shift.getHospital().equals("hospital a")) ||
            (hospital == 2 && shift.getHospital().equals("hospital b")) ||
            (hospital == 3 && shift.getHospital().equals("hospital c")) ||
            (hospital == 4 && shift.getHospital().equals("hospital d"))
        )
        {
            filteredShifts.add(shift);
        }
    }

    return filteredShifts;
}
/**
 * Filters the provided list of shifts based on the specified shift type (day or night).
 * @param shifts The list of Shift objects to filter.
 * @param type The shift type criteria for filtering.
 * @return A filtered ArrayList of Shift objects based on the shift type criteria.
 */
private ArrayList<Shift> filterShiftsByType(ArrayList<Shift> shifts, int type)
{
    //filer by hospital, type, date

    ArrayList<Shift> filteredShifts = new ArrayList<>();

    for(Shift shift: shifts)
```

```
    {
        if( (type == 1 && shift.getIsDay()==true) || (type == 2 && shift.getIsDay() ==false))
        {
            filteredShifts.add(shift);
        }
    }

    return filteredShifts;
}
/**
 * Filters the shifts by hospital using the class's instance variable.
 * @param hospital The hospital criteria for filtering.
 * @return A filtered ArrayList of Shift objects.
 */
 public ArrayList<Shift> filterShiftsByHospital(int hospital) {
    return filterShiftsByHospital(shifts, hospital);
}
 /**
 * Filters the shifts by type (day or night) using the class's instance variable.
 * @param type The shift type criteria for filtering.
 * @return A filtered ArrayList of Shift objects.
 */
// Method to filter shifts by type without passing a list (assumes use of instance variable)
public ArrayList<Shift> filterShiftsByType(int type) {
    return filterShiftsByType(shifts, type);
}

/**
 * Returns the getNoneButton
```

```
 * @return
 */
public JRadioButton getNoneBtn()
{
   return noneBtn;
}
/**
 * Returns the getABtn
 * @return
 */
public JRadioButton getABtn()
{
   return ABtn;
}
/**
 * Returns the BBtn
 * @return
 */
public JRadioButton getBBtn()
{
   return BBtn;
}
/**
 * Returns the CBtn
 * @return
 */
public JRadioButton getCBtn()
{
   return CBtn;
```

```
}
/**
 * Returns the DBtn
 * @return
 */
public JRadioButton getDBtn()
{
   return DBtn;
}
/**
 * Returns the DayBtn
 * @return
 */
public JRadioButton getDayBtn()
{
   return dayBtn;
}
/**
 * Returns the getNightBtn
 * @return
 */
public JRadioButton getNightBtn()
{
   return nightBtn;
}
/**
 * Returns the Submit Button
 * @return
 */
```

```
    public JButton getSubmitBtn()
    {
        return submitButton;
    }

}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 *Class represents the login frame. Sets up buttons and functionalities for login.
 */
public class LoginFrame {
    private JFrame frame = new JFrame("Login");
    private JButton loginBtn = new JButton("Login");
    private JTextField name = new JTextField(10);
    private JPasswordField password = new JPasswordField(10);

    /**
     * Constructor for the login frame.
```

```
 */
public LoginFrame()
{

   //Creating JLabel
   JLabel l1 = new JLabel("ID");
   JLabel l2 = new JLabel("Password");



   l1.setLabelFor(name);

   //Creating JPasswordField
   l2.setLabelFor(password);
   // Creating JLabel for the heading
   JLabel heading = new JLabel("Welcome to eNurse");
   heading.setFont(new Font("Poppins", Font.BOLD, 20));
   heading.setBounds(100, 50, 300, 30);

   JLabel subheading = new JLabel("please sign in");
   subheading.setFont(new Font("Poppins", Font.ITALIC, 10));
   subheading.setBounds(100, 70, 300, 30);



   // Creating JButton
   loginBtn.setFont(new Font("Poppins", Font.ITALIC, 10));
   loginBtn.setBounds(100, 200, 100, 30);
   frame.add(loginBtn);
```

```
/* This method specifies the location and size
 * of any component = > setBounds(x, y, width, height)
 * where x & y are cordinates from the top left
 * corner and remaining two parameters are the width
 * and height of the specific component.
 */
l1.setBounds(100, 100, 50, 30);
name.setBounds(200,100,150,30);

l2.setBounds(100, 150, 125, 30);
password.setBounds(200,150,150,30);



/* changing appearance of the label
 * Font,text color,background color
 */
l1.setFont(new Font("Poppins", Font.BOLD, 15));
name.setFont(new Font("Poppins", Font.ITALIC, 15));

l2.setFont(new Font("Poppins", Font.BOLD, 15));
name.setFont(new Font("Poppins", Font.ITALIC, 15));



//adds the labels to the frame
frame.add(l1);
```

```
frame.add(name);
frame.add(l2);
frame.add(password);
frame.add(heading);
frame.add(subheading);

frame.setLayout(null);

//sets the frame visibility to true
frame.setVisible(true);

//This method sets the width and height of the frame
frame.setSize(400, 500);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

ViewManager viewmanager = ViewManager.getInstance();
AppSystem appsys = AppSystem.getInstance();

viewmanager.attachListener(loginBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         String id = name.getText();
         String pass = password.getText();

         if (appsys.verifyLogin(Integer.parseInt(id), pass)==true)
         {
```

```
                viewmanager.setVisibility(frame, false);
                frame.dispose();

                if(Integer.parseInt(id) == appsys.getAdminID())
                {
                   viewmanager.refreshADashboardFrame();
                   viewmanager.setVisibility(viewmanager.getADashboardFrame().getFrame(), true);
                }
                else
                {
                   viewmanager.refreshNDashboardFrame();
                   viewmanager.setVisibility(viewmanager.getNDashboardFrame().getFrame(), true);
                }

                name.setText("");
                password.setText("");

            }
            else {
                JOptionPane.showMessageDialog(frame, "Invalid ID or Password. Please try again.", "Login
Failed", JOptionPane.ERROR_MESSAGE);
                name.setText("");
                password.setText("");
            }
        }
    }
);
```

```
    }

    /**
     * Returns the login frame.
     * @return
     */
    public JFrame getFrame()
    {
        return frame;
    }
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Class that extends the DashboardFrame class, specifies and adds the menu
 * buttons for the nurse user by overriding the abstract method from the
 * superclass. Class also provides an accessor method to the frame which
 * will enable the viewmanager class to attach the correct action listeners
 * @author Romari
```

```
 */
public class NDashboardFrame extends DashboardFrame{
   //private instance variables
   private JButton changePassBtn = new JButton("Change Password");
   private JButton viewShiftsBtn = new JButton("View Shifts");
   private JButton viewSchedBtn = new JButton("View Schedule");

   /**
    * constructor method that sets the size of the menu buttons, adds the menu
    * button panel to the content panel and attaches action listeners to the
    * buttons
    */
   public NDashboardFrame()
   {
      //setting the dimensions for the menu buttons
      Dimension buttonSize = new Dimension(250, 200); // Adjust the size as needed
      viewShiftsBtn.setPreferredSize(buttonSize);
      viewSchedBtn.setPreferredSize(buttonSize);
      changePassBtn.setPreferredSize(buttonSize);

      //adding menu panel to the content panel that was inherited from the super class
      getContentPanel().add( this.menuSetup(),BorderLayout.SOUTH);

      //retrieving the single instance of viewmanager and appsytem
      ViewManager viewmanager = ViewManager.getInstance();
      AppSystem appsys = AppSystem.getInstance();

      //logout button action listener
      viewmanager.attachListener(getLogoutBtn(),
```

```
    new ActionListener()
    {
        @Override
        public void actionPerformed(ActionEvent event)
        {
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshLoginFrame();
            viewmanager.setVisibility(viewmanager.getLoginFrame().getFrame(), true);
            appsys.setCurrentID(0);

        }
    }
);

//change password button action listener
viewmanager.attachListener(changePassBtn,
    new ActionListener()
    {
        @Override
        public void actionPerformed(ActionEvent event)
        {
            viewmanager.setVisibility(getFrame(), false);
            getFrame().dispose();
            viewmanager.refreshChangePasswordFrame();
            viewmanager.setVisibility(viewmanager.getChangePasswordFrame().getFrame(), true);
        }
    }
);
```

```
//view shifts button action listener
 viewmanager.attachListener(viewShiftsBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(getFrame(), false);
         getFrame().dispose();
         viewmanager.refreshNShiftsFrame();
         viewmanager.setVisibility(viewmanager.getNShiftsFrame().getFrame(), true);

      }
   }
);

 //view schedule button action listener
viewmanager.attachListener(viewSchedBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(getFrame(), false);
         getFrame().dispose();
         viewmanager.refreshScheduleFrame();
         viewmanager.setVisibility(viewmanager.getScheduleFrame().getFrame(), true);
      }
```

```
        }
      );
   }

   /**
    * method that overrides the abstract class method and adds the correct
    * buttons for the admin dashboard options
    * @param menu the panel that contains the menu buttons
    */
   @Override
   protected void addMenuItems(JPanel menu)
   {
      menu.add(viewShiftsBtn);
      menu.add(viewSchedBtn);
      menu.add(changePassBtn);

   }
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.util.ArrayList;


/**
 *Represents the frame that shows the nurses their shifts.
 * Implements the GUI functionality for the frame.
 */
public class NShiftsFrame {
    AppSystem appsys = AppSystem.getInstance();
    private JFrame frame = new JFrame("Available Shifts");
    private JButton homeBtn = new JButton("Home");
    private JButton logoutBtn = new JButton("Logout");
    private JButton reserveShiftsBtn = new JButton("Reserve Shift");
    private ArrayList<Shift> shiftsList = appsys.getAvailableShifts();
    private Filter filter = new Filter(shiftsList);
    private Sort sort = new Sort();
    /**
     * Constructor for NShiftsFrame. Initializes the frame and sets up the user interface components.
     */
    public NShiftsFrame()
    {
        JLabel heading = new JLabel("eNurse");
        heading.setFont(new Font("Poppins", Font.BOLD, 20));
        JLabel subheading = new JLabel("Nurse Shifts Frame");
        subheading.setFont(new Font("Poppins", Font.ITALIC, 10));



        // Creating a subpanel for horizontal buttons on the right
        JPanel rightPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
```

```
        rightPanel.add(homeBtn);
        rightPanel.add(logoutBtn);



        JPanel leftPanel = new JPanel();
        leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y_AXIS));
        leftPanel.add(heading);
        leftPanel.add(subheading);

        // Set layout manager for the frame
        frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));



        // Creating a panel for the entire content with BorderLayout
        JPanel headerPanel = new JPanel(new BorderLayout());
        headerPanel.add(leftPanel, BorderLayout.WEST);
        headerPanel.add(rightPanel, BorderLayout.EAST);
        headerPanel.add(new JSeparator(), BorderLayout.SOUTH); // Horizontal Line

        // Add the content panel to the frame
        frame.add(headerPanel);



        // Second Layer
        JPanel meepPanel = new JPanel(new FlowLayout(FlowLayout.CENTER)); // Use FlowLayout.CENTER
for center alignment

        JLabel subheading2 = new JLabel("Welcome");
        subheading2.setFont(new Font("Poppins", Font.ITALIC, 30));
```

```
meepPanel.add(subheading2);



// Set layout manager for the frame
frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));
frame.add(meepPanel);



Schedule schedule = new Schedule();
schedule.createScheduleFrame(frame.getContentPane());



frame.setSize(1000, 800); // Adjust the size as needed
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

ViewManager viewmanager = ViewManager.getInstance();
AppSystem appsys = AppSystem.getInstance();

//logout button action listener
viewmanager.attachListener(logoutBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(frame, false);
         frame.dispose();
         viewmanager.refreshLoginFrame();
         viewmanager.setVisibility(viewmanager.getLoginFrame().getFrame(), true);
```

```
            appsys.setCurrentID(0);
        }
    }
  );

  //home button action listener
  viewmanager.attachListener(homeBtn,
    new ActionListener()
    {
      @Override
      public void actionPerformed(ActionEvent event)
      {
        viewmanager.setVisibility(frame, false);
        frame.dispose();
        viewmanager.refreshNDashboardFrame();
        viewmanager.setVisibility(viewmanager.getNDashboardFrame().getFrame(), true);
      }
    }
  );
}
/**
 * Returns the nurses shifts frame
 * @return
 */
public JFrame getFrame()
{
  return frame;
}
```

```
/**
 * Inner class schedule that manages the display of the shifts.
 */
private class Schedule
{
   DefaultListModel<Shift> listModel = new DefaultListModel<>();
   JList<Shift> jList = new JList<>(listModel);

   public void createScheduleFrame(Container container) {

      displayShifts();



      jList.setCellRenderer(new MiniGrid());

      // Create a JScrollPane to allow scrolling if the list is too long
      JScrollPane scrollPane = new JScrollPane(jList);



      JButton addShiftBtn = new JButton("addShift");
      addShiftBtn.addActionListener(e -> addShift());

      filter.getSubmitBtn().addActionListener(e -> filterShifts());

      sort.getSubmitBtn().addActionListener(e -> sortShifts());

      JPanel westPanel = new JPanel();
      westPanel.setLayout(new BoxLayout(westPanel, BoxLayout.Y_AXIS));
      westPanel.add(filter);
```

```
        westPanel.add(sort);

        JPanel eastPanel = new JPanel();
        eastPanel.setLayout(new BoxLayout(eastPanel, BoxLayout.Y_AXIS));
        eastPanel.add(addShiftBtn);

        // Create a panel to hold the JList and the Remove Shift button
        JPanel mainPanel = new JPanel(new BorderLayout());
        mainPanel.add(westPanel,BorderLayout.WEST);
        mainPanel.add(scrollPane, BorderLayout.CENTER); // Place JList at the top
        mainPanel.add(eastPanel, BorderLayout.EAST); // Remove button on the bottom of th
        container.add(mainPanel);



    }
    /**
     * Customizes the format of the minigrid.
     */
    private class MiniGrid extends DefaultListCellRenderer
    {

        private static final int CELL_PADDING = 5;
        private static final int BORDER_THICKNESS = 1;
        private static final Color BORDER_COLOR = Color.BLACK;



        @Override
        public Component getListCellRendererComponent(JList<?> list, Object value, int index, boolean
isSelected, boolean cellHasFocus)
```

```
        {
            Shift shift = (Shift) value;

            JPanel panel = new JPanel(new GridLayout(4, 2, CELL_PADDING, CELL_PADDING));
            panel.setBackground(isSelected ? list.getSelectionBackground() : list.getBackground());
             //The Details of the panel

            panel.add(createLabel("Shift ID:"));
            panel.add(createLabel(Integer.toString(shift.getShiftID())));
            panel.add(createLabel("Date:"));
            panel.add(createLabel(shift.getShiftDateString()));
            panel.add(createLabel("Time:"));
            panel.add(createLabel(shift.getTimeString()));
            panel.add(createLabel("Hospital:"));
            panel.add(createLabel(shift.getHospital()));



            //The Border around the Panel
            panel.setBorder(BorderFactory.createCompoundBorder(
                    BorderFactory.createMatteBorder(0, 0, BORDER_THICKNESS, BORDER_THICKNESS,
BORDER_COLOR),
                    BorderFactory.createEmptyBorder(CELL_PADDING, CELL_PADDING, CELL_PADDING,
CELL_PADDING)
            ));
            return panel;
        }
        /**
        * Method to create a JLabel with specified text.
        * @param text Text for the label.
```

```
    * @return A JLabel with the given text.
    */
    private JLabel createLabel(String text)
    {
        JLabel label = new JLabel(text);
        label.setHorizontalAlignment(SwingConstants.LEFT);
        return label;
    }
}
/**
 * Populates the JList with available shifts.
 */
private void displayShifts()
{
    var shifts = appsys.getAvailableShifts();
    System.out.println("here");
    ShiftCollection shiftCollection = new ShiftCollection(shifts);

    ShiftIterator shiftIterator =shiftCollection.createIterator();
    while (shiftIterator.hasNext())
    {
        Shift shift = shiftIterator.next();
        listModel.addElement(shift);
    }
}
/**
* Adds the selected shift from the list.
*/
private void addShift()
```

```
    {
        int selectedIndex = jList.getSelectedIndex();
        if (selectedIndex != -1)
        {
            Shift selectedShift = listModel.get(selectedIndex);

            ShiftManager shiftManager = new ShiftManager();
            //shiftManager.reserveShift(AppSystem.getAvailableShifts(), AppSystem.getCurrentNurse(),
selectedShift); // Implement this method in NurseManager
            shiftManager.reserveShift(selectedShift);
            listModel.remove(selectedIndex);

        }

    }
    /**
     * Applies filter to the shift list based on the selected filter options.
    */
    private void filterShifts()
    {
        listModel.clear();

        var shifts = appsys.getAvailableShifts();

        if(filter.getNoneBtn().isSelected())
        {
            System.out.println("radio button none");
            shifts = appsys.getAvailableShifts();
```

```
}
else if(filter.getABtn().isSelected())
{
   System.out.println("radio button A");
   shifts = filter.filterShiftsByHospital(1);
}
else if(filter.getBBtn().isSelected())
{
   System.out.println("radio button B");
   shifts = filter.filterShiftsByHospital(2);
}
else if(filter.getCBtn().isSelected())
{
   System.out.println("radio button C");
   shifts = filter.filterShiftsByHospital(3);
}
else if(filter.getDBtn().isSelected())
{
   System.out.println("radio button D");
   shifts = filter.filterShiftsByHospital(4);
}
else if(filter.getDayBtn().isSelected())
{
   System.out.println("radio button day");
   shifts = filter.filterShiftsByType(1);
}
else if(filter.getNightBtn().isSelected())
{
   System.out.println("radio button night");
```

```
        shifts = filter.filterShiftsByType(2);
    }


    ShiftCollection shiftCollection = new ShiftCollection(shifts);

    ShiftIterator shiftIterator =shiftCollection.createIterator();
    while (shiftIterator.hasNext())
    {
        Shift shift = shiftIterator.next();
        listModel.addElement(shift);
    }
}
/**
* Sorts the shift list based on the selections.
*/
private void sortShifts()
{
    listModel.clear();

    var shifts = appsys.getAvailableShifts();

    if(sort.getHospitalOpt().isSelected())
    {
        sort.sortCollection(shifts, sort.getCompByShiftHospital());
    }
    else if(sort.getTimeOpt().isSelected())
    {
        sort.sortCollection(shifts, sort.getCompByShiftType());
```

```
        }
        else if(sort.getDateOpt().isSelected())
        {
           sort.sortCollection(shifts, sort.getCompByShiftDate());
        }

        ShiftCollection shiftCollection = new ShiftCollection(shifts);

        ShiftIterator shiftIterator =shiftCollection.createIterator();
        while (shiftIterator.hasNext())
        {
           Shift shift = shiftIterator.next();
           listModel.addElement(shift);
        }


      }
   }
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import java.util.ArrayList;
import java.io.File;
import java.io.FileOutputStream;
```

```
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.Serializable;

/**
 *Makes the nurse object.
 */
public class Nurse extends User implements Serializable{
   private  ArrayList<Shift> nurseSchedule = new ArrayList<>();
   private boolean serialized;
   /**
    * Constructor for the nurse object.
    * @param nurseID ID for the new nurse object.
    * @param name Name for the new nurse object.
    * @param password Password for the new nurse object.
    */
   public Nurse(int nurseID, String name, String password)
   {

      super(nurseID, name, password);



      try {
         this.serializeNurse(Integer.toString(getID()));
      } catch (IOException e) {
         // Handle exception or propagate it
         System.err.println("Serialization failed: " + e.getMessage());
      }
   }
```

```
/**
 * Returns the nurse's schedule.
 * @return ArrayList of nurse schedule.
 */
public ArrayList<Shift> getNurseSchedule()
{
   return this.nurseSchedule;
}
/**
 * Method to change the nurse's password.
 * @param newPassword for the password to be changed to.
 */
public void changePassword(String newPassword)
{
   this.password = newPassword;
}
/**
 * Method to reserialize the nurse object after a shift has been reserved.
 * @param shift that is added.
 */
public void reserveShift(Shift shift)
{
   this.nurseSchedule.add(shift);
   try {
      this.serializeNurse(Integer.toString(getID()));
      System.out.println("After reserve Shift");
   } catch (IOException e) {
      // Handle exception or propagate it
      System.err.println("Serialization failed: " + e.getMessage());
```

```
    }
    AppSystem appsys = AppSystem.getInstance();
    appsys.addNurse(this);
}
/**
 * Method that reserializes the nurse object after a shift has been canceled.
 * @param shift that will be canceled.
 */
public void cancelShift(Shift shift)
{
    this.nurseSchedule.remove(shift);
    try {
        this.serializeNurse(Integer.toString(getID()));
        System.out.println("After reserve Shift");
    } catch (IOException e) {
        // Handle exception or propagate it
        System.err.println("Serialization failed: " + e.getMessage());
    }

}
/**
 * Changes the format when printing a nurse object.
 * @return the string format of the nurse object.
 */
@Override
public String toString()
{
    return Integer.toString(getID()) + " " + getName() + " " + getPassword();
}
```

```
    /**
     * Serializes the nurse object after it has been created.
     * @param filename that the serialized file will be given
     * @throws IOException
     */
    public void serializeNurse(String filename) throws IOException {

        File directory = new File("nurses");
        if (!directory.exists()) {
            directory.mkdirs(); // Create the folder if it doesn't exist
        }

        // Combine the folder path and filename
        String fullPath = "nurses" + File.separator + filename;

        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(fullPath))) {
            this.serialized = true;
            out.writeObject(this);
        }
    }
}



package project;

/**
 * Interface that provides a method to create an iterator.
 */
public interface NurseAggregator {
```

```
    NurseIterator createIterator();
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;



import java.util.ArrayList;


/**
 * The NurseApp class serves as the entry point for the nurse scheduling app.
 */
public class NurseApp {
    /**
     * The main method initializes the application systems and demonstrates functionality.
     *
     * @param args
     */
    public static void main(String[] args)
    {
        AppSystem appsys = AppSystem.getInstance();
        ViewManager viewmanager = ViewManager.getInstance();


        viewmanager.refreshLoginFrame();


        System.out.println("Listing nurses");
```

```
        ArrayList<Nurse> nursesTest = appsys.getNurseList();
        for(Nurse nurse: nursesTest)
        {
            System.out.println(nurse.toString());
        }
        System.out.println("");

        System.out.println("Listing shifts");
        ArrayList<Shift> shiftsTest = appsys.getAvailableShifts();
        for(Shift shift: shiftsTest)
        {
            System.out.println(shift.toString());
        }

    }
}
package project;

import java.util.ArrayList;

/**
 *Implements the nurse iterator and holds a collection of objects.
 */
public class NurseCollection implements NurseAggregator {
    private ArrayList<Nurse> nurses;

    /**
     * Saves the ArrayList of nurses
     * @param nurses objects that will be saved to the array list.
```

```
     */
    public NurseCollection(ArrayList<Nurse> nurses) {
        this.nurses = nurses;
    }
    /**
     * Creates the nurse iterator.
     * @return The nurse iterator.
     */

    @Override
    public NurseIterator createIterator() {
        return new ConcreteNurseIterator(nurses);
    }
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Iterator;
import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.JButton;
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSeparator;
import java.util.ArrayList;

import javax.swing.JFrame;

/**
 *Class that implements the nurses frame where the admin sees all the nurses.
 * Implements the GUI functionality.
 */
public class NursesFrame {
    AppSystem appsys = AppSystem.getInstance();
    private JFrame frame = new JFrame("Nurses");
    private JButton homeBtn = new JButton("Home");
    private JButton logoutBtn = new JButton("Logout");

    private JButton deleteShiftsBtn = new JButton("Delete Nurse");

    private DefaultListModel<Nurse> nurseListModel = new DefaultListModel<>();
    private JList<Nurse> nurseList = new JList<>(nurseListModel);
    /**
     * Constructor for the nurse frame.
     */
    public NursesFrame()
    {

        JLabel heading = new JLabel("eNurse");
```

```
heading.setFont(new Font("Poppins", Font.BOLD, 20));
JLabel subheading = new JLabel("Admin Nurses Frame");
subheading.setFont(new Font("Poppins", Font.ITALIC, 10));

// Setup left panel for heading and subheading
JPanel leftPanel = new JPanel();
leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y_AXIS));
leftPanel.add(heading);
leftPanel.add(subheading);

// Setup top panel for buttons
JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
buttonPanel.add(homeBtn);
buttonPanel.add(logoutBtn);

// Header panel
JPanel headerPanel = new JPanel(new BorderLayout());
headerPanel.add(leftPanel, BorderLayout.WEST);
headerPanel.add(buttonPanel, BorderLayout.EAST);

// Setting up the nurse list
nurseList.setFont(new Font("Arial", Font.PLAIN, 18)); // Larger font for nurse list
nurseList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
JScrollPane listScrollPane = new JScrollPane(nurseList);
listScrollPane.setPreferredSize(new Dimension(300, 400)); // Adjust size as needed
nurseList.setCellRenderer(new DefaultListCellRenderer() {
    @Override
    public Component getListCellRendererComponent(JList<?> list, Object value, int index, boolean
isSelected, boolean cellHasFocus) {
```

```
            Component c = super.getListCellRendererComponent(list, value, index, isSelected,
cellHasFocus);
            if (c instanceof JLabel) {
                JLabel label = (JLabel) c;
                label.setBorder(BorderFactory.createMatteBorder(0, 0, 1, 0, Color.GRAY));
                label.setPreferredSize(new Dimension(label.getWidth(), 50)); // Increase cell height

            }
            return c;
        }
    });


    /////deleteShiftsBtn.addActionListener(e -> deleteSelectedNurse());
    JPanel deleteButtonPanel = new JPanel();
    deleteButtonPanel.add(deleteShiftsBtn);


    frame.setLayout(new BorderLayout());
    frame.add(headerPanel, BorderLayout.NORTH);
    frame.add(listScrollPane, BorderLayout.CENTER);
    frame.add(deleteButtonPanel, BorderLayout.SOUTH);

    // Display nurses
    displayNurses();

    frame.setSize(800, 800); // Adjust the size as needed
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
ViewManager viewmanager = ViewManager.getInstance();
AppSystem appsys = AppSystem.getInstance();

//logout button action listener
viewmanager.attachListener(logoutBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(frame, false);
         frame.dispose();
         viewmanager.refreshLoginFrame();
         viewmanager.setVisibility(viewmanager.getLoginFrame().getFrame(), true);
         appsys.setCurrentID(0);

      }
   }
);

//home button action listener
viewmanager.attachListener(homeBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         frame.dispose();
         viewmanager.setVisibility(frame, false);
```

```
            viewmanager.refreshADashboardFrame();
            viewmanager.setVisibility(viewmanager.getADashboardFrame().getFrame(), true);
        }
      }
   );

   //remove nurse action listener
   viewmanager.attachListener(deleteShiftsBtn,
      new ActionListener()
      {
         @Override
         public void actionPerformed(ActionEvent event)
         {
            viewmanager.getNursesFrame().deleteSelectedNurse();
         }
      }
   );

}
/**
 * Returns the Nurse Frame.
 * @return Nurse Frame
 */
public JFrame getFrame()
{
   return frame;
}

/**
```

```
     * Uses the iterator pattern to display the nurses.
     */
    private void displayNurses() {
        var nurses = appsys.getNurseList();
        NurseCollection nurseCollection = new NurseCollection(nurses);

        NurseIterator nurseIterator =nurseCollection.createIterator();
        while (nurseIterator.hasNext()) {
            Nurse nurse = nurseIterator.next();
            nurseListModel.addElement(nurse);
        }
    }
    /**
     * Deletes a selected nurse after the delete button is pressed.
     */
    private void deleteSelectedNurse() {
        int selectedIndex = nurseList.getSelectedIndex();
        if (selectedIndex != -1) {
            Nurse selectedNurse = nurseListModel.get(selectedIndex);
            appsys.removeNurseFile(selectedNurse); // Implement this method in NurseManager
            nurseListModel.remove(selectedIndex);
            appsys.deleteNurse(selectedNurse.getID());
    }
    }
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
```

```
 */
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Date;

/**
 *Makes the schedule frame where the nurses sees their schedule.
 * Implements the GUI functionality.
 */
public class ScheduleFrame {
   AppSystem appsys = AppSystem.getInstance();
   private JFrame frame = new JFrame("Schedule");
   private JButton homeBtn = new JButton("Home");
   private JButton logoutBtn = new JButton("Logout");
   private Sort sort = new Sort();

   /**
    * Constructor for the schedule frame.
    */
   public ScheduleFrame()
   {
      JLabel heading = new JLabel("eNurse");
      heading.setFont(new Font("Poppins", Font.BOLD, 20));
      JLabel subheading = new JLabel("Schedule Frame");
```

```
subheading.setFont(new Font("Poppins", Font.ITALIC, 10));



// Creating a subpanel for horizontal buttons on the right
JPanel rightPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
rightPanel.add(homeBtn);
rightPanel.add(logoutBtn);


JPanel leftPanel = new JPanel();
leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y_AXIS));
leftPanel.add(heading);
leftPanel.add(subheading);

// Set layout manager for the frame
frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));


// Creating a panel for the entire content with BorderLayout
JPanel headerPanel = new JPanel(new BorderLayout());
headerPanel.add(leftPanel, BorderLayout.WEST);
headerPanel.add(rightPanel, BorderLayout.EAST);
headerPanel.add(new JSeparator(), BorderLayout.SOUTH); // Horizontal Line

// Add the content panel to the frame
frame.add(headerPanel);
```

```
    // Second Layer
    JPanel meepPanel = new JPanel(new FlowLayout(FlowLayout.CENTER)); // Use FlowLayout.CENTER for center alignment

    JLabel subheading2 = new JLabel("Welcome");
    subheading2.setFont(new Font("Poppins", Font.ITALIC, 30));
    meepPanel.add(subheading2);


    // Set layout manager for the frame
    frame.setLayout(new BoxLayout(frame.getContentPane(), BoxLayout.Y_AXIS));
    frame.add(meepPanel);



    ////frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    Schedule schedule = new Schedule();
    schedule.createScheduleFrame(frame.getContentPane());


    frame.setSize(1000, 800); // Adjust the size as needed
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


    ViewManager viewmanager = ViewManager.getInstance();
    AppSystem appsys = AppSystem.getInstance();
```

```
//logout button action listener
viewmanager.attachListener(logoutBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(frame, false);
         frame.dispose();
         viewmanager.refreshLoginFrame();
         viewmanager.setVisibility(viewmanager.getLoginFrame().getFrame(), true);
         appsys.setCurrentID(0);

      }
   }
);

//home button action listener
viewmanager.attachListener(homeBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
      {
         viewmanager.setVisibility(frame, false);
         frame.dispose();
         viewmanager.refreshNDashboardFrame();
         viewmanager.setVisibility(viewmanager.getNDashboardFrame().getFrame(), true);
      }
```

```
        }
    );
}
/**
 * Returns the schedule frame.
 * @return schedule frame
 */
public JFrame getFrame()
{
    return frame;
}

/**
* Inner class Schedule manages the display and interactions with the shift schedule.
* It handles the creation of UI components for displaying, filtering, sorting, and
* removing shifts.
*/
public class Schedule
{
    DefaultListModel<Shift> listModel = new DefaultListModel<>();
    JList<Shift> jList = new JList<>(listModel);

    public void createScheduleFrame(Container container) {
        // Create an ArrayList to store data
        //ArrayList<Shift> dataList = new ArrayList<>();

        displayShifts();
```

```
jList.setCellRenderer(new MiniGrid());

// Create a JScrollPane to allow scrolling if the list is too long
JScrollPane scrollPane = new JScrollPane(jList);



JButton removeButton = new JButton("Remove Shift");
removeButton.addActionListener(e -> deleteSelectedShift());

ArrayList<Shift> shiftsList = (ArrayList<Shift>) appsys.getAvailableShifts();




JPanel westPanel = new JPanel();
westPanel.setLayout(new BoxLayout(westPanel, BoxLayout.Y_AXIS));
//westPanel.add(sortPanel);

JPanel eastPanel = new JPanel();
eastPanel.setLayout(new BoxLayout(eastPanel, BoxLayout.Y_AXIS));
eastPanel.add(removeButton);

// Create a panel to hold the JList and the Remove Shift button
JPanel mainPanel = new JPanel(new BorderLayout());
mainPanel.add(westPanel,BorderLayout.WEST);
mainPanel.add(scrollPane, BorderLayout.CENTER); // Place JList at the top
mainPanel.add(eastPanel, BorderLayout.EAST); // Remove button on the bottom of th
container.add(mainPanel);
```

```
    }
    /**
     * Inner class MiniGrid extends DefaultListCellRenderer to customize the rendering of list items.
    */
    private class MiniGrid extends DefaultListCellRenderer
    {

        private static final int CELL_PADDING = 5;
        private static final int BORDER_THICKNESS = 1;
        private static final Color BORDER_COLOR = Color.BLACK;



        @Override
        public Component getListCellRendererComponent(JList<?> list, Object value, int index, boolean
isSelected, boolean cellHasFocus)
        {
            Shift shift = (Shift) value;

            JPanel panel = new JPanel(new GridLayout(4, 2, CELL_PADDING, CELL_PADDING));
            panel.setBackground(isSelected ? list.getSelectionBackground() : list.getBackground());
             //The Details of the panel

            panel.add(createLabel("Shift ID:"));
            panel.add(createLabel(Integer.toString(shift.getShiftID())));
            panel.add(createLabel("Date:"));
            panel.add(createLabel(shift.getShiftDateString()));
            panel.add(createLabel("Time:"));
            panel.add(createLabel(shift.getTimeString()));
            panel.add(createLabel("Hospital:"));
```

```
            panel.add(createLabel(shift.getHospital()));



            //The Border around the Panel
            panel.setBorder(BorderFactory.createCompoundBorder(
                BorderFactory.createMatteBorder(0, 0, BORDER_THICKNESS, BORDER_THICKNESS,
BORDER_COLOR),
                BorderFactory.createEmptyBorder(CELL_PADDING, CELL_PADDING, CELL_PADDING,
CELL_PADDING)
            ));
            return panel;
        }


        private JLabel createLabel(String text)
        {
            JLabel label = new JLabel(text);
            label.setHorizontalAlignment(SwingConstants.LEFT);
            return label;
        }
    }
    /**
     * Populates the JList with available shifts.
     * Displays the shifts using the iterator.
     */
    public void displayShifts()
    {

        var shifts = appsys.getCurrentNurse().getNurseSchedule();
```

```
    sort.sortCollection(shifts, sort.getCompByShiftDate());



    ShiftCollection shiftCollection = new ShiftCollection(shifts);

    ShiftIterator shiftIterator =shiftCollection.createIterator();
    while (shiftIterator.hasNext())
    {
        Shift shift = shiftIterator.next();
        listModel.addElement(shift);
        System.out.print(shift.toString() + " kool");
    }



    for(Shift shift: shifts)
    {
        System.out.println(shift.toString());
    }
}
/**
* Deletes the selected shift from the list and system.
*/
private void deleteSelectedShift()
{

    int selectedIndex = jList.getSelectedIndex();
    if (selectedIndex != -1)
    {
        Shift selectedShift = listModel.get(selectedIndex);
```

```
            ShiftManager shiftManager = new ShiftManager();
            shiftManager.recreateShift(appsys.getAvailableShifts(), selectedShift); // Implement this method in NurseManager
            listModel.remove(selectedIndex);
            shiftManager.cancelReservation(selectedShift);


        }
      }



   }
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import java.util.Date;
import java.text.SimpleDateFormat;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.Serializable;

/**
 *Creates the shift object.
```

```
 */
public class Shift implements Serializable{
   private int shiftID;
   private boolean isDay;
   private String hospital;
   private Date shiftDate = new Date();
   /**
    * Constructor to create the shift object.
    * @param shiftID for the new shift objet
    * @param isDay whether the shift is during the day
    * @param hospital where the shift is
    * @param shiftDate when the shift is
    */
   public Shift(int shiftID, boolean isDay, String hospital, Date shiftDate)
   {
      this.shiftID = shiftID;
      this.isDay = isDay;
      this.hospital = hospital.toLowerCase();
      this.shiftDate = shiftDate;

      try {
         this.serializeShift(Integer.toString(this.shiftID));
      } catch (IOException e) {
         // Handle exception or propagate it
         System.err.println("Serialization failed: " + e.getMessage());
      }
   }
   /**
    * Returns the shift ID.
```

```
 * @return Shift ID
 */
public int getShiftID()
{
   return shiftID;
}
/**
 * Returns whether the shift is during the day.
 * @return Boolean, true for day, false otherwise
 */
public boolean getIsDay()
{
   return isDay;
}
/**
 * Returns the hospital where the shift takes place.
 * @return string of the hospital
 */
public String getHospital()
{
   return hospital;
}
/**
 * Returns the date of the shift.
 * @return Date object with the shift date.
 */
public Date getShiftDate()
{
   return shiftDate;
```

```
}

/**
 * Serializes the shift object.
 * @param filename that the serialized object will have
 * @throws IOException
 */
public void serializeShift(String filename) throws IOException {

    File directory = new File("shifts");
    if (!directory.exists()) {
        directory.mkdirs(); // Create the folder if it doesn't exist
    }

    // Combine the folder path and filename
    String fullPath = "shifts" + File.separator + filename;

    try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(fullPath))) {
        out.writeObject(this);
    }
}
/**
 * Formats the shift object from when its printed.
 * @return The formatted shift object.
 */
public String getShiftDateString()
{
    SimpleDateFormat formatter = new SimpleDateFormat("MM-dd-yyyy");
    return (formatter.format(getShiftDate()));
```

```
    }
    /**
     * Gets the time when the shift takes place.
     * @return string with time
     */
    public String getTimeString()
    {
        if (this.getIsDay()==true)
        {
            return "7am-7pm";
        }
        else{
            return "7pm-7am";
        }
    }
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import java.util.ArrayList;

/**
 *Implements the shift iterator and holds a collection of objects.
 */
public class ShiftCollection implements ShiftAggregator{
```

```
    private ArrayList<Shift> shifts;
    /**
     * Saves the ArrayList of shifts
     * @param shift objects that will be saved to the array list.
     */
    public ShiftCollection(ArrayList<Shift> shifts) {
        this.shifts = shifts;
    }
    /**
     * Creates the shift iterator.
     * @return The shift iterator.
     */
    @Override
    public ShiftIterator createIterator() {
        return new ConcreteShiftIterator(shifts);
    }
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Date;
```

```
/**
 * Implements the Shift Creation Frame.
 * Implements the GUI functionality.
 */
public class ShiftCreationFrame {
    private JFrame frame = new JFrame("Create Shift");

    private JButton noChangesBtn = new JButton("Go Back/No Changes");
    private JButton createShiftBtn = new JButton("Create Shift");
    private JTextField id = new JTextField(10);
    private JTextField type = new JTextField(10);
    private JTextField hospital = new JTextField(10);
    private JTextField dateD = new JTextField(3);
    private JTextField dateM = new JTextField(3);
    private JTextField dateY = new JTextField(3);
    /**
     * Constructor for the shift creation frame.
     */
    public ShiftCreationFrame()
    {
        //Creating JLabel
        JLabel l1 = new JLabel("Shift ID");
        JLabel l2 = new JLabel("Shift Type");
        JLabel l3 = new JLabel("Hospital");
        JLabel l4 = new JLabel("Date");
```

```
l1.setLabelFor(id);

l2.setLabelFor(type);

l3.setLabelFor(hospital);

l4.setLabelFor(dateD);

// Creating JLabel for the heading
JLabel heading = new JLabel("Create New Shift");
heading.setFont(new Font("Poppins", Font.BOLD, 20));
heading.setBounds(75, 50, 300, 30);

// Creating JButton
createShiftBtn.setFont(new Font("Poppins", Font.ITALIC, 10));
createShiftBtn.setBounds(50, 325, 200, 30);
frame.add(createShiftBtn);

noChangesBtn.setFont(new Font("Poppins", Font.ITALIC, 10));
noChangesBtn.setBounds(50, 375, 200, 30);
frame.add(noChangesBtn);



/* This method specifies the location and size
 * of any component = > setBounds(x, y, width, height)
 * where x & y are cordinates from the top left
 * corner and remaining two parameters are the width
```

```
 * and height of the specific component.
 */
l1.setBounds(50, 100, 100, 30);
id.setBounds(175,100,170,30);

l2.setBounds(50, 150, 100, 30);
type.setBounds(175,150,170,30);

l3.setBounds(50, 200, 100, 30);
hospital.setBounds(175,200,170,30);

l4.setBounds(50, 250, 100, 30);
dateD.setBounds(175,250,30,30);
dateM.setBounds(235,250,30,30);
dateY.setBounds(295,250,50,30);

JLabel dateLabelD = new JLabel("DD");
dateLabelD.setFont(new Font("Poppins", Font.ITALIC, 10));
dateLabelD.setBounds(180, 275, 30, 30);

JLabel dateLabelM = new JLabel("MM");
dateLabelM.setFont(new Font("Poppins", Font.ITALIC, 10));
dateLabelM.setBounds(240, 275, 30, 30);

JLabel dateLabelY = new JLabel("YYYY");
dateLabelY.setFont(new Font("Poppins", Font.ITALIC, 10));
dateLabelY.setBounds(303, 275, 30, 30);
```

```
/* changing appearance of the label
 * Font,text color,background color
 */
l1.setFont(new Font("Poppins", Font.BOLD, 15));
id.setFont(new Font("Poppins", Font.ITALIC, 15));

l2.setFont(new Font("Poppins", Font.BOLD, 15));
type.setFont(new Font("Poppins", Font.ITALIC, 15));

l3.setFont(new Font("Poppins", Font.BOLD, 15));
hospital.setFont(new Font("Poppins", Font.ITALIC, 15));

l4.setFont(new Font("Poppins", Font.BOLD, 16));
dateD.setFont(new Font("Poppins", Font.ITALIC, 16));
dateM.setFont(new Font("Poppins", Font.ITALIC, 16));
dateY.setFont(new Font("Poppins", Font.ITALIC, 16));


//adds the labels to the frame
frame.add(l1);
frame.add(id);
frame.add(l2);
frame.add(type);
frame.add(l3);
frame.add(hospital);
frame.add(l4);
frame.add(dateD);
frame.add(dateM);
```

```
frame.add(dateY);
frame.add(heading);
frame.add(dateLabelD);
frame.add(dateLabelM);
frame.add(dateLabelY);


/* layout set to null ->
 * as no layout managers used
 * in this example (like FlowLayout,BoxLayout.etc)
 */
frame.setLayout(null);


//This method sets the width and height of the frame
frame.setSize(400, 500);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


ViewManager viewmanager = ViewManager.getInstance();
AppSystem appsys = AppSystem.getInstance();
ShiftManager shiftmanager = new ShiftManager();

//go back button action listener
viewmanager.attachListener(noChangesBtn,
   new ActionListener()
   {
      @Override
      public void actionPerformed(ActionEvent event)
```

```
        {
            viewmanager.setVisibility(frame, false);
            viewmanager.setVisibility(viewmanager.getADashboardFrame().getFrame(), true);
            appsys.setCurrentID(0);

        }
    }
);

//create shift button action listener
viewmanager.attachListener(createShiftBtn,
    new ActionListener()
    {
        @Override
        public void actionPerformed(ActionEvent event)
        {

            Integer ID = Integer.valueOf(id.getText());

            boolean isday = true;
            String Type = type.getText().toLowerCase();
            if(Type.equals("night"))
            {
                isday = false;
            }

            String hosp = hospital.getText().toLowerCase();

            Integer D = Integer.valueOf(dateD.getText());
```

```
                Integer M = Integer.valueOf(dateM.getText());
                Integer Y = Integer.valueOf(dateY.getText());
                Date date = new Date(Y-1900,M-1,D);




                shiftmanager.createShift(appsys.getAvailableShifts(), new Shift(ID,isday,hosp,date));
                JOptionPane.showMessageDialog(frame, "Shift successfully created.", "Success",
JOptionPane.INFORMATION_MESSAGE);


                viewmanager.setVisibility(frame, false);
                frame.dispose();
                viewmanager.refreshADashboardFrame();
                viewmanager.setVisibility(viewmanager.getADashboardFrame().getFrame(), true);


                appsys.setCurrentID(0);


            }
        }
    );
  }
  /**
   * Returns the shift creation frame.
   * @return shift creation frame.
   */
  public JFrame getFrame()
  {
    return frame;
  }
```

```
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to edit this template
 */
package project;

/**
 *Interface for the shift iterator.
 */
public interface ShiftIterator {
    boolean hasNext();
    Shift next();
    boolean isDone();
}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
import static project.AppSystem.removeShiftFile;

/**
 *Class that manages the shifts.
 */
public class ShiftManager
{
    private Date date = new Date();
    AppSystem appsys = AppSystem.getInstance();
    /**
     * Constructor for the shift manager.
     */
    public ShiftManager()
    {

    }
    /**
     * Method that creates the shift.
     * @param availableShifts ArrayList of the available shifts.
     * @param shift New shift object that will be added.
     */
    public void createShift(ArrayList<Shift> availableShifts, Shift shift)
    {
        availableShifts.add(shift);
    }

    /**
     * Method that reserves shift.
     * @param shift that will be reserved.
```

```
 */
public void reserveShift( Shift shift)
{

   Nurse currentNurse = appsys.getCurrentNurse();
   appsys.removeNurseFile(currentNurse);
   currentNurse.reserveShift(shift);
   appsys.setCurrentID(currentNurse.getID());
   appsys.removeShiftFile(shift);




}
/**
 * Method that cancels a shift on the nurse's schedule
 * @param shift that is being canceled
 */
public void cancelReservation(Shift shift)
{

   appsys.getCurrentNurse().cancelShift(shift);
}



/**
 * Method that deletes a shift from the available shifts array list
 * @param availableShifts
 * @param shiftID
 */
```

```
public void deleteShift(ArrayList<Shift> availableShifts, int shiftID)
{
   for (Shift availableShift : availableShifts)
   {
      if(availableShift.getShiftID() == shiftID)
      {
         availableShifts.remove(availableShift);
      }
   }
}
/**
 * Method to recreate a shift after nurse deletes it from schedule.
 * @param availableShifts
 * @param shift
 */
public void recreateShift(ArrayList<Shift> availableShifts, Shift shift)
{
   availableShifts.add(shift);
   try {
      shift.serializeShift(Integer.toString(shift.getShiftID()));
   } catch (IOException ex) {
      Logger.getLogger(ShiftManager.class.getName()).log(Level.SEVERE, null, ex);
   }
}
/**
 * Method to add shift to available shifts.
 * @param shift
 */
public void addShift(Shift shift)
```

```
    {
        appsys.getAvailableShifts().add(shift);
    }
    /**
     * Method to delete shift from available shifts and delete the file.
     * @param shift
     */
    public void removeShift(Shift shift)
    {
        appsys.getAvailableShifts().remove(shift);
        removeShiftFile(shift);
    }

}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import java.util.Comparator;
import java.util.Collections;
import java.util.ArrayList;
import javax.swing.*;

/**
 *Provides a UI to sort the shifts.
 */
```

```
public class Sort extends JPanel {
    private JLabel titleLabel = new JLabel("Sorting");
    private JRadioButton hospitalOpt = new  JRadioButton("Hospital");
    private JRadioButton timeOpt = new JRadioButton("Time");
    private JRadioButton dateOpt = new JRadioButton("Date");
    private JButton submitBtn = new JButton("Sort");
    /**
     * Constructor for the sort class that deals with the GUI and functionalities.
     */
    public Sort()
    {
        int topPadding = 50;
        this.setBorder(BorderFactory.createEmptyBorder(topPadding,0,0,0));
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));

        ButtonGroup sortGroup = new ButtonGroup();

        sortGroup.add(timeOpt);
        sortGroup.add(hospitalOpt);
        sortGroup.add(dateOpt);

        this.add(titleLabel);
        this.add(hospitalOpt);
        this.add(timeOpt);
        this.add(dateOpt);

        this.add(submitBtn);
    }
    /**
```

```
 * Returns a comparator to compare the shifts by date.
 * @return
 */
public Comparator<Shift> getCompByShiftDate() {
   return new Comparator<Shift>() {
      @Override
      public int compare(Shift shift1, Shift shift2) {
         return shift1.getShiftDate().compareTo(shift2.getShiftDate());
      }
   };
}
/**
 * Returns a comparator to compare the shifts by hospital.
 * @return
 */
public Comparator<Shift> getCompByShiftHospital() {
   return new Comparator<Shift>() {
      @Override
      public int compare(Shift shift1, Shift shift2) {
         return shift1.getHospital().compareTo(shift2.getHospital());
      }
   };
}
/**
 * Returns a comparator to compare the shifts by shift type.
 * @return
 */
public Comparator<Shift> getCompByShiftType() {
   return new Comparator<Shift>() {
```

```
        @Override
        public int compare(Shift shift1, Shift shift2) {
           String s1;
           String s2;
           if (shift1.getIsDay() == true)
           {
              s1 = "true";
           }
           else
           {
              s1 = "false";
           }
           if (shift2.getIsDay() == true)
           {
              s2 = "true";
           }
           else
           {
              s2 = "false";
           }
           return s1.compareTo(s2);
        }
     };
}
/**
 * Method to pass in the array list that needs to be sorted.
 * @param shifts
 * @param comp
 */
```

```
public void sortCollection(ArrayList<Shift> shifts, Comparator<Shift> comp)
{
   Collections.sort(shifts, comp);
}
/**
 * Returns button for hospital option.
 * @return
 */
public JRadioButton getHospitalOpt()
{
   return hospitalOpt;
}
/**
 * returns button for time option
 * @return
 */
public JRadioButton getTimeOpt()
{
   return timeOpt;
}
/**
 * returns button for get date option
 * @return
 */
public JRadioButton getDateOpt()
{
   return dateOpt;
}
/**
```

```
     * returns button for submit to sort
     * @return
     */
    public JButton getSubmitBtn()
    {
        return submitBtn;
    }

}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import java.io.Serializable;

/**
 *Abstract class user that is used for admin and nurse.
 */
public abstract class User implements Serializable {
    private int id;
    private String name;
    protected String password;
    /**
     * Constructor for the user abstract class .
     * @param id
     * @param name
```

```
 * @param password
 */
public User(int id, String name, String password)
{
   this.id = id;
   this.name = name;
   this.password = password;
}
/**
 * Returns user ID
 * @return
 */
public int getID()
{
   return id;
}
/**
 * Returns user name
 * @return
 */
public String getName()
{
   return name;
}
/**
 * Returns user password.
 * @return
 */
public String getPassword()
```

```
    {
        return password;
    }

}

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package project;

import javax.swing.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;



/**
 * The ViewManager class is a singleton that manages the different frames in the application.
 * It offers methods to get and refresh these frames, change their visibility, and attach action listeners to buttons.
 */
public class ViewManager {

    private static ViewManager instance = new ViewManager();

    private LoginFrame loginframe;
    private NDashboardFrame ndashboard;
    private ADashboardFrame adashboard;
```

```
private NShiftsFrame nshiftsframe;
private AShiftsFrame ashiftsframe;
private ScheduleFrame scheduleframe;
private ChangePasswordFrame changepasswordframe;
private NursesFrame nursesframe;
private ShiftCreationFrame shiftcreationframe;
private AddNurseFrame addnurseframe;
/**
 * Private constructor for singleton pattern.
 */
private ViewManager()
{
}
/**
 * Creates a single instance of view manager.
 * @return
 */

public static ViewManager getInstance()
{
   if(instance == null)
   {
      instance = new ViewManager();
   }
   return instance;
}
/**
 * sets the visibility of a frame depending on the boolean.
 * @param frame
```

```
 * @param bool
 */
public void setVisibility(JFrame frame, boolean bool)
{
   frame.setVisible(bool);
}
/**
 * Attaches action listener to the button.
 * @param btn
 * @param listener
 */

public void attachListener(JButton btn, ActionListener listener)
{
   btn.addActionListener(listener);
}

/**
 * Returns the login frame.
 * @return
 */
public LoginFrame getLoginFrame()
{
   return loginframe;
}
/**
 * Returns the NDashboardframe
 * @return
 */
```

```
public NDashboardFrame getNDashboardFrame()
{
   return ndashboard;
}
/**
 * Returns the ADashboardFrame
 * @return
 */
public ADashboardFrame getADashboardFrame()
{
   return adashboard;
}
/**
 * Returns the NShifts Frame.
 * @return
 */
public NShiftsFrame getNShiftsFrame()
{
   return nshiftsframe;
}
/**
 * Returns the AShiftsFrame
 * @return
 */
public AShiftsFrame getAShiftsFrame()
{
   return ashiftsframe;
}
/**
```

```
 * Returns the schedule frame
 * @return
 */

public ScheduleFrame getScheduleFrame()
{
   return scheduleframe;
}
/**
 * Returns the change password frame.
 * @return
 */
public ChangePasswordFrame getChangePasswordFrame()
{
   return changepasswordframe;
}
/**
 * returns the nurses frame.
 * @return
 */
public NursesFrame getNursesFrame()
{
   return nursesframe;
}
/**
 * Returns the shift creation frame.
 * @return
 */
public ShiftCreationFrame getShiftCreationFrame()
```

```
{
   return shiftcreationframe;
}
/**
 * returns the add nurse frame
 * @return
 */

public AddNurseFrame getAddNurseFrame()
{
   return addnurseframe;
}



/**
 * refreshes the login frame
 */
public void refreshLoginFrame()
{
   loginframe = new LoginFrame();
}
/**
 * refreshes the NDashbaordFrame
 */
public void refreshNDashboardFrame()
{
   ndashboard = new NDashboardFrame();
}
/**
```

```
 * Refreshes the ADashboardFrame
 */
public void refreshADashboardFrame()
{
   adashboard = new ADashboardFrame();
}
/**
 * Refreshes the NShiftsFrame
 */
public void refreshNShiftsFrame()
{
   nshiftsframe = new NShiftsFrame();
}
/**
 * Refreshes the AShiftsFrame
 */
public void refreshAShiftsFrame()
{
   ashiftsframe = new AShiftsFrame();
}
/**
 * Refreshes the schedule frame
 */
public void refreshScheduleFrame()
{
   scheduleframe = new ScheduleFrame();
}
/**
 * Refreshes the change password frame
```

```
 */
public void refreshChangePasswordFrame()
{
   changepasswordframe = new ChangePasswordFrame();
}
/**
 * refreshes the nurses frame
 */
public void refreshNursesFrame()
{
   nursesframe = new NursesFrame();
}
/**
 * refreshes the shift creation frame
 */
public void refreshShiftCreationFrame()
{
   shiftcreationframe = new ShiftCreationFrame();
}
/**
 * refreshes the add nurse frame
 */
public void refreshAddNurseFrame()
{
   addnurseframe = new AddNurseFrame();
}
```

```
}
```