Object Oriented Programming Final Project Report

SchedNara

BY

Name : Ella Raputri

Student ID : 2702298154

Class : L2AC

Computer Science Program

School Of Computing and Creative Arts

Bina Nusantara International University

Jakarta

2024

Table of Contents

SCHEDNARA

A. Introduction

1. Background

As a part of our Object-Oriented Programming class that is conducted using the Java programming language, we are expected to create a project that should solve a small but interesting problem in our daily life. The project at the bare minimum should focus on the topic that we have covered in class, which is related to Object Oriented Programming, so that we can learn new things while creating this project.

Hence, after some thought, I decided to initiate a project that I have planned to do since my first semester holiday, which is a workflow and scheduling app. The reason why I planned to create this application is closely connected to my jobs in one of the organizations that I have joined, which is KTOM (Kontes Terbuka Olimpiade Matematika).

KTOM is an organization that holds a monthly free Math Olympiad contest to help Indonesian people learn and practice high school Math Olympiad problems. One of my jobs there is to help the committee arrange the monthly schedule based on KTOM's workflow.

For example, let the contest day be D-day, then 21 days before D-day (D-21) until 15 days before D-day (D-15) is the problem creation date range. After that, on D-14, the head of the problem setters will hand over the problems to the QC division, and so on. Then, if D-day is on June 14th, then the problem setters have to start create the problems for the contest from May 24th until May 30th. The head of them will hand the problem to the QC division on May 31st. One of my jobs is to help remind each division of a particular date based on the workflow. In this example, I have to remind the problem setter division on May 24th and May 30th, while for the division head and the QC division, I have to remind him on May 31st.

Every month has different dates, but the workflow is still the same, so it is tiresome to count and set the date manually every month. Besides that, several kinds of contests have different flows or checkpoints. Before this, I have created a similar function terminal program to help me with this problem when I was in my first semester. However, the

functions are not sufficient enough for my job now. So, I thought of creating a GUI app with more features that can help me with my job in arranging workflows and schedules.

Besides that, to make the app look more fun and intriguing, I designed the app based on my favorite NPCs in Genshin Impact, which is the Aranaras. I also created an Aranara menu to let the user interact with them without wasting too much time (because the main point of this app is to help arrange schedules) with the cute Aranara to help relieve the pressure in our daily lives.

2. Problem Identification

The first thing that I thought of in this Object-Oriented Programming Final Project is to make an app that can help increase my work efficiency, so that I can use it frequently even after the final project is submitted. That is why I was motivated to upgrade the previously terminal-based counting date program to a GUI program with more features.

User needs to sign up or log in to their account first so that they can access the tasks and workflows specified to them. Although this program will be only used by me in the future, in my opinion, it is beneficial to separate the workflows and tasks for individual matters and works.

After that, there are four main menus that users can access, which are the homepage, the add workflow menu, the calendar, and the Aranara menu. The main function of this program lies in the second and third menus, which are the add workflow and the calendar menu. In the add workflow menu, users can create, delete, and edit workflows. Meanwhile, in the calendar menu, users can create, delete, and edit tasks and also insert tasks based on specific workflow into the calendar. The homepage only functions as a landing page and contains user task completion streaks and some random quotes. Meanwhile, the Aranara menu allows users to interact with particular Aranara. User can also ask the Aranara about their task, weather, etc.

So, my main objectives for creating this program are:

a. Help user to arrange their workflows.
b. Let user insert tasks based on their workflow into their schedule.
c. Help user arrange their tasks.
d. Relieve user stress or motivate users by providing interaction with the Aranaras.

B. Project Specification

1. Program Name

"SchedNara"

SchedNara is an abbreviation of the Schedule of Nara. Schedule refers to my main goal in creating this app which is to assist in arranging schedules based on the tasks and workflows. Meanwhile, Nara is what the Aranaras in Genshin call humans. So, SchedNara is the schedule of humans, or to be exact, the schedule of the user.

2. Program Display

The size for all the mainframes in this program is 1280 px × 720 px. The background pictures of all frames are created by me in Figma using shapes, pen, and pencil tools provided there. The icons are also all imported from Figma using the Iconify plugins that are available there. The only pictures that are imported from other sources are the Aranara pictures. Reference and pictures used in this program will be further discussed in part C1.

In total, this program has seven different backgrounds. However, some of them are similar and are different because of certain elements that are added to the picture. If we remove that certain element, then some frames are basically the same. So, the frames that have drastic differences are only four, which are the welcome page frame, the default main menu frame, the edit workflow frame, and the edit Aranara frame. Besides four of them, several mini frames only have white color as the background.

The welcome page frame consists of a hillside scenery with blue sky, white clouds, and green grass. This background can be viewed on the welcome page, the login, and the signup page. The default main menu frame is a standard white frame with a blue rectangle on the left. Inside the blue rectangle are the icons or buttons to direct the user to other main menus or log out. Both the welcome page and default main menu background have the SchedNara logo. For the welcome page, it is positioned on the top, while for the default main menus, it is positioned on the bottom left of the frame.

As its name suggests, the edit workflow background is used only for the edit workflow frame, where the user can add, update, or delete the flows or checkpoints in a certain workflow. Lastly, the edit Aranara frame which is the main frame for the user when interacting with the Aranara also has its unique background. The background is in a living

room with a window, wooden floor, and some decorations. The Aranara will stand on a round carpet.

This program also utilizes the custom Cloneable Panel in an abundant amount. The Cloneable Panel is a JPanel that can be cloned based on the user input. These cloneable panels can be seen on things that the user can perform CRUD (create, read, update, delete) with, for example: workflows, flows, tasks, etc.

3. Program Features

The program has several main menus. Each main menu or main page has its special features. Below are the explanations of all menus and their features in SchedNara.

a. Welcome Page, Login, and Sign Up pages

The welcome page only consists of two buttons, which prompt the user to log in to their existing account or sign up for a new account. If the user chooses Login, then the user only needs to fill in their username and password. Meanwhile, if the user chooses Sign Up, the user needs to also enter their username and password, but they need to also confirm it. Password needs to be in the length of 8 and consists of letters and numbers. Both the Login and Sign Up pages also have validation that can change the field that is not correct yet to be colored in red. After the user presses the submit button and all fields are correct, the user will be directed to the homepage.

b. Home Page

On the homepage, user can view their task completion rate or streaks in the form of a bar chart. The bar chart will be reset weekly. There is also the text that tells the user about their completion rate for today. Another thing on the homepage is the random quote generator. All the quotes are from Genshin characters, including the Aranaras. Some buttons can redirect you to other main menus or log out.

c. Add Workflow Menu

The add workflow menu consists of several important features, from adding workflow, editing the name of the workflow, and deleting workflow. Users can also

search the workflow based on its name. The add workflow menu can also redirect the user to the edit workflow menu to edit the flows or checkpoints for each workflow.

d. Edit Workflow Menu

The edit workflow menu consists of a white panel that consists of fields related to a workflow (name, type, day, notes, color), a save button, and a delete button on the left side. Meanwhile, on the right side, there is a scroll pane for all the flows that the workflow contains. If the user selects the flow inside the scroll pane, then the white panel on the left will automatically be filled with the information of the selected flow. Users can edit or delete that flow from the white panel. If the user does not select any flow, then the delete button is disabled and only the save button is enabled. If the user clicks that save button and already fills in all the information, then a new flow will be added. Besides that, there is also a generate workflow to text button. Users can input a specific date as D-day and the program will generate the date of each flow based on D-day. The date generated are also copiable through a button provided there.

e. Calendar Menu

When first accessing this page, the user can see a calendar with a size of about 1080 × 540 pixels. Users can use the < and > arrow to move around to see other months. The date of today is marked differently than others, while Sundays are marked red. Dates from other months are colored lighter than the dates in the current month. Below the date are dots that represent the color of the tasks on that day. On the right side of the calendar, there is a task panel that contains the tasks for the selected date. There is also an add task button that enables the user to add a new task. For every task that is displayed in the scroll pane in the task panel, there is also the edit button and a checkmark box. The edit button allows users to edit specific tasks. Meanwhile, the checkmark box can be ticked by clicking on it and it is used to help users know if the task is completed or not. Above the calendar are the WhatsApp button and the Insert Workflow Tasks button. Similar to the generate workflow to text button, the insert workflow tasks will prompt the user to input a date as the D-day and then it will insert all the flows as tasks into the calendar. The WhatsApp button is created to help me

with my jobs to remind people. It will direct the user to WhatsApp chat based on the number inputted and type the message in the chat based on the message template available in SchedNara. The message template and the contacts can also be edited through the menu directed by this button.

f. Aranara Menu

The Aranara menu contains three different panels which lead to three different Aranara, which are Arama, Ararycan, and Arabalika. Three of them have similar interaction dialogues. Only for some dialogues that each of them is unique. Arama is the most cheerful one, Ararycan is friendly but kind of pessimistic, and Arabalika is antagonistic with his signature “Hmph”. Arama is the default Aranara for every user. The user has to reach a total affection of 60 to unlock Ararycan and 120 to unlock Arabalika.

g. Edit Aranara Menu

Edit Aranara Menu or Aranara Activity Page is where the user interacts with a specific Aranara. On the right side, there is the set default and pat button. All users default Aranara when signing up as Arama. They can switch to another Aranara by using the set default button. Every Aranara also has different background music, so if the user switches to another Aranara, then the background music will be changed. The pat button is used to pat the Aranara, every user is limited only to a total of three times patting for a day. When patting, there will be also specific dialogue from the Aranara. The chat button on the left side is used to open the Aranara Chat Menu on top of the Edit Aranara Menu.

h. Aranara Chat Menu

The Aranara Chat Menu can be closed using a left arrow button on the Edit Aranara Menu which is only visible when the Aranara Chat Menu opens. Meanwhile, Aranara Chat Menu itself has eight buttons inside it. All of the eight buttons can not increase the Aranara affection towards the user.

- The ‘Greet’ button, is used to greet or say hi to the Aranara. Each Aranara will reply to your greetings differently.
- The ‘Tasks’ button, is used to ask the Aranara about the task in today, tomorrow, or other dates. The Aranara will tell the user the task names on that particular date.
- The ‘Word’ button, is used to open Microsoft Word.
- The ‘Excel’ button, is used to open Microsoft Excel.
- The ‘PowerPoint’ button, is used to open Microsoft PowerPoint.
- The ‘Timer’ button, is used to set a timer and remind the user when the time is up.
- The ‘Weather’ button, is used to ask the Aranara about the weather today in a specific city. The Aranara will tell the user the temperature, the weather description, and some friendly recommendations regarding the weather conditions.
- The ‘Game’ button is used to play the simple rock, paper, scissors game with the Aranara.

4. Program Input

a. Mouse’s left button, to click the buttons inside the program.
b. Keyboard key ‘Enter’, which can be used when filling out the information on Sign Up and Login page. Clicking the field is also allowed here besides the ‘Enter’ key.
c. Other keyboard keys, to input the information needed in all fields and text areas inside the program.

5. Program Multimedia Output

Besides the text output, which is the tasks, workflows, affection rate, etc., there is also multimedia output which is the image of the frame, icons, and buttons in this program. All of the images can be accessed in the ‘src/App/img’ folder. Meanwhile, the audio files (background music and sound effects) are available in the ‘src/App/sound’ folder with the specifications below.

a. Background music
   - EnchantingBedtimeStories.wav for Welcome Page, Login, and Sign Up.
   - MelodyofHiddenSeeds.wav for the main menu background music if the default Aranara is Arama.

- IveNeverForgotten.wav for the main menu background music if the default Aranara is Ararycan.
- ForRiddlesForWonder.wav for the main menu background music if the default Aranara is Arabalika.

b. Sound Effects

- win.wav, when the user wins the rock paper scissors game.
- select.wav, when the user clicks a button in the Edit Aranara menu or Aranara Chat Menu, except for the chat button.
- lose.wav, when the user loses the rock paper scissors game.
- draw.wav, when the rock paper scissors game ends in a draw.
- chat.wav, when the user clicks the chat button or the button to collapse the Aranara Chat Menu.

6. Imported Java Packages

a. AWT and Swing

AWT and Swing are two APIs that are used to develop the GUI (Graphical User Interface) of the application. These two APIs are used to construct the display of this program.

b. SQL

The java.sql is a package that provides a method for accessing and processing data stored in the SQL database. This program stores user, workflow, task, contact, message template, and other information in the MySQL database. Hence, to access and update it based on user input, this program has to utilize the java.sql.

c. Text

The java.text is a package that provides classes and interfaces for handling text, date, and strings. In this program, this text package is mainly used to format and parse dates (task dates).

d. Time

Java Date and Time API contains classes and an interface that handles date and time values and is available in java.time package. This program is highly dependent on counting dates because it is a scheduling app. This package provides methods that help me in counting and handling dates and times (mostly used in task dates).

e. Util

The java,util package is a package that contains collections framework, date and time facilities, random generator, and many more. This program heavily relies on the usage of Linked List to store the list of tasks, workflows, etc. Besides that, this program also uses HashMap to store specific months and their corresponding number and also the random generator to generate Aranara dialogue randomly. Linked List, HashMap, and Random are all from the java.util package.

f. io

The java.io package contains classes and interfaces that help users with input and output-related operations. One of its main features is for reading and writing data to files. This program needs to read a txt file that contains the quotes so that it can display it on the homepage. To do so, it will utilize a class from the java.io package.

g. Net

The java.net package is used to establish a communication link between devices over the internet. In other words, it is used to implement networking applications. In this program, when user wants to contact others through WhatsApp, they will be redirected to the WhatsApp web on the internet. Hence, to achieve this, this program needs the java.net package methods.

h. Sound

The javax.sound is a collection of classes and interfaces for controlling and handling sound media in Java. In this program, there are several different background

music and sound effects which are all audio files. Therefore, we need the javax.sound to help us to handle them.

7. Other Libraries and API

a. Absolute Layout.jar

The Absolute Layout is used to ease the process of designing in Java Netbeans so that we can specify the exact location (x,y) of the children in the frame. When designing, Absolute Layout also allows the designer frame in the Netbeans to not enlarge its size automatically (most of the time).

b. Mysqlconnector.jar

This jar file is used to help connect the program with the existing MySQL local database so that the program can access and process the data stored there. Another advantage of using the database is that when we close the program, we will not lose the data that we inputted before so we do not need to restart everything when we start our program again.

c. jcommon.jar and jfreechart.jar

The JFreeChart is a Java class library that is used to handle and create charts. In this program, I used to create the bar chart on the homepage based on the user's task completion rate. The JFreeChart itself utilizes the JCommon class library which is a collection of useful classes for text utilities, displaying information, custom layout managers, etc.

d. OpenWeatherMap API

The Open Weather Map API is a weather API that allows users to access various weather data. In this program, this API is used to get the current weather in a specific city that is inputted by the user.

e. json.jar

This jar is used to handle the JSON file from the weather API call. When we call the OpenWeatherMap API through the weather button in the Edit Aranara menu, the API will return the weather result in the form of a JSON file. This jar contains methods to help us handle and get the information that we want from the returned JSON file.

8. Program Files and Folders

All documentations of this program are stored inside the 'document' folder, while all the libraries used are stored in the 'libs' folder. Codes and media (image and audio) are stored inside the 'src/App' folder where the image files are stored inside the 'img' folder and the audio files are stored inside the 'sound' folder. Below is the more detailed description for each file inside the 'src/App' folder (excluding the 'img' and 'sound' folder files).

a. AddNewTask.java, used for adding new tasks to the calendar.
b. AddWorkflowMenu.java, used for creating, editing, and deleting workflows.
c. AranaraChatMenu.java, contains buttons to interact with the Aranara.
d. AranaraDropShadowPanel.java, a custom panel class that is used as an individual panel for each Aranara in the Aranara menu, so that the user can choose which Aranara that they want to visit.
e. AranaraMenu.java, contains panels of different Aranaras.
f. ButtonCustom.java, a custom button class for buttons with round edges.
g. CalendarCell.java, a custom button class for the calendar cell for each date.
h. CalendarCustom.java, a custom panel class that contains the calendar panel, task panel, and several labels and buttons to create the whole calendar display.
i. CalendarPage.java, a menu that consists of the calendar and several features related to tasks.
j. CalendarPanel.java, a custom layered pane class that contains all the calendar cells that construct the calendar panel.
k. CalendarToday.java, a class that is used to store methods for handling today's date.
l. CloneablePanelContact.java, a custom panel class for displaying list of contacts.
m. CloneablePanelFlow.java, a custom panel class for displaying list of flows.

n. CloneablePanelMsg.java, a custom panel class for displaying list of message templates.
o. CloneablePanelTask.java, a custom panel class for displaying list of tasks.
p. CloneablePanelWorkflow.java, a custom panel class for displaying list of workflows.
q. Contact.java, to initialize and handle Contact objects.
r. ContactOthers.java, contains feature to contact other people through WhatsApp.
s. ContactsList.java, used for creating, updating, and deleting contacts.
t. CreateNewWorkflow.java, used for creating a new workflow.
u. EditAranara.java, used as the main page for interacting with an Aranara.
v. EditTask.java, used for updating information of a specific task.
w. EditWorkflow.java, used for editing flows inside a workflow.
x. Flow.java, used for handling Flow objects.
y. HomePage.java, initialize the HomePage frame and display information inside it.
z. InsertWorkflowTask.java, used to insert the workflow as tasks based on the inputted D-day.
aa. Login.java, initialize the login page to let the user login into their existing account.
bb. Message.java, used for handling Message objects.
cc. MsgTemplate.java, used for creating, updating, and deleting message templates.
dd. MusicPlayer.java, used to play the background music.
ee. PanelSlide.java, a custom panel class that allows the slide animation. This class mainly used when user switches month in the calendar.
ff. PlaceHolderJTextArea.java, a custom textarea class that contains a placeholder.
gg. PlaceHolderTextField.java, a custom textfield class that contains a placeholder.
hh. Quotes.txt, contains the random quotes that will be displayed in the homepage.
ii. RoundJTextField.java, a custom textfield class that is round edged and has no placeholder.
jj. SfxPlayer.java, used to play the sound effects in the Edit Aranara menu.
kk. Signup.java, initialize the sign up page to let the user create a new account.
ll. Task.java, for handling task objects.
mm. TextResult.java, used to generate the date in form of text based on the workflow and inputted D-day.

nn. WelcomePage.java, initialize the welcome page.

oo. Workflow.java, for handling workflow objects.

pp. WrappedLabel.java, a custom label class that allows the text to be wrapped or continue to the next line after the text has a certain width.

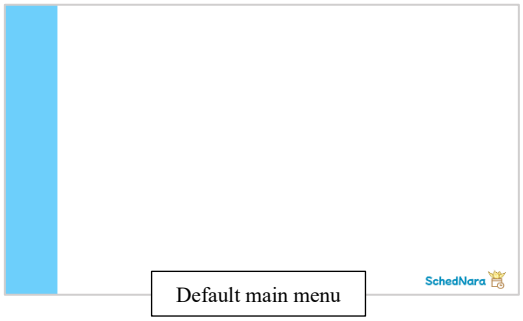
C. Solution Design

1. Design Choices

For this program, I did not want to have a rigid static design as a formal application although its main feature is closely related to productivity. Hence, I thought of using Genshin based cute character in the main design as Genshin Impact is currently my favorite game. I then thought of my favorite characters in the game which are Nahida and the Aranaras. At last, I chose the Aranaras as I thought they were easier to implement as part of the element in the app. Hence, the main design reference of this program is from HoYoVerse Genshin Impact or 原神 (https://genshin.hoyoverse.com/).

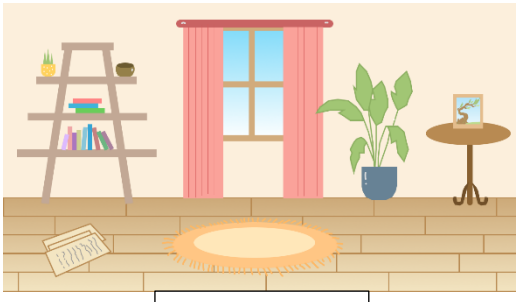
As stated before in part B2, all the background images inside the program are drawn by me using the tools on Figma. These are the four main backgrounds that are distinct from each other as I have explained before in B2.

Welcome Page



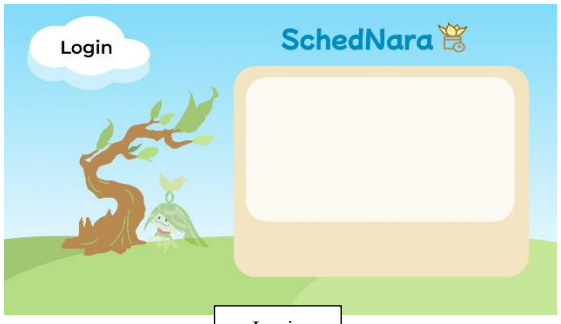
Default main menu

Edit Workflow

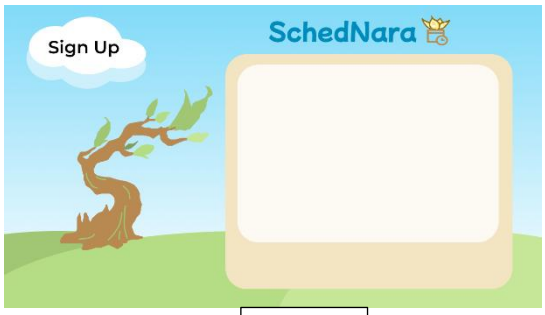


Edit Aranara

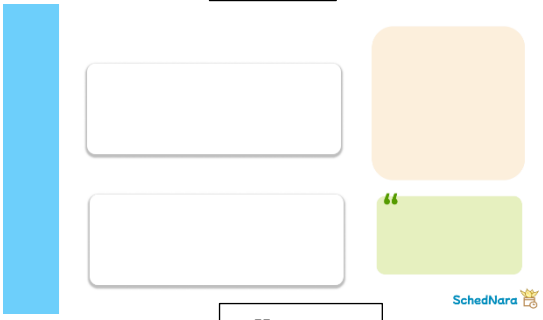
The login and the signup page is similar to the welcome page with the presence of Arama and the Ashvattha Tree as at the end of the Aranara quest, Arama becomes the seed of that specific tree. Another background page is the homepage background which is similar to the default main menu background with the only distinction being having several shadow panels.



Login



Sign up



Homepage

Meanwhile, the icons are imported from the Iconify plugins in Figma.

The only objects that I did not create or use Figma assets are the Aranara (Arama, Ararycan, and Arabalika). The images are shown below. The source of these three images is from Misaaa 哈啾 on Weibo (https://weibo.com/6671732232/M9lWyBnCp) where she/he allows personal use of these images.

To further enhance the liveliness of the Aranaras, I also animate the Aranara, so that they are floating and I export them into .gif files. Therefore, in the Edit Aranara menu, the Aranara is doing several minor movements (the .gif file), while on the homepage, they are still static (in the form of .png file).

2. Font and Audio

The font that I used in this application is mainly Montserrat (Regular, Medium, and SemiBold) because, after all, this program is used to help with user work, so an upright sans-serif font is more suitable. However, for the logo and some of the text in the Edit Aranara menu, I used the Mochiy Pop One font due to its cuteness. It is also in line with my desire as I did not intend SchedNara to be a fully rigid and formal schedule app.

- Source for Montserrat: https://fonts.google.com/specimen/Montserrat
- Source for Mochiy Pop One: https://fonts.google.com/specimen/Mochiy+Pop+One

Meanwhile, for the audio files, as this program design heavily references Genshin Impact, the background music is also mainly from HOYO-MIX. Below is the detailed source for each background music and sound effect.

a. Background Music

- Enchanting Bedtime Stories (by HOYO-MIX)
  https://youtu.be/5bp51REb1pw?si=Vd_7S71qH6NiLA18
- For Riddles, For Wonders (by HOYO-MIX)
  https://youtu.be/p2QSNcAJLTI?si=4J_JXqUUVM-gSTBr
- I've Never Forgotten (by ChillChill for 2023 原神新春会, piano by Frost Piano (프뮤))
  https://youtu.be/SnrDK5kbYZk?si=6SOpCb_zlqlwG_c7
- Melody of Hidden Seeds (by HOYO-MIX)
  https://youtu.be/w_14Ibukqo4?si=5KTH_a0j1DuWllqE

b. Sound Effects

- win.wav, by Pixabay from Pixabay (https://pixabay.com/sound-effects/success-fanfare-trumpets-6185/) .

- select.wav, by Otto VoiceBosch from Pixabay (https://pixabay.com/sound-effects/menu-select-button-182476/).
- lose.wav, by UNIVERSFIELD from Pixabay (https://pixabay.com/sound-effects/fail-144746/).
- draw.wav, by floraphonic from Pixabay (https://pixabay.com/sound-effects/90s-game-ui-6-185099/).
- chat.wav, by UNIVERSFIELD from Pixabay (https://pixabay.com/sound-effects/button-124476/).

3. Diagrams

All the diagrams (the use case diagram, the activity diagram, and the class diagram) are inside the 'document' folder (in the same folder as this report).

4. Data Structure Used

There are three data structures used in this program, which are the Linked List, HashMap, and Array. Linked List is mainly used for storing objects that are going to be created cloneable panels for it. For instance, contacts, workflows, flows, tasks, and so on. Linked List is great for storing them because their order matters and Linked List has methods to properly update objects in a particular index.

Meanwhile, there are two types of HashMap in this program which are the month Map and the color Map. The month Map stores a pair of String keys and String values with the key being the name of the month and the value being the number representation of the month (example: "January" : "01"). The color Map stores a pair of String key (color name) and Color object value. The month Map is used to get the representative number of a specific month based on its name, while the color Map is used to get the Color object based on the color name.

Lastly, Array is used to store the quotes read from the txt file and for storing Strings that will be set in the JComboBox. The array has a fixed size, so it is safer for these kinds of data so that they can not easily be changed.

D. Essential Algorithms

In this part, I will only show and explain some crucial code and algorithms for each class.

1. Classes in Database Connection

The classes in the Database Connection folder is used to get the connection to the SQL database and store the API key for OpenWeatherMap API.

a. ConnectionProvider

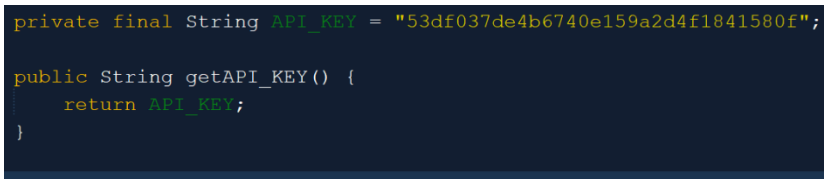
```
public static Connection getCon(){
    try{

        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/schednara", "root", "");
        return con;

    }catch(Exception e){
        System.out.print("error: " + e);
        return null;
    }
}
```

The getCon() method in the ConnectionProvider class is used to connect the program with the local SQL database. It will return the connection to the program if it is called.

b. CONFIG



```
private final String API_KEY = "53df037de4b6740e159a2d4f1841580f";

public String getAPI_KEY() {
    return API_KEY;
}
```

This class is hidden in the .gitignore and is only visible in my local device. I did not set the API key immediately in the AranaraChatMenu frame (where the API is called) because API key should be hidden, so that it would not be misused by other people in the GitHub.

2. Regular Classes

This program uses several regular classes to initialize objects as instances of them to ease the code implementation. For example, a flow has several attributes, so we will make a Flow class that contains all the attributes needed so that we do not need to initialize multiple variables and keep it on track every time. We only need to initialize the Flow object. All regular classes have getters and setters for all of the attributes that they have and a constructor.

a. Contact

```java
public class Contact {
    //attributes
    private String id;
    private String name;
    private String phone_number;

    //constructor
    public Contact(String id, String name, String phone_number){
        this.id = id;
        this.name = name;
        this.phone_number = phone_number;
    }
```

Contact instances are used to represent a person's contact. Contact objects are used in ContactsList and ContactOther when the user wants to contact others through WhatsApp. A Contact object has id, name, and phone number as its attributes.

b. Flow

```java
public class Flow {
    //attributes
    private String id;
    private String workflowID;
    private String nameInput;
    private String typeInput;
    private int dayFromInput;
    private int dayToInput;
    private String noteInput;
    private String colorInput;

    //constructor
    public Flow(String id, String workflowID, String nameInput, String typeInput, int dayFromInput, int dayToInput,
            String noteInput, String colorInput) {
        this.id = id;
        this.workflowID = workflowID;
        this.nameInput = nameInput;
        this.typeInput = typeInput;
        this.dayFromInput = dayFromInput;
        this.dayToInput = dayToInput;
        this.noteInput = noteInput;
        this.colorInput = colorInput;
    }
```

A Flow object is used to represent a flow or a checkpoint inside a workflow. The word flow is used to refer to the task, step, or process that has to be done in the workflow. For example, to make wine, we need to harvest the grape, clean and crush it, and after that ferment it. Each step is called flow in this context. Flow has 8 attributes, which are id, workflowID, name, type, dayFrom, dayTo, note, and color. Flow has two types, which are one-day event and multiple-day event. Meanwhile, dayFrom is the starting day and dayTo is the last day. dayTo is only observed if the Flow type is a multiple-day event.

c. Message

```
public class Message {
    //attributes
    private String id;
    private String msg;

    //constructor
    public Message (String id, String msg){
        this.id = id;
        this.msg = msg;
    }
```

The Message object is used to represent a message. The message object will mostly be used in the MsgTemplate and ContactOthers menu where the user can customize the message template for sending it to other people. A Message object only contains two attributes, which are the id and the msg (the message content).

d. Task

```
public class Task {
    //attributes
    private String id;
    private String userID;
    private String nameInput;
    private String typeInput;
    private String timeFromInput;
    private String timeToInput;
    private String noteInput;
    private String colorInput;
    private boolean completed = false;

    //constructor
    public Task(String id, String nameInput, String typeInput, String timeFromInput, String timeToInput,
            String noteInput, String colorInput, String userID, boolean comp) {
        this.id = id;
        this.userID = userID;
        this.nameInput = nameInput;
        this.typeInput = typeInput;
        this.timeFromInput = timeFromInput;
        this.timeToInput = timeToInput;
        this.noteInput = noteInput;
        this.colorInput = colorInput;
        this.completed = comp;
    }
```

As its name suggests, the Task object is used to depict a task in the calendar. The Task object is similar to the Flow object, but it has the completed attribute and the time attributes are in String. Flow object only represents a step for doing something, while Task object represents a real task that you have to accomplish (you are already in the process to do it). That is why the time attributes (timeFrom and timeTo) are in String because they represent the date and completed represents whether the task is completed or not. Task also uses userID rather than workflowID because it is not part of a workflow. Other than that, the other attributes (id, name, type, note, color) are the same.

e. Workflow

```java
public class Workflow {
    //attributes
    private String title;
    private int checkpoint;
    private String id;
    private String userID;

    //constructor
    public Workflow(String title, int checkpoint, String id, String userid) {
        this.title = title;
        this.checkpoint = checkpoint;
        this.id = id;
        this.userID = userid;
    }
```

The Workflow object is used to represent the workflows. It only has four attributes, which are title, checkpoint, id, and userID. The checkpoint here refers to the number of checkpoints or flows inside the workflow.

f. CalendarToday

```java
public class CalendarToday {
    //attributes
    private int day;
    private int month;
    private int year;

    //constructor
    public CalendarToday(int day, int month, int year) {
        this.day = day;
        this.month = month;
        this.year = year;
    }

    //see whether the day, month, and year is the same as the one represented in CalendarToday
    public boolean isToday(CalendarToday date){
        return day == date.getDay() && month == date.getMonth() && year == date.getYear();
    }
```

The CalendarToday object is used to represent today's date. It has three attributes, which are day, month, and year (all in integer). Besides getters and setters, it has another method, which is the isToday method. The isToday method will see whether the date from another object is the same or not as the one represented by CalendarToday (today's date).

3. Custom JComponent Classes

This program utilizes many custom classes that inherit JComponent because of the limitations that the default JComponent has. For example, the JButton in the default

javax.swing package did not have round edges. Therefore, to realize the appearance of these components, I have used several custom classes, which are specified below.

a. ButtonCustom

```
public ButtonCustom() {
    //  Init Color as all null
    setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
    setColor(null);
    colorOver = null;
    colorClick = null;
    borderColorNotOver = null;
    color2 = null;
    colorOver2 = null;
    colorClick2 = null;
    borderColorOver = null;
    setContentAreaFilled(false);


    //  Add event mouse and set the colors
    addMouseListener(new MouseAdapter() {
        @Override
        public void mouseEntered(MouseEvent me) { //mouse entered
            setBackground(colorOver);
            setForeground(colorOver2);
            setBorderColor(borderColorOver);
            over = true; //hover become true
        }

        @Override
        public void mouseExited(MouseEvent me) { //mouse exited
            setBackground(color);
            setForeground(color2);
            setBorderColor(borderColorNotOver);
            over = false; //hover become false
        }

        @Override
        public void mousePressed(MouseEvent me) { //mouse pressed
            setBackground(colorClick);
            setForeground(colorClick2);
            setBorderColor(borderColorOver);
        }

        @Override
        public void mouseReleased(MouseEvent me) { //mouse released
            if (over) { // if it is hovered
                setBackground(colorOver);
                setForeground(colorOver2);
                setBorderColor(borderColorOver);

            } else { //if it is not hovered
                setBackground(color);
                setForeground(color2);
                setBorderColor(borderColorNotOver);
            }
        }
    });
}
```

The ButtonCustom is a custom class inherited from the JButton class.

When first initializing ButtonCustom, all the colors are set to null. After that, for every mouse event, we change and condition the color. Then, we override the paintComponents method from JComponent, so that it will draw a round rect (so that we have round edges).

```
@Override
protected void paintComponent(Graphics grphcs) {
    Graphics2D g2 = (Graphics2D) grphcs;
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
            RenderingHints.VALUE_ANTIALIAS_ON);
    //  Paint Border
    g2.setColor(borderColor);
    g2.fillRoundRect(0, 0, getWidth(), getHeight(),
            radius, radius);
    g2.setColor(getBackground());
    //  Border set 2 Pixel
    g2.fillRoundRect(2, 2, getWidth() - 4, getHeight() - 4,
            radius, radius);
    super.paintComponent(grphcs);
}
```

b. WrappedLabel

The WrappedLabel is a custom class that inherits from the JLabel class, where it can wrap the text inside the label so that it will continue in a new line (not changing to ...).

```
@Override
protected void paintComponent(Graphics g) {
    String text = getText();
    FontMetrics fm = g.getFontMetrics();

    int lineHeight = fm.getHeight();
    int x = (getWidth() - getPreferredSize().width) / 2; //center the text horizontally
    int y = fm.getAscent();

    //for every line in the array
    for (String line : getWrappedLines(text, fm)) {
        g.drawString(line, x, y); //draw current line
        y += lineHeight; //go to next line
    }
}
```

It will first get the text and calculate the height of the text. Then, it will calculate the x starting position. After that, it will iterate over the line wrapped array and draw each line of text.

```
private String[] getWrappedLines(String text, FontMetrics fm) {
    StringBuilder currentLine = new StringBuilder();
    java.util.List<String> wrappedLines = new java.util.ArrayList<>();
    String[] words = text.split("\\s+");

    for (String word : words) {
        // Check if the word itself is longer than maxWidth
        if (fm.stringWidth(word) > maxWidth) {
            int start = 0;
            while (start < word.length()) {
                int end = start + 1;
                while (end <= word.length() && fm.stringWidth(word.substring(start, end)) <= maxWidth) {
                    end++;
                }
                String subword = word.substring(start, end - 1);
                // Check if adding the subword to the current line exceeds maxWidth
                if (currentLine.length() > 0 && fm.stringWidth(currentLine.toString() + " " + subword) > maxWidth) {
                    wrappedLines.add(currentLine.toString());
                    currentLine = new StringBuilder(subword); // Reset currentLine
                } else {
                    // Add the subword to the current line
                    if (currentLine.length() > 0 && currentLine.charAt(currentLine.length() - 1) != ' ') {
                        currentLine.append(" ");
                    }
                    currentLine.append(subword);
                }
                start = end - 1;
            }
        } else {
            // Check if adding the word to the current line exceeds maxWidth
            if (currentLine.length() > 0 && fm.stringWidth(currentLine.toString() + " " + word) > maxWidth) {
                wrappedLines.add(currentLine.toString());
                currentLine = new StringBuilder(word);
            } else {
                // Add the word to the current line
                if (currentLine.length() > 0 && currentLine.charAt(currentLine.length() - 1) != ' ') {
                    currentLine.append(" ");
                }
                currentLine.append(word);
            }
        }
    }
    // Add the last line if there's any remaining content
    if (currentLine.length() > 0) {
        wrappedLines.add(currentLine.toString());
    }

    return wrappedLines.toArray(new String[0]);
}
```

The getWrappedLines() method is used to return a String array of each line content or text. It will first check whether the text width is larger than the max width. If it is larger, then it will iterate to split the text into two lines. If the current line is longer than the max width, it will split again until the last line. It will then return the wrapped lines as an array of String.

```java
@Override
public Dimension getPreferredSize() {
    FontMetrics fm = getFontMetrics(getFont());
    String text = getText();
    String[] lines = getWrappedLines(text, fm);

    int maxWidth = 0;
    for (String line : lines) {
        //calculate the maximum width of the wrapped lines
        int lineWidth = fm.stringWidth(line);
        if (lineWidth > maxWidth) {
            maxWidth = lineWidth;
        }
    }
    //calculate the total height required
    int lineHeight = fm.getHeight();
    int totalHeight = lines.length * lineHeight;

    //return it in form of dimension
    return new Dimension(maxWidth, totalHeight);
}
```

The getPreferredSize is overridden from the JComponent method. Here, this method will get the max width and the total height (line height times the amount of lines) as the preferred size dimension.

c. CloneablePanelContact

In this program, there is a total of 5 different but similar cloneable panels that inherit from the JPanel class. The difference between each of them is only the content inside the panel which is often specified inside the constructor. Some have buttons and labels, some only have labels, and so on.

```java
//init the name label based on contact name
JLabel title = new JLabel();
title.setText(nameInput);
title.setForeground(Color.black);
title.setFont(new Font("Montserrat Semibold", 0, 14));
title.setBounds(10, 5, title.getPreferredSize().width, title.getPreferredSize().height);
title.setPreferredSize(new Dimension(300, title.getPreferredSize().height));
add(title);

//init the phone number label based on the phone number
JLabel phone = new JLabel();
phone.setText(phoneInput);
phone.setForeground(Color.black);
phone.setFont(new Font("Montserrat", 0, 14));
phone.setBounds(10, 30, title.getPreferredSize().width, title.getPreferredSize().height);
add(phone);
```

```
//init the delete button
JLabel deleteBtn = new JLabel();
deleteBtn.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/delete_contact.png")));
deleteBtn.setBounds(235, 10, deleteBtn.getPreferredSize().width, deleteBtn.getPreferredSize().height);
deleteBtn.addMouseListener(new MouseAdapter() {
    //delete btn behaviour
    @Override
    public void mouseClicked(MouseEvent e) { //when clicked
        deleteContact();
    }
    @Override
    public void mouseEntered(MouseEvent e) { //when hovered
        deleteBtn.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/delete_contact_hover.png")));
    }
    @Override
    public void mouseExited(MouseEvent e) { //when not hovered
        deleteBtn.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/delete_contact.png")));
    }
});
add(deleteBtn);
```

For contacts, the cloneable panel has the title and phone number label and a delete button. The labels and buttons have properties as specified above.

```
@Override
protected void paintBorder(Graphics g) {
    super.paintBorder(g);
    double strokeWidth;

    Graphics2D g2d = (Graphics2D) g.create();
    if (isClicked) {
        //set border as blue and thicker when the panel is clicked
        g2d.setColor(new Color(125,201,255));
        g2d.setStroke(new BasicStroke(4)); // Set border width
        strokeWidth = 4f;
    } else {
        //set the border as the default black
        g2d.setColor(Color.black);
        g2d.setStroke(new BasicStroke(borderWidth)); // Set border width
        strokeWidth = borderWidth;
    }

    //draw border
    int offset = (int) (strokeWidth / 2);
    g2d.drawRoundRect(offset, offset, getWidth() - 1 - offset * 2, getHeight() - 1 - offset * 2, borderRadius, borderRadius);

    g2d.dispose();
}
```

If the panel is clicked, then it will have a blue border color, while in default, the border color is supposed to be black.

```
private void deleteContact(){
    //confirm whether the user wants to delete or not
  String question = "Do you really want to delete " + nameInput + "?";
  int a = JOptionPane.showConfirmDialog(home.getContentPane(), question, "SELECT", JOptionPane.YES_OPTION);

    if (a == 0){
        //delete from database
        try{
            //run query and delete the contact based on its ID
            Connection conn = ConnectionProvider.getCon();
            String query = "DELETE FROM contact WHERE id = ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, id);

            ps.executeUpdate();

            //display success message
            String message = "Contact deleted successfully.";
            JOptionPane.showMessageDialog(home.getContentPane(), message);
            //reload home
            home.reloadSelf();

        }catch(SQLException se){
            //show error message
            JOptionPane.showMessageDialog(home.getContentPane(), se);
        }
    }
}
```

When the delete button is clicked, then it will first ask for confirmation from the user. If the user says yes, then it will delete the contact from the contact table in the MySQL database based on the contact ID. After that, it will display the success message and reload the frame where this cloneable panel belongs.

d. CloneablePanelMsg

The cloneable panel for messages is highly similar to the cloneable panel for contacts. The difference is only that this cloneable panel is made for messages, not contacts.

```
// init the name label
WrappedLabel msg = new WrappedLabel(250);
msg.setText(msgInput);
msg.setForeground(Color.black);
msg.setFont(new Font("Montserrat", 0, 14));
msg.setBounds(10, 5, msg.getPreferredSize().width, msg.getPreferredSize().height);
msg.setPreferredSize(new Dimension(300, msg.getPreferredSize().height));
add(msg);

//init the delete button
JLabel deleteBtn = new JLabel();
deleteBtn.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/delete_contact.png")));
deleteBtn.setBounds(270, 10, deleteBtn.getPreferredSize().width, deleteBtn.getPreferredSize().height);
//the delete button behaviour
deleteBtn.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) { //when clicked, perform delete message
        deleteMsg();
    }
    @Override
    public void mouseEntered(MouseEvent e) { //when hovered, change to lighter color
        deleteBtn.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/delete_contact_hover.png")));
    }
    @Override
    public void mouseExited(MouseEvent e) { //when not hovered, change back to default
        deleteBtn.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/delete_contact.png")));
    }
});
```

The cloneable panel for messages only has the label for the message and a delete button. If the delete button is clicked, then it will delete the selected message. The selected message panel (similar to the contact cloneable panel) also has a blue border color while the default border color is black. That is also why the paintBorder method for CloneablePanelMsg is the same as the one in CloneablePanelContact.

```
private void deleteMsg(){
    //ask for confirmation
  String question = "Do you really want to delete " + msgInput + "?";
  int a = JOptionPane.showConfirmDialog(home.getContentPane(), question, "SELECT", JOptionPane.YES_OPTION);

    if (a == 0){
        //delete from message table in database based on the ID
        try{
            Connection conn = ConnectionProvider.getCon();
            String query = "DELETE FROM message_template WHERE id = ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, id);

            ps.executeUpdate();
            //show success message
            String message = "Message deleted successfully.";
            JOptionPane.showMessageDialog(home.getContentPane(), message);
            //reload home
            home.reloadSelf();

        }catch(SQLException se){
            JOptionPane.showMessageDialog(home.getContentPane(), se);
        }
    }
}
```

The deleteMsg() function is also almost the same as the delete contact function. It will first ask the user for confirmation, then it will delete the message from the message_template in the database based on the message ID.

e. CloneablePanel Workflow

The cloneable panel for workflow is made for workflows.

```
// init title label
WrappedLabel title = new WrappedLabel(250);
title.setText(titleInput);
title.setFont(new Font("Montserrat SemiBold", 0, 36));
setComponentBounds(title, 25, 27, title.getPreferredSize().width, title.getPreferredSize().height);
add(title);

//init text field for changing name
JTextField titleField = new JTextField();
titleField.setVisible(false);
titleField.setFont(new Font("Montserrat SemiBold", 0, 36));
setComponentBounds(titleField, 25, 27, titleField.getPreferredSize().width, titleField.getPreferredSize().height);
add(titleField);
```

```
//add behaviour for title label
title.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) { //when clicked
        //switch to textfield, so that user can change the workflow title
        titleField.setText(title.getText());
        titleField.setVisible(true);
        title.setVisible(false);

        Dimension preferredSize = title.getPreferredSize();
        setComponentBounds(titleField, 25, 27, preferredSize.width + 10, preferredSize.height);  // Adding some extra width

        titleField.requestFocus();
        titleField.selectAll();
    }
});

//behaviour of the titlefield
titleField.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { //when user press Enter key
        //set the title label text the same as the title field text
        title.setText(titleField.getText());

        //update name in the database
        updateNameDatabase(title);

        //swutch to become label again
        title.setVisible(true);
        titleField.setVisible(false);
        setComponentBounds(title, 25, 27, title.getPreferredSize().width, title.getPreferredSize().height);
        //reload home to show change
        home.reload();
    }
});
```

Its title is displayed in a WrappedLabel, but when the user clicks that label, then it will switch to a text field. Users can change the workflow name in the text field and press Enter. After that, the frame will reload and the name in the frame and database will be changed.

```
private void updateNameDatabase(JLabel nameLabel){
    //update workflow title in the database based on its ID
    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);

        PreparedStatement ps = con.prepareStatement("UPDATE workflow SET title = ? WHERE workflowID = ?");
        ps.setString(1, nameLabel.getText());
        ps.setString(2, id);
        ps.executeUpdate();

    }catch(SQLException e){
        e.printStackTrace();
    }
}
```

The updateNameDatabase method will first get the new title from the label. Then, it will update the workflow table and set the title of the corresponding workflow (based on its ID) with the new name.

```
//init total checkpoint label
JLabel total_check = new JLabel();
total_check.setFont(new Font("Montserrat", 0, 24));
total_check.setText("Total: " + checkpointInput + " checkpoints");
setComponentBounds(total_check, 25, title.getY()+title.getHeight()+30, total_check.getPreferredSize().width+10, total_check.getPreferredSize().height);
add(total_check);

//init delete button
ButtonCustom deleteButton = new App.ButtonCustom();
deleteButton.setBorder(null);
deleteButton.setBorderColor(bgColor);
deleteButton.setBorderColorOver(bgColor);
deleteButton.setBorderColorNotOver(bgColor);
deleteButton.setText("-");
deleteButton.setColor2(new Color(31, 139, 217));
deleteButton.setForeground(new Color(31, 139, 217));
deleteButton.setColor(Color.white);
deleteButton.setColorClick2(new Color(125, 201, 255));
deleteButton.setColorClick(Color.white);
deleteButton.setColorOver(Color.white);
deleteButton.setColorOver2(new Color(125, 201, 255));
deleteButton.setFont(new java.awt.Font("Montserrat Black", 0, 36));
deleteButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        deleteButtonActionPerformed(evt); //when the delete button is clicked
    }
});
setComponentBounds(deleteButton, 255, 10, deleteButton.getPreferredSize().width, deleteButton.getPreferredSize().height);
add(deleteButton);
```

```
//init edit button
ButtonCustom editButton = new App.ButtonCustom();
editButton.setForeground(new java.awt.Color(255, 255, 255));
editButton.setText("Edit");
editButton.setBorderColorNotOver(new java.awt.Color(31, 139, 217));
editButton.setBorderColor(new java.awt.Color(31, 139, 217));
editButton.setBorderColorOver(new java.awt.Color(125, 201, 255));
editButton.setColor2(Color.white);
editButton.setColor(new Color(31, 139, 217));
editButton.setColorClick2(Color.white);
editButton.setColorClick(new Color(125, 201, 255));
editButton.setColorOver(new Color(125, 201, 255));
editButton.setColorOver2(Color.white);
editButton.setFont(new java.awt.Font("Montserrat SemiBold", 0, 24));
editButton.setRadius(20);
editButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        editButtonActionPerformed(evt); //when the edit button is clicked
    }
});
setComponentBounds(editButton, 95, 189, editButton.getPreferredSize().width+25, editButton.getPreferredSize().height+7);
add(editButton);
```

Besides the title label and text field, the cloneable panel also has a label that shows how many flows or checkpoints the workflow has, a delete button to delete the workflow, and an edit button to go to the edit workflow frame.

```
private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
    AddWorkflowMenu home = (AddWorkflowMenu) SwingUtilities.getWindowAncestor(this);
    if (home.open == 0){
        //ask user for confirmation
        String message = "Do you really want to delete " + titleInput + "?";
        int a = JOptionPane.showConfirmDialog(home.getContentPane(), message, "SELECT", JOptionPane.YES_OPTION);

        if(a==0){
            // if yes
            try{
                //delete the workflow from the workflow table in the database based on its ID
                Connection con = ConnectionProvider.getCon();
                PreparedStatement ps = con.prepareStatement("DELETE FROM workflow where workflowID = ?");
                ps.setString(1, id);
                ps.executeUpdate();

                //show success message
                JOptionPane.showMessageDialog(home.getContentPane(), "Successfully deleted");
                home.reload(); //reload home frame

            }catch(Exception e){
                JOptionPane.showMessageDialog(home.getContentPane(), e);
            }
        }
    }
    else{
        JOptionPane.showMessageDialog(home.getContentPane(), "One window is already open.");
    }
}

private void editButtonActionPerformed(java.awt.event.ActionEvent evt) {
    AddWorkflowMenu home = (AddWorkflowMenu) SwingUtilities.getWindowAncestor(this);
    home.goToEdit(id); //go to edit workflow frame
}
```

When the delete button is clicked, it will first ask the user for confirmation. If the user says yes, then it will delete the workflow with the corresponding ID from the workflow table in the database. After that, it will display a success message and reload the home frame, so that the change will be shown. Meanwhile, if the edit button is clicked, then the user will be redirected to a new frame where they can edit the corresponding workflow flows.

f. CloneablePanelFlow

The cloneable panel for the Flow object exists in the Edit Workflow frame which is the frame that can be accessed when clicking the edit button in the previous CloneablePanelWorkflow.

```
// init the title label
WrappedLabel title = new WrappedLabel(270);
title.setText(nameInput);
title.setFont(new Font("Montserrat SemiBold", 0, 24));
title.setHorizontalAlignment(SwingConstants.CENTER);
title.setBounds(18, 15, title.getPreferredSize().width, title.getPreferredSize().height);
add(title);
```

```
//init the day label
JLabel total_day = new JLabel();
total_day.setFont(new Font("Montserrat", 0, 18));

//set up the text for the day label
String dayFrom = appendPlusToDay(dayFromInput);
String dayTo = appendPlusToDay(dayToInput);

//if it is one day event
if (typeInput.equals("One-day event")){
    //then only note the dayFrom because dayTo value is not important
    total_day.setText("(D " + dayFrom + ")");
}
//if tt is multiple day event
else if (typeInput.equals("Multiple-day event")){
    //set the dayfrom and dayto
    total_day.setText("(D " + dayFrom + ") to (D " + dayTo + ")");
}
setComponentBounds(total_day, 15, title.getY()+title.getHeight()+30,
        total_day.getPreferredSize().width+10, total_day.getPreferredSize().height);
add(total_day);

//the color label
JLabel color_label = new JLabel();
//set the color based on the value of the color name in the color map
color_label.setBackground(color_map.get(colorInput));
color_label.setOpaque(true);
setComponentBounds(color_label, 230, title.getY()+title.getHeight()+30, 33, 23);
add(color_label);
```

The CloneablePanelFlow is used to display information about a flow. It has a title label to display its title, a total_day label to display its duration, and a color label to display its color. The color label will only have color and no text. The custom appendPlusToDay method here is to condition the String so that when the dayFrom equals 1, the final result String will be D + 1.

The CloneablePanelFlow also has a different border color when it is clicked. The method is the same as the one in the previous CloneablePanelMsg and CloneablePanelContact where we override the paintBorder method. It will have a blue border color when it is clicked, but by default, the border color is black.

g. CloneablePanelTask

CloneablePanelTask is a cloneable panel for the Task object. Same as CloneablePanelWorkflow, it can not be selected.

```
// init the title label
WrappedLabel title = new WrappedLabel(220);
title.setText(nameInput);
title.setForeground(Color.white);
title.setFont(new Font("Montserrat", 0, 20));
title.setHorizontalAlignment(SwingConstants.CENTER);
title.setBounds(18, 10, title.getPreferredSize().width, title.getPreferredSize().height);
add(title);

//the view more or edit label in form of ...
JLabel more_label = new JLabel();
more_label.setForeground(Color.white);
more_label.setFont(new Font("Montserrat", 0, 36));
more_label.setText("...");
setComponentBounds(more_label, 226, -20, 23, 54);
more_label.addMouseListener(new MouseAdapter() {
    //the behaviour of this label
    @Override
    public void mouseClicked(MouseEvent e) {
        if (more_label.isEnabled()){ //if it is not disabled
          //then when we click it, we can edit the corresponding task
          home.goToEditTask(id);
        } else{
          // if disabled, then show the error messagee
          JOptionPane.showMessageDialog(home.getContentPane(), "The task has already been completed.");
        }

    }
    @Override
    public void mouseEntered(MouseEvent e) { //when mouse hover
        more_label.setForeground(new Color(125, 201, 255));
    }
    @Override
    public void mouseExited(MouseEvent e) { //when mouse not hover
        more_label.setForeground(Color.white);
    }
});
add(more_label);
```

```
//init the color label
JLabel color_label = new JLabel();
color_label.setBackground(color_map.get(colorInput));
color_label.setOpaque(true);

//if the task is completed, then set the color label to be ticked
if (completedInput == true){
    color_label.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/checkmark.png")));
    title.setFont(getStrikethrough(new Font("Montserrat", 0, 20))); //set strikethrough to task title
    more_label.setEnabled(false);//disable the view more label
}
setComponentBounds(color_label, 231, 55, 14, 14);
color_label.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) { //ithe color label is clicked
        //set ticked or not
        handleCheckmarks(color_label, title, more_label);
        //update the database based on this data
        updateCompletedDatabase();
    }
});
add(color_label);
```

The CloneablePanelTask has a title label to display its title, the more label to redirect the user to edit the corresponding task, and a color label that can be ticked (ticked color label means that the task is already completed). When the task is

completed, the color label will be ticked, the title text will be strikethrough, and the more label will be disabled. If we untick the color label, all of the components displayed will be back to the default one.

```
private void handleCheckmarks(JLabel color_label, JLabel title, JLabel more){
    //if the color label is ticked (the task is completed) and it is clicked again
    if (completedInput){
        //set the icon become null
        color_label.setIcon(null);
        title.setFont(new Font("Montserrat", 0, 20));
        more.setEnabled(true); //enable the view more label again
        this.completedInput = false; //set the task to be not completed
        updateTaskCompletionDatabase(-1); //update task completion table database to be minus by 1 for today
    }else{
        //if the task is not completed and it is clicked
        //set checkmark
        color_label.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/checkmark.png")));
        title.setFont(getStrikethrough(new Font("Montserrat", 0, 20))); //set the title to be strikethrough
        more.setEnabled(false); //disable the view more label
        this.completedInput = true;      //set compeleted to be true
        updateTaskCompletionDatabase(1); //update task completion table database to be added by 1 for today
    }
    //revalidate and repaint the color label
    revalidate();
    repaint();
}
```

The handleCheckmarks() method is used to handle the display of the components when the color label is clicked. If the task is already completed and clicked, then untick the color label, set the title to default, enable the more label, and update the task amount in the task_completion table minus by 1. If the task is clicked, but not completed, then tick the color label, set the title to strikethrough, disable the more label, and update the task amount in the task_completion table plus by 1.

```
private void updateCompletedDatabase(){
    //update the tasks database based on taskID to set whether the task is completed or not
    try{
        Connection con = ConnectionProvider.getCon();

        String query = "UPDATE tasks SET completed = ? WHERE taskID = ?";
        PreparedStatement ps = con.prepareStatement(query);
        ps.setBoolean(1, this.completedInput);
        ps.setString(2, this.id);

        ps.executeUpdate();

    }catch(Exception e){
        e.printStackTrace();
    }
}
```

This method will update the tasks table, where it will set the completed to be true or false based on the color label (whether it is ticked or not).

```
private void updateTaskCompletionDatabase(int a){
    //get today day name
    String day = LocalDate.now().getDayOfWeek().name();
    //get column name based on today day name
    String col_name = day.charAt(0) + day.substring(1,3).toLowerCase();
    //get the current completion rate and add it with a
    //a = 1 if user complete a new task and conversely a = -1
    int curr_completion = getTasksCompletion() + a;

    try{
        //update the task_completion database based on the column name and user ID
        Connection con = ConnectionProvider.getCon();

        String query = "UPDATE task_completion SET " + col_name + " = ? WHERE userID = ?";
        PreparedStatement ps = con.prepareStatement(query);
        ps.setInt(1, curr_completion);
        ps.setString(2, home.userID);

        ps.executeUpdate();

    }catch(Exception e){
        e.printStackTrace();
    }
}
```

Besides the tasks table, we also need to update task_completion table. The task_completion table stores the number of tasks that the user completes each day so that it can generate a chart on the homepage. So, first, it will get the name of today and after that update the number of tasks that the user completed today. Note that this table will be reset after each week (it will be discussed more in the homepage frame).

h. PlaceHolderJTextArea

```
public PlaceHolderJTextArea(String placeholder) {
    this.placeholder = placeholder;

    // Add focus listener to repaint the field on focus gain/loss
    setBorder(null);
    addFocusListener(new FocusAdapter() {
        @Override
        public void focusGained(FocusEvent e) {
            //if focus gained and the intial field is empty (only placeholder),
            //then delete all placeholder text
            if (getText().equals(placeholder) || getText().equals("")){
              setText("");
            }else{
              //if in ex: editTask, the initial field has text that is not placeholder
              //then dont delete it
              setText(getText());
            }
            repaint();
        }
```

```
        @Override
        public void focusLost(FocusEvent e) {
            repaint();
        }
    });
}

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);

    // Draw the placeholder text if needed
    if (getText().isEmpty() && !isFocusOwner()) {
        Graphics2D g2d = (Graphics2D) g.create();
        g2d.setColor(Color.GRAY); // Set the color for the placeholder text
        int padding = (getHeight() - getFont().getSize()) / 2;

        g2d.drawString(placeholder, getInsets().left, padding);
        g2d.dispose();
    }
```

The PlaceHolderJTextArea is a custom text area class that inherits from JTextArea. The difference is that it has placeholder text, while the regular JTextArea does not. Based on the above code, the placeholder is configured when focus is gained and focus is lost. When the focus is gained and the text area only contains the placeholder (that means that it is still empty), then the text area text will be empty. If the text area is not empty, then when the focus is gained, the content will be the same. The overridden paintComponent method also provides a way to only paint the placeholder text when the text area becomes empty.

i. PlaceHolderTextField

The PlaceHolderTextField is a custom text field class that inherits from JTextField. It also has placeholder text, while the regular JTextField does not. PlaceHolderTextField code is highly similar to PlaceHolderJTextArea's. The difference between them is that PlaceHolderTextField has another attribute, which is the adder, hence making it more customizable than PlaceHolderTextArea. The adder is used to set the x position of the text inside the text field.

```
    // Add focus listener to repaint the field on focus gain/loss
    setBorder(null);
    addFocusListener(new FocusAdapter() {
        @Override
        public void focusGained(FocusEvent e) {
            if (getText().equals(placeholder) || getText().equals("")){
                //if focus gained and the intial field is empty (only placeholder),
                //then delete all placeholder text
              setText("");
            }else{
                //if in ex: editTask, the initial field has text that is not placeholder
               //then dont delete it
              setText(getText());
            }
            repaint();
        }

        @Override
        public void focusLost(FocusEvent e) {
            repaint();
        }
    });
}

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);

    // Draw the placeholder text if needed
    if (getText().isEmpty() && !isFocusOwner()) {
        Graphics2D g2d = (Graphics2D) g.create();
        g2d.setColor(Color.GRAY); // Set the color for the placeholder text
        int padding = (getHeight() - getFont().getSize()) / 2;

        g2d.drawString(placeholder, getInsets().left+this.adder, getHeight() - padding - 1);
        g2d.dispose();
    }
}
```

When the focus is gained and the text field only contains the placeholder (the text field is still empty), then the text field text will be empty when we start typing on it. If the text field is not empty (there is content inside the text field), then when the focus is gained, the content will be the same. When the focus is lost, it will check whether the field is empty or not. If empty, then set the placeholder again by calling the repaint to trigger paintComponent. The overridden paintComponent method provides a way to only paint the placeholder text when the text field becomes empty.

j. RoundJTextField

RoundJTextField also inherits from the JTextField class. Unlike PlaceHolderTextField, it has round edges. It also has a placeholder similar to PlaceHolderTextField, but its placeholder is more rigid.

```
    // Remove the default border
    setBorder(null);

    addFocusListener(new FocusAdapter() {
        @Override
        public void focusGained(FocusEvent e) {
            //when focus gained, delete all text inside the text field
            setText("");
            repaint();
        }
        @Override
        public void focusLost(FocusEvent e) {
            repaint();
        }
    });
}
```

As shown above, the RoundJTextField will not check whether there is non-placeholder content or not inside the text field and automatically remove all text inside it. The paintComponent method is similar to the one in PlaceHolderJTextArea due to not having an adder.

```
@Override
protected void paintBorder(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;
    Stroke oldStroke = g2.getStroke(); //get the original stroke

    g2.setStroke(new BasicStroke(1)); //the thickness of the border
    g.setColor(getForeground());
    g.drawRoundRect(1, 1, getWidth() - 3, getHeight() - 3, 30, 30); //draw the round rect for the border

    g2.setStroke(oldStroke); // Restore the original stroke
}

@Override
public boolean contains(int x, int y) {
    if (shape == null || !shape.getBounds().equals(getBounds())) {
        shape = new RoundRectangle2D.Float(0, 0, getWidth()-1, getHeight()-1, 30, 30);
    }
    return shape.contains(x, y);
}
```

Meanwhile, the more essential thing in this class is the overridden paintBorder method, which will draw a round rect (border with round edges for the text field). The contains method is used to determine if a point (x, y) lies within a custom shape, specifically a RoundRectangle2D for the text field.

k. AranaraDropShadowPanel

AranaraDropShadowPanel is used inside the Aranara menu to represent each Aranara that the user has unlocked or not. It inherits from the JPanel class and is similar to CloneablePanel. The difference is that it has a drop shadow behind it and has two states, which are locked and unlocked.

```
//tArama is automatically unlocked for all users
//if it is other aranaras, the affection has to be greater than 0 to represent that it is unlocked
if (name.equals("Arama") || affection > 0){
    setOpaque(false);
    setLayout(null);
    setBackground(Color.white);

    //add the name of the aranara
    JLabel name_label = new JLabel(aranaraName);
    name_label.setBounds(35, 33, 200, 48); // Set the position and size of the label
    name_label.setFont(new java.awt.Font("Montserrat SemiBold", 0, 32));
    name_label.setForeground(new java.awt.Color(0,0,0));
    add(name_label);

    //add affection text
    JLabel affection_label = new JLabel ("Affection: " + aranaraAffection + "/100");
    affection_label.setBounds(35, 100, 220, 36); // Set the position and size of the label
    affection_label.setFont(new java.awt.Font("Montserrat", 0, 24));
    affection_label.setForeground(new java.awt.Color(0,0,0));
    add(affection_label);

    //add visit button
    App.ButtonCustom visit_btn = new App.ButtonCustom();
    visit_btn.setForeground(new java.awt.Color(255, 255, 255));
    visit_btn.setText("Visit");
    visit_btn.setBorderColor(new java.awt.Color(31, 139, 217));
    visit_btn.setBorderColorNotOver(new java.awt.Color(31, 139, 217));
    visit_btn.setBorderColorOver(new java.awt.Color(109, 207, 251));
    visit_btn.setColor(new java.awt.Color(31, 139, 217));
    visit_btn.setColor2(java.awt.Color.white);
    visit_btn.setColorClick(new java.awt.Color(109, 207, 251));
    visit_btn.setColorClick2(java.awt.Color.white);
    visit_btn.setColorOver(new java.awt.Color(109, 207, 251));
    visit_btn.setColorOver2(java.awt.Color.white);
    visit_btn.setFont(new java.awt.Font("Montserrat SemiBold", 0, 24)); // NOI18N
    visit_btn.setRadius(50);
    visit_btn.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            //when visit button is clicked
            visitBtnActionPerformed(userID, home, player);
        }
    });
    visit_btn.setBounds(68, 189, 135, 53);
    add(visit_btn);
}
```

Arama is the default Aranara for every user, so it will be automatically unlocked. The other two Aranaras (Ararycan and Arabalika) are still locked and will be unlocked when the affection reaches a certain number. When it is unlocked, by default, the unlocked Aranara will have 1 affection. So, if the Aranara is Arama or the affection is

greater than 0, then set the name label, affection label (to show how much the affection of the corresponding Aranara), and a visit button.

```
    else{
        //if the aranara is still locked
        setOpaque(false);
        setLayout(null);
        setBackground(new Color(234,234,234));

        //the requirements for each aranara other than arama
        int requirement =0;
        if (name.equals("Ararycan")){
            requirement = 60;
        }else if (name.equals("Arabalika")){
            requirement = 120;
        }
        //add the lock icon to the panel
        JLabel lock_label = new JLabel();
        lock_label.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/locked.png")));
        lock_label.setBounds(105, 68, 66, 88); // Set the position and size of the label\
        add(lock_label);

        //add the explanation text about how many affections needed to unlock
        WrappedLabel explanation = new WrappedLabel(250);
        explanation.setText("Reach a total affection of "+ requirement + " to unlock");
        explanation.setBounds(13, 180, 257, 60); // Set the position and size of the label
        explanation.setFont(new java.awt.Font("Montserrat Medium", 0, 20));
        explanation.setForeground(new java.awt.Color(155, 154, 154));
        add(explanation);
    }
}

private void visitBtnActionPerformed(String uid, AranaraMenu home, MusicPlayer player){
    //go to edit aranara menu which is the aranara activity page
    home.setVisible(false);
    new EditAranara(aranaraName, uid, player).setVisible(true);
}
```

If the Aranara is still locked, then display the lock icon and the explanation about how much affection requirement the user needs to unlock the Aranara. Meanwhile, when the previous visit button is clicked, then the user will be redirected to the Edit Aranara page to interact with the Aranara.

```
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g.create();
    //initialize the shadow size
    int x = shadowSize;
    int y = shadowSize;
    int shadow_width = getWidth() - shadowSize * 2;
    int shadow_height = getHeight() - shadowSize * 2;

    // Enable anti-aliasing for better quality
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);

    // Draw shadow
    g2.setColor(shadowColor);
    g2.fillRoundRect(x, y, shadow_width, shadow_height, arcWidth, arcHeight);
```

```
    // Draw the panel itself
    g2.setColor(getBackground());
    g2.fillRoundRect(0, 0, shadow_width, shadow_height, arcWidth, arcHeight);

    //draw the border
    g2.setColor(Color.BLACK); // You can change the border color as needed
    Graphics2D border = (Graphics2D) g;
    Stroke oldStroke = border.getStroke(); //get the original stroke

    border.setStroke(new BasicStroke(1)); //the thickness of the border
    g.setColor(getForeground());
    g.drawRoundRect(1, 1, shadow_width, shadow_height, arcWidth, arcHeight); //draw the round rect for the border

    g2.setStroke(oldStroke); // Restore the original stroke
    g2.dispose();
}
```

The overridden paintComponent plays a great part in drawing the shadow for the panel. It will first get the size and properties of the shadow. Then, it will draw the shadow based on the size and color. After that, it will draw the panel and the border for the panel. The panel has round edges, so the border will be round rect.

l. CalendarCell

The CalendarCell class represents the cell inside the calendar panel on the CalendarPage. The CalendarCell inherits from the JButton class and has several additional attributes, which are date, title (or isTitle), isToday, isSelected, color_map, hasTasks, colorStr, and taskAmount. CalendarCell has getters and setters for title, date, isToday, isSelected, hasTasks, and taskAmount. Additionally, the setter for hasTasks also sets the colorStr. The setter for isToday also sets the text color of the cell that has today's date as blue.

```
//color the cell content based on its attributes
public void currentMonth(boolean act, boolean isSunday){
    //if it is in current month, but not Sunday
    if(act && !isSunday){
        setForeground(new Color(58,58,58)); //black
        setFont(new java.awt.Font("Montserrat Medium", 0, 22));
    }
    //if it is in current month and Sunday
    else if (act && isSunday){
        setForeground(new Color(234, 111, 111)); //red
        setFont(new java.awt.Font("Montserrat Medium", 0, 22));
    }
    //if it is not in current month, but it is Sunday
    else if (!act && isSunday){
        setForeground(new Color(255, 167, 167)); //light red
    }
    //if it is neither in current month nor sunday
    else {
        setForeground(new Color(200,200,200)); //gray
    }
}
```

The above method is used to color the text inside the cell based on its attributes. For Sundays, it is red (light red if it's not in the current month) and for the rest, it is black if it is in the current month and grey if it is not.

```
@Override
protected void paintComponent(Graphics g){
    if(title){
        //if it is title, blue text and there is a bottom border
        g.setColor(new Color(0,141,189));
        g.drawLine(20, getHeight()-1, getWidth()-20, getHeight()-1);
    }
    if (isToday){
        //if it is today, light blue circle background
        Graphics2D g2 = (Graphics2D) g;
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
        g2.setColor(new Color(222, 247, 255));
        g2.fillRoundRect(30, 10, 48, 48, 100, 100);
    }
    super.paintComponent(g);
    if (hasTask && !title ){
        if (!colorStr.trim().isEmpty()) {
            //if the cell has task, draw the dot based on the color
            g.setColor(color_map.get(colorStr));
            int dotSize = 10; // Size of the dot
            int x = getWidth() / 2 - dotSize / 2;
            int y = getHeight() - dotSize - 5;
            g.fillOval(x, y, dotSize, dotSize);
        }
        else{
            System.out.print(colorStr);
        }
    }
}

@Override
protected void paintBorder(Graphics g) {
    //if it is selected, paint a gray border
   if (isSelected && !title) {
    int borderWidth = 2;
    Graphics2D g2d = (Graphics2D) g.create();
    super.paintBorder(g); // Ensure any superclass painting is done

    // Set border color and width
    g2d.setColor(new Color(214, 214, 214)); //gray
    g2d.setStroke(new BasicStroke(borderWidth)); // Set border width

    // Draw the rounded rectangle border
    g2d.drawRoundRect(2, 2, getWidth() - 4, getHeight() - 4, 10, 10); // Adjusted position and size

    g2d.dispose();
   }
}
```

The overridden paintComponent is used to draw the title (day name) to have a blue color and bottom line border, a light blue circle background behind today's date text, and draw the dot on the cell based on the color. Meanwhile, the paintBorder method is used to draw a grey border for the cell if the cell is selected and is not the title (day name).

m. CalendarPanel

CalendarPanel is a custom class that inherits from the JLayeredPane class. It has the 7 x 7 GridLayout so that it can contain 49 CalendarCells.

```
private void setDate(){
    //set all date in the cells in the panel
    //get the calendar and set the date, year, and month to the first day
    Calendar calendar = Calendar.getInstance();
    calendar.set(Calendar.YEAR, year);
    calendar.set(Calendar.MONTH, month-1); //month is zero based on Calendar
    calendar.set(Calendar.DATE, 1);

    //calculate the starting day of the first day of the week
    int start_day = calendar.get(Calendar.DAY_OF_WEEK) -1;
    calendar.add(Calendar.DATE, -start_day); //move calendar back to the start of the week
    CalendarToday today = getToday(); //get today date

    for (Component comp : getComponents()){
        CalendarCell cell = (CalendarCell) comp;
        if (!cell.isTitle()){ //skip cells that are title
            //add the cell to LinkedList
            cells.add(cell);
            //set the text of the cell to the date
            cell.setText(calendar.get(Calendar.DATE) + "");
            cell.setDate(calendar.getTime());
            //color and format the text color
            cell.currentMonth(calendar.get(Calendar.MONTH) == month -1,
                    calendar.get(Calendar.DAY_OF_WEEK) == Calendar.SUNDAY);

            //if the cell is clicked
            cell.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    //if it is not the current cell
                    if (current_cell != cell) {
                        if (current_cell != null) {
                            //if the current cell is not null, set the selected cell to be false
                            current_cell.setAsSelected(false);
                            home.refresh();
                        }
                        current_cell = cell; //set cell as current cell
                        home.refresh();
                    }
                }
```

```
            //if the date is today's date
            if (today.isToday(new CalendarToday(calendar.get(Calendar.DATE),
                    calendar.get(Calendar.MONTH), calendar.get(Calendar.YEAR)))){
                //set as today and default selected to be true
                cell.setAsToday();
                cell.setAsSelected(true);
                cell.repaint();
                current_cell = cell;
            }
            //move to the next date
            calendar.add(Calendar.DATE, 1);
        }
    }
```

The above method is the essential algorithm to set the date for the CalendarPanel. It will first get the calendar instance and set the date, year, and month for the first day. After that, it will calculate the first day of the week of that first day and move the calendar back to the start of the week. It will now color and format the date based on

their month (whether they are in the current month or they are Sundays). Then, when the cell is selected, the selected cell will become the current cell. After that, this method will set the date which is the same as today's date to be selected by default. In other words, the default current cell will be the cell with today's date.

```
//set the first cell to be selected when navigating to other months
public void setCell1Selected(){
    current_cell = cell1;
    current_cell.setAsSelected(true);
    current_cell.repaint();
}
```

When navigating to another month, this method will automatically set the first cell in the calendar to be the current cell (or is selected).

n. CalendarCustom

The CalendarCustom is a custom class that inherits from the JPanel. It consists of two major panels, which are the previous CalendarPanel and the taskPanel. There are also two arrows to navigate back and forth from one month to another month.

```
prevBtn.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        //if it is clicked, then navigate to previous month
        if (month == 1){
            //if the month is January, go back to December and year deducted by 1
            month = 12;
            year--;
        }else{
            //else, go back 1 month
            month--;
        }
        updateCalendarPanel(PanelSlide.AnimateType.TO_RIGHT); //animate to right
    }
```

```
//behaviour for the next arrow
nextBtn.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) { //when clicked
        //if month is December, then go to January and year added by 1
        if (month == 12){
            month = 1;
            year++;
        }else{
            //else, add 1 month from current
            month++;
        }
        updateCalendarPanel(PanelSlide.AnimateType.TO_LEFT); //animate to left
    }
```

The prevBtn is used to move to the month before the current month, while the nextBtn is to move to the month after the current month. When the prevBtn is clicked, the month will be reduced by 1, except if the current month is January. The month will instead become 12, but the year will be reduced by 1. Meanwhile, when the nextBtn is clicked, the month will be added by 1, except if the current month is December. The month will instead become 1 and the year will be added by 1. When moving backward, the animation is to the right, while when moving forward, the animation is to the left.

```
private void updateCalendarPanel(PanelSlide.AnimateType type) {
    //create new current panel based on the month and year
    currentPanel = new CalendarPanel(month, year, home);
    //animate the panel movement based on the direction
    slide.show(currentPanel, type);
    //set the first date cell in the panel to be selected
    currentPanel.setCell1Selected();
    showMonthYear();
    refreshTaskDots();
}

private void thisMonth(){
    //set the default month and year to this month
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(new Date());
    month = calendar.get(Calendar.MONTH) +1;
    year = calendar.get(Calendar.YEAR);
}

private void showMonthYear(){
    //show the month and year in the month_year JLabel based on
    //the month and year value
    Calendar calendar = Calendar.getInstance();
    calendar.set(Calendar.MONTH, month-1); //deducted by 1 because month starts from 0 here
    calendar.set(Calendar.YEAR, year);
    calendar.set(Calendar.DATE, 1);

    //the display format in MMMM yyyy
    SimpleDateFormat date_format = new SimpleDateFormat("MMMM yyyy");
    month_year.setText(date_format.format(calendar.getTime()));
}
```

The updateCalendarPanel method will update the panel based on the month and year and also animate the panel movement. The first date cell will also be set to be selected by default. The thisMonth method is called when we first initialize this CalendarCustom. It will set the default month and year as our current month and year. The showMonthYear method is used to show the month and year in the JLabel on top of the CalendarPanel in the format of month year.

```
public void refreshTaskDots(){
    //get all cells in current panel
    LinkedList<CalendarCell> panelCells = currentPanel.getCells();

    //check for every task in the task list
    for (int i = 0; i < home.taskList.size(); i++){
        LocalDate date_from = home.convertStrDate(home.taskList.get(i).getTimeFromInput()); //get the date
        String color_str = home.taskList.get(i).getColorInput(); //get the color

        //one day event
        if (home.taskList.get(i).getTypeInput().equals("One-day event")){
            for (CalendarCell cell1 : panelCells){
                //check whether the date in the cell is the same with one in the task or not
               LocalDate date_cell = cell1.getDate().toInstant().atZone(ZoneId.systemDefault()).toLocalDate();
               if (date_from.equals(date_cell)){
                   if (cell1.hasATask()){
                       cell1.setTaskAmount(cell1.getTaskAmount() +1); //set task amount
                   }
                   cell1.setHasTasks(true, color_str); //set the dot color
                }
            }
        //multiple day event
        }else{
            //get the last date
            LocalDate date_to = home.convertStrDate(home.taskList.get(i).getTimeToInput());
            for (CalendarCell cell1 : panelCells){
                //get cell date
                LocalDate date_cell = cell1.getDate().toInstant().atZone(ZoneId.systemDefault()).toLocalDate();

                //see whether the cell date is in the range of the multiple day ebvent
               if (date_cell.isAfter(date_from.minusDays(1)) && date_cell.isBefore(date_to.plusDays(1))){
                    if (cell1.hasATask()){
                       cell1.setTaskAmount(cell1.getTaskAmount() +1); //set task amount
                    }
                    cell1.setHasTasks(true, color_str); //set dot color
                }
            }
        }
    }
}
```

The refreshTaskDots method is used to refresh the task dots inside the CalendarCell. It will first iterate for every task in the LinkedList and every cell in the panel to check whether the date is the same (if it is one-day event) or the date is in the range of the task date (if it is multiple-day event). If yes, then it will set the colored dot. The task amount of the corresponding cell will be incremented too.

```
public void addTaskBtnActionPerformed(){
    //if there is no window open
    if (CalendarPage.open == 0){
        //open the add new task window
       CalendarPage.open = 1;
       new AddNewTask(home.userID, home).setVisible(true);
    }else{
        //show message
       JOptionPane.showMessageDialog(home.getContentPane(), "One window is already open.");
    }
}
```

The addTask button inside the taskPanel will open a new AddNewTask window when it is clicked if there are no other windows that are open in the CalendarPage.

o. PanelSlide

PanelSlide is a custom class that inherits from the JPanel. It has a special property which is it can be animated. Below is the essential algorithm for animating the panel.

```
//show component with sliding animation
public void show(Component com, AnimateType animateType) {
    // Check if the timer is not already running
    if (!timer.isRunning()) {
        this.animateType = animateType;
        this.comShow = com;
        com.setSize(getSize()); // Set the size of the new component

        // If there are no components currently in the panel
        if (getComponentCount() == 0) {
            add(com); // Add the new component to the panel
            comExit = com; // Set comExit to the new component
            repaint();
            revalidate();
        } else {
            // Set the initial location of the new component for the animation
            if (animateType == AnimateType.TO_RIGHT) {
                comShow.setLocation(-comShow.getWidth(), 0); // Start from the left
            } else {
                comShow.setLocation(getWidth(), 0); // Start from the right
            }

            add(com); // Add the new component to the panel
            repaint();
            revalidate();
            timer.start(); // Start the animation timer
        }
    }
}
```

The show method is used to show the old and new component movement or animation. The PanelSlide uses a timer to track the movement time. If the timer is not running, then we will set the size of the new component. If there are no components currently in the panel, we add the new component to the panel and set the existing component to be the new component. If there is still an existing component, then set the location of the new component for the animation.

```
private void animate() {
    //animate from left to right
    if (animateType == AnimateType.TO_RIGHT) {
        if (comShow.getLocation().x < 0) {
            //move the new and existing component to the right
            comShow.setLocation(comShow.getLocation().x + animate, 0);
            comExit.setLocation(comExit.getLocation().x + animate, 0);
        } else {
            // Stop animate
            comShow.setLocation(0, 0);
            timer.stop();
            remove(comExit);
            comExit = comShow;
        }
```

```
    } else {
        if (comShow.getLocation().x > 0) {
            //move the new and existing component to the left
            comShow.setLocation(comShow.getLocation().x - animate, 0);
            comExit.setLocation(comExit.getLocation().x - animate, 0);
        } else {
            //stop animate
            comShow.setLocation(0, 0);
            timer.stop();
            remove(comExit);
            comExit = comShow;
        }
    }
}
```

The animate method will first get the animation type and the location of the new and existing components. After that, it will move by incrementing its x position with the animation speed (if the animation is from left to right) or decrementing its x position with the animation speed (if the animation is from right to left).

```
//enum for the animation type (to left or to right)
public static enum AnimateType {
    TO_RIGHT, TO_LEFT
}
```

The AnimateType is an enumeration or a group of constants. The aim is to make a datatype that consists of only TO_RIGHT and TO_LEFT AnimateType.

4. Sound Classes

The sound classes in this program are divided into two, which are the MusicPlayer to play the background music, and the SfxPlayer to play the sound effects. Both of them are similar.

a. MusicPlayer

```
public void loadMusic(String filePath) {
    //stop the clip if the clip is not null and still running
    if (clip != null && clip.isRunning()) {
        clip.stop();
    }
    //close the clip if the clip is not null
    if (clip != null) {
        clip.close();
    }
```

```
    try {
        // Open an audio input stream.
        File soundFile = new File(filePath);
        AudioInputStream audioIn = AudioSystem.getAudioInputStream(soundFile);

        // Get a sound clip resource.
        clip = AudioSystem.getClip();

        // Open audio clip and load samples from the audio input stream.
        clip.open(audioIn);

        // Get the volume control from the clip
        volumeControl = (FloatControl) clip.getControl(FloatControl.Type.MASTER_GAIN);

    } catch (UnsupportedAudioFileException | IOException | LineUnavailableException e) {
        e.printStackTrace();
    } catch (IllegalArgumentException e) {
        System.err.println("Volume control not supported.");
    }
}
```

The loadMusic method is the most crucial method for both MusicPlayer and SfxPlayer as it will load the audio file. It will first stop and close the existing clip if it is not null. After that, it will load the audio file based on the filePath and get the clip. It will also open the audio clip, load samples from it, and get the volume control.

```
public void play() {
    //if the clip is not null, play the clip from the beginning
    if (clip != null) {
        clip.setFramePosition(0);
        clip.start();
        clip.loop(Clip.LOOP_CONTINUOUSLY); // Loop the clip continuously
    }
}
```

Meanwhile, the play method will play the clip from the beginning and loop it continuously.

b. SfxPlayer

SfxPlayer is highly similar to background music although there are some differences.

```
public void loadSound(String filePath, float volume) {
    try {
        // Open an audio input stream.
        File soundFile = new File(filePath);
        AudioInputStream audioIn = AudioSystem.getAudioInputStream(soundFile);

        // Get a sound clip resource.
        clip = AudioSystem.getClip();

        // Open audio clip and load samples from the audio input stream.
        clip.open(audioIn);
```

```
            // Get the volume control from the clip
            volumeControl = (FloatControl) clip.getControl(FloatControl.Type.MASTER_GAIN);
            setVolume(volume);

        } catch (UnsupportedAudioFileException | IOException | LineUnavailableException e) {
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
            System.err.println("Volume control not supported.");
        }
    }

    public void play() {
        //if clip is not null
        if (clip != null) {
            clip.setFramePosition(0); //start from beginning
            clip.start();
        }
    }
```

Same as loadMusic in MusicPlayer, loadSound in SfxPlayer will first get the clip from the audio file specified by the file path. After that, it will get the volumeControl. However, here, it will also set the volume based on the inputted volume. Meanwhile, for the play method, the SfxPlayer does not loop the audio file continuously as it is only a sound effect.

5. Frames

Below are the frames that are available in this program. I will focus more on explaining the operations behind the frame (how to add, update, or others), while the basic GUI, such as initializing the JLabel or other similar things will not be explained.

Because most of the frames used Absolute Layout, then to add the component inside the frame, we will use the below syntax: `getContentPane().add(component, new org.netbeans.lib.awtextra.AbsoluteConstraints(x, y, -1, -1));` where the component is the name of the component, x, and y is the x and y position relative to the frame.

a. WelcomePage

WelcomePage is only a simple page with several labels and two buttons, which are the login and signup buttons. If the user clicks the login button, they will be redirected to the login page, and so does it for the signup button.

```
signUpButton.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) { //when clicked
        setVisible(false);
        new Signup(musicPlayer).setVisible(true);
    }
```

```
//login button mouse listeners
loginButton.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) { //when clicked
        setVisible(false);
        new Login(musicPlayer).setVisible(true);
    }
```

```
public void run() {
    MusicPlayer player = new MusicPlayer("src/App/sound/EnchantingBedtimeStories.wav", 0.9f);
    player.play();
    new WelcomePage(player).setVisible(true);
}
```

The most crucial thing in the WelcomePage is this run method. Only the WelcomePage frame can be runnable and has the main method. The others do not have it. The WelcomePage then has to initialize the MusicPlayer and play the background music. The initial background music for WelcomePage, Login, and Signup is Enchanting Bedtime Stories and it will be changed when the user goes to their homepage.

b. SignUp

The Sign Up page is used to let the user create a new account for SchedNara so that when they open SchedNara next time, they only need to log in. The Sign Up page consists of three fields, which are for the username, password, and password confirmation.

```
//valid password pattern is at least 8 characters containing both alphabet and numbers
public static final Pattern VALID_PASSWORD_REGEX =
Pattern.compile("^(?=.*[A-Za-z])(?=.*\\d)[A-Za-z\\d]{8,}$", Pattern.CASE_INSENSITIVE);

//validate the password by checking whether it matches the regex pattern or not
public static boolean validatePassword(String passwordStr) {
    Matcher matcher = VALID_PASSWORD_REGEX.matcher(passwordStr);
    return matcher.matches();
}
```

The validatePassword method is used to validate whether the password meets the password criteria. A valid password has to have at least 8 characters and contains both alphabet and number. If the password format is not correct, then it will return false.

```
//set the click action for hide and show password
show_pass.addMouseListener(new MouseAdapter() {
    //by default, the password will not be shown
    boolean showPass = false;

    @Override
    public void mouseClicked(MouseEvent e) { //if clicked
        if (!showPass){ //if the passqord is not shown yet
            //set the password to be visible
            show_pass.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/hide_password.png")));
            password_field.setEchoChar((char)0);
            password_field.setFont(new java.awt.Font("Montserrat", Font.PLAIN, 24));
            showPass = true;
        }else{
            //if the password already visible, set it to be hidden again
            show_pass.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/show_password.png")));
            password_field.setEchoChar((char)8226);
            password_field.setFont(new java.awt.Font("Montserrat", 1, 22));
            showPass = false;
        }
    }
});
```

The password field and password confirmation field have the view or hide icon. The above code is used for the view and hides icon behavior in the password field. The view and hide icon for the password confirmation field is also the same. If the icon is clicked and the password is still hidden, then make the password unhidden (set the text to be viewable). Conversely, if the user wants to hide it again, then when clicking the previous icon, all the text will automatically be converted to ... again.

```
username_field.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // When "Enter" is pressed in username_field, move focus to password_field
        password_field.requestFocusInWindow();
    }
});
```

Users can press the ‘Enter’ key to navigate when filling each field. The above code shows that when ‘Enter’ is pressed in the username field, the focus will be moved to the password field. Meanwhile, when it is pressed in the password field, the focus will move to the confirmation field and if it is pressed in the confirmation field, the submit button will be performed.

```
private void backBtnActionPerformed(java.awt.event.ActionEvent evt) {
    //back to the welcome page when back button is pressed
    setVisible(false);
    new WelcomePage(musicPlayer).setVisible(true);
}
```

The above method is used when the user navigates back to the Welcome Page.

```
private void recolorField(JTextField field, JLabel label){
    //get text of the field
    String text = field.getText();
    //if the text is empty, set the color to be red
    if(text.trim().isEmpty()){
        label.setForeground(Color.red); //red text color
        setBottomBorder(field, 255, 0, 0); //red border
        field.setForeground(Color.red); //red text color
    }
    else{
        // if not, then like default, it is black (text and border)
        label.setForeground(Color.black);
        setBottomBorder(field, 0, 0, 0);
        field.setForeground(Color.black);
    }
}
```

The recolorField method is used to check and color the fields based on the user input. Every field has a document listener to check if there is an add, remove, or change in the field, then it will check the field content. One of these check methods is achieved through the recolorField method. It will check whether the field is empty or not, if it becomes empty, then the color of the field will become red, while by default it is black.

```
private void checkPassConfirm(){
    //get the text in password and confirm password field
    String text = password_field.getText();
    String textConfirm = confirm_field.getText();

    //if both text is not the same
    if(!(text.equals(textConfirm))){
        //set both fields border and text color to be red
        passwordtxt.setForeground(Color.red);
        setBottomBorder(password_field, 255, 0, 0);
        password_field.setForeground(Color.red);

        confirmtxt.setForeground(Color.red);
        setBottomBorder(confirm_field, 255, 0, 0);
        confirm_field.setForeground(Color.red);
    }
    else if(!(validatePassword(text))){
        //if the password is not valid
        //set the password border and text to be red
        passwordtxt.setForeground(Color.red);
        setBottomBorder(password_field, 255, 0, 0);
        password_field.setForeground(Color.red);
    }
    else{
        //set the color to be default again, which is black
        //for password field
        passwordtxt.setForeground(Color.black);
        setBottomBorder(password_field, 0, 0, 0);
        password_field.setForeground(Color.black);

        //for confirm field
        confirmtxt.setForeground(Color.black);
        setBottomBorder(confirm_field, 0, 0, 0);
        confirm_field.setForeground(Color.black);
    }
}
```

Meanwhile, for the password and confirmation fields, there is an additional method to check them besides the recolorField. It will check whether the password and the password confirmation are the same or not and whether the password is valid or not. If it is not equal or the password is not valid, then set the corresponding field red.

```
private void submitBtnActionPerformed(java.awt.event.ActionEvent evt) {
    //get the data from the textfields
    String username_str = username_field.getText();
    String pass_str = password_field.getText();
    String confirm_str = confirm_field.getText();

    //validation
    if(username_str.trim().isEmpty()){ //if username is still empty
        JOptionPane.showMessageDialog(getContentPane(), "Username is still empty.");
    }
    else if(pass_str.trim().isEmpty()){ //if password is still empty
        JOptionPane.showMessageDialog(getContentPane(), "Password is still empty.");
    }
    else if(confirm_str.trim().isEmpty()){ //if confirm password is still empty
        JOptionPane.showMessageDialog(getContentPane(), "Confirm Password is still empty.");
    }
    else{
       if(!(pass_str.equals(confirm_str))){ //if password and confirm password is not the same
           JOptionPane.showMessageDialog(getContentPane(), "Password and Confirm Password is not the same.");
       }
       else{
           if(!(validatePassword(pass_str))){ //if password pattern is not valid
                JOptionPane.showMessageDialog(getContentPane(), "Password must have "
                        + "8 characters with at least one number and one character");
           }
       }
    }
```

The above method is for when the submit button is pressed or performed. It will first validate the user input (if it is empty or not), then it will validate the password and confirmation field (if it is equal or not), and then the password (if it is valid or not).

```
if(pass_str.equals(confirm_str) && validatePassword(pass_str)){ //if all requirements met
    //get today day name
    String day = LocalDate.now().getDayOfWeek().name();
    String day_name = day.charAt(0) + day.substring(1,3).toLowerCase();
    try{
        //set ID first
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
        ResultSet rs = st.executeQuery("SELECT COUNT(userID) FROM user");
        String idStr ="";
        if(rs.first()){
            //if there exists other users before you, your ID will be the last ID + 1
            String id = rs.getString(1);
            int idInt = Integer.parseInt(id);
            idInt = idInt+1;
            idStr = "u" + String.valueOf(idInt);
        }
        else{
            idStr = "u1"; //if you are the first user
        }

        //insert into database
        String sql = "INSERT INTO user VALUES(?,?,?,?,?,?,?,?,?)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, idStr);
        ps.setString(2, username_str);
        ps.setString(3, pass_str);
        ps.setString(4, "arama");
        ps.setInt(5, 0);
```

```
            ps.setInt(6, 0);
            ps.setInt(7, 0);
            ps.setString(8, day_name);
            ps.setInt(9, 0);

            ps.executeUpdate();

            updateTaskCompletions(idStr);

            //after sign up, auto redirect to the homepage
            setVisible(false);

            //because it is their first signup, then the default aranara will be arama
            //set the background music to play arama song
            musicPlayer.loadMusic("src/App/sound/MelodyofHiddenSeeds.wav");
            musicPlayer.play();
            new HomePage(idStr, musicPlayer).setVisible(true);

        }catch(Exception e){
            e.printStackTrace();
        }
```

If all conditions are met, then it will generate an ID for the user based on the last ID in the table and insert the user data into the user table. The default aranara will be set to Arama and all Aranara affections are still 0. The pat amount is also 0. After that, it will also update the task completion table to insert the new user data into it. After that, it will redirect the user to the homepage with the default song (Arama song) because the default Aranara for all users is Arama.

```
private void updateTaskCompletions(String userID){
    //get today date
    LocalDate currentDate = LocalDate.now();
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    // Convert the date to string using the custom format
    String dateString = currentDate.format(formatter);

    //update the task completion database
    try{
        //insert all value to be 0 to the task completon database
        //because the user is new and does not complete any task before
        Connection con = ConnectionProvider.getCon();
        String query = "INSERT INTO task_completion VALUES(?,?,?,?,?,?,?,?,?)";

        PreparedStatement pst = con.prepareStatement(query);
        pst.setString(1, userID);
        pst.setInt(2, 0);
        pst.setInt(3, 0);
        pst.setInt(4, 0);
        pst.setInt(5, 0);
        pst.setInt(6, 0);
        pst.setInt(7, 0);
        pst.setInt(8, 0);
        pst.setString(9, dateString);

        pst.executeUpdate();

    }catch(Exception e){
        e.printStackTrace();
    }
```

The updateTaskCompletions method is used to update the task_completion table in the database. It will set all task completions for the user this week to 0 because the user is still new and set the last login date to be today's date.

c. Login

The login frame is highly similar to the signup page. The difference is that the login page does not have the password confirmation field so it does not have to check the password confirmation. Besides that, when the user presses the 'Enter' key in the password field, the submit button will be performed. Other methods are similar to the one on the Signup page. The two very distinct methods are the method when the user submits the page and the check password method.

```
private void checkPass(){
    //get password text
    String text = password_field.getText();
    if(!(validatePassword(text))){ //if it is not valid
        //set the password field border and text to be red
        passwordtxt.setForeground(Color.red);
        setBottomBorder(password_field, 255, 0, 0);
        password_field.setForeground(Color.red);
    }
    else{
        //if not, set back to default (black text and border)
        passwordtxt.setForeground(Color.black);
        setBottomBorder(password_field, 0, 0, 0);
        password_field.setForeground(Color.black);
    }
}
```

As explained before, the checkPass method now only needs to check whether the password is valid or not and set the password field to red if it is not valid.

```
private void submitBtnActionPerformed(java.awt.event.ActionEvent evt) {
    //get the data from the textfields
    String username_str = username_field.getText();
    String pass_str = password_field.getText();

    //validation
    if(username_str.trim().isEmpty()){ //if username is empty
        JOptionPane.showMessageDialog(getContentPane(), "Username is still empty.");
    }
    else if(pass_str.trim().isEmpty()){ //if password is empty
        JOptionPane.showMessageDialog(getContentPane(), "Password is still empty.");
    }
    else{
        if(!(validatePassword(pass_str))){ //if password is not valid
            JOptionPane.showMessageDialog(getContentPane(), "Password must have 8 "
                    + "characters with at least one number and one character");
        }
    }
```

When submitting the data, the method will also only check the password whether it is valid or not and display the appropriate message if it is not valid.

```
if(validatePassword(pass_str)){ //if password is valid
    try{
        //search the username in database
        Connection con = ConnectionProvider.getCon();
        String query = "SELECT * FROM user WHERE username = ?";

        PreparedStatement ps = con.prepareStatement(query);
        ps.setString(1, username_str);

        try(ResultSet rs = ps.executeQuery()){

            if (rs.next()){
                //get the password and userID based on the username
                String pass = rs.getString("password");
                String id = rs.getString("userID");

                //check password
                if (pass_str.equals(pass)){ //if the password is the same
                    //get the default aranara
                    String default_aranara = rs.getString("default_aranara");
                    String bgmPath = "src/App/sound/";

                    //get the background music based on the default aranara
                    if (default_aranara.equals("arama")){
                        bgmPath += "MelodyofHiddenSeeds.wav";
                    }
                    else if (default_aranara.equals("ararycan")){
                        bgmPath += "IveNeverForgotten.wav";
                    }else if (default_aranara.equals("arabalika")){
                        bgmPath += "ForRiddlesForWonders.wav";
                    }

                    //load the background music and go to homepage
                   setVisible(false);
                   musicPlayer.loadMusic(bgmPath);
                   musicPlayer.play();
                   new HomePage(id, musicPlayer).setVisible(true);
                }
                else{
                    //if password is not correct
                    JOptionPane.showMessageDialog(getContentPane(), "Password is incorrect.");
                }
            }else{
                //if user is not in the database
                JOptionPane.showMessageDialog(getContentPane(),"User is not available.");
```

If the password is valid, it will then check whether such a user exists based on the username. If yes and the password in the database is the same as the one inputted by the user, then the method will get the default Aranara of the user and play background music based on the default Aranara.

d. HomePage

The homepage is the landing page when the user first signs up or logs in to SchedNara. It consists of several contents, like the default Aranara picture, the task completion bar chart, a random quote, and the upcoming task for today.

```
private void inputQuote() {
    try {
        //read the Quotes file
        FileReader fr =new FileReader("src/App/Quotes.txt");
        BufferedReader reader = new BufferedReader(fr);
        String eachLine;
        //while reading each line
        while ((eachLine = reader.readLine()) != null) {
            //split each line by the '_' symbol
            String[] part = eachLine.split("_");
            if (part.length == 2) {
                quotes[count] = part[0]; // the first part is the quote
                by[count] = part[1]; // the second part is the person who said the quote
                count++;
            } else {
                System.out.println("Invalid format in line: " + eachLine);
            }
        }
        reader.close();

    } catch (IOException e) {
        //exception if file is not available
        System.out.println("Cannot find file. Please try again.");
    }
}

private void setRandomQuote(){
    //input the quote into the array
    inputQuote();
    //randomize quote choice
    Random random = new Random();
    int index = random.nextInt(80);

    //set and display the quote
    quotetxt.setText(quotes[index]);
    quotetxt.setHorizontalAlignment(SwingConstants.CENTER);
    quoteby_txt.setText("~" +by[index]);
    quoteby_txt.setHorizontalAlignment(SwingConstants.RIGHT);
}
```

The inputQuote method is used to input all the quotes in the txt file to the array on the homepage. The first array is the quote, while the second array is the person who said that quote. Meanwhile, after getting the quotes, the setRandomQuote method will randomize the quote choice and display it on the homepage.

```
//handle hover button in the navbar
public void hoverButton(String image_path, int colorR, int colorG, int colorB, JLabel[] labels){
    for (JLabel label : labels){
        if (label.getIcon() != null){
            label.setIcon(new javax.swing.ImageIcon(getClass().getResource(image_path)));
        }else{
            label.setForeground(new java.awt.Color(colorR, colorG, colorB));
        }
    }
}
```

A button inside the side navigation bar consists of an image label and a text label that is grouped in an array. Hence, to ease the hover behavior, the hoverButton method will get the array and set the icon if it is an image label and the color if it is a text label.

```
private boolean isNewWeek(LocalDate referenceDate, LocalDate currentDate) {
    // Get the week fields for the default locale
    WeekFields weekFields = WeekFields.of(Locale.getDefault());

    // Get the week number of the reference date and current date
    int referenceWeek = referenceDate.get(weekFields.weekOfWeekBasedYear());
    int currentWeek = currentDate.get(weekFields.weekOfWeekBasedYear());

    // Get the year of the reference date and current date
    int referenceYear = referenceDate.get(weekFields.weekBasedYear());
    int currentYear = currentDate.get(weekFields.weekBasedYear());

    // Check if the year and week number are different
    return currentYear > referenceYear || (currentYear == referenceYear && currentWeek > referenceWeek);
}
```

The isNewWeek method is used to check whether the reference date and the current date are on the same week or not. It will check it by getting the week number and the year number of the dates.

```
private void resetTaskCompletions(boolean isReset, LocalDate todayDate){
    //convert the date to String
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    String todayStr = todayDate.format(formatter);

    try{
        Connection con = ConnectionProvider.getCon();

        if (isReset){
            //reset the task completion database weekly
            String query = "UPDATE task_completion SET Mon = ?, Tue = ?, Wed = ?, Thu = ?, Fri = ?, Sat = ?, Sun = ?, "
                    + "last_login = ? WHERE userID = ?";
            //set all the task completion to be 0 again
            PreparedStatement ps = con.prepareStatement(query);
            ps.setInt(1, 0);
            ps.setInt(2, 0);
            ps.setInt(3, 0);
            ps.setInt(4, 0);
            ps.setInt(5, 0);
            ps.setInt(6, 0);
            ps.setInt(7, 0);
            ps.setString(8, todayStr); //set today as the last login
            ps.setString(9, userID);
            ps.executeUpdate();

        }else{
            //only set today as the last login
            String query = "UPDATE task_completion SET last_login = ? WHERE userID = ?";
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, todayStr);
            ps.setString(2, userID);
            ps.executeUpdate();
        }

    }catch(Exception e){
        JOptionPane.showMessageDialog(getContentPane(), e);
        e.printStackTrace();
    }
}
```

The resetTaskCompletions method is used to reset all the task completions (to become 0 again) in the task completion table if it is in a new week. After that,

regardless, of whether it is a new week or not, it will also set the last login date as today's date.

```
private void queryTaskCompletions(){
    try{
        //query from the database
        Connection con = ConnectionProvider.getCon();
        String query = "SELECT * FROM task_completion WHERE userID = ?";

        PreparedStatement pst = con.prepareStatement(query);
        pst.setString(1, this.userID);

        try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                //get today date and last login date
                String last = rs.getString("last_login");
                LocalDate today = LocalDate.now();
                LocalDate last_login = convertStrDate(last);

                //get task completions for this week
                int amtMon = rs.getInt("Mon");
                int amtTue = rs.getInt("Tue");
                int amtWed = rs.getInt("Wed");
                int amtThu = rs.getInt("Thu");
                int amtFri = rs.getInt("Fri");
                int amtSat = rs.getInt("Sat");
                int amtSun = rs.getInt("Sun");

                //if it is a new week, then reset the task completion in the database
                if (isNewWeek(last_login, today)){
                    resetTaskCompletions(true, today);
                    amtMon = 0;
                    amtTue = 0;
                    amtWed = 0;
                    amtThu = 0;
                    amtFri = 0;
                    amtSat = 0;
                    amtSun = 0;
                }else{
                    //do not reset the task completion, only reset the last login
                    resetTaskCompletions(false, today);
                }

                //put the completion rate to the hashmap to ease chart making
                completionRate.put("Mon", amtMon);
                completionRate.put("Tue", amtTue);
                completionRate.put("Wed", amtWed);
                completionRate.put("Thu", amtThu);
                completionRate.put("Fri", amtFri);
                completionRate.put("Sat", amtSat);
                completionRate.put("Sun", amtSun);
            }
```

The queryTaskCompletions method is used to populate the completionRate HashMap that will be used to set the bar chart at the next process. It will first get today's date and check whether it is the same week as the last login date or not. If not, then it will resetTaskCompletions and set all task amounts to 0. After that, it will populate the HashMap based on the task amounts.

```
private void showBarChart(){
    //show the bar chart
    queryTaskCompletions(); //query the task completion of the user

    //set the dataset based on the completion rate amount of each day
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    dataset.setValue(completionRate.get("Mon"), "Amount", "Mon");
    dataset.setValue(completionRate.get("Tue"), "Amount", "Tue");
    dataset.setValue(completionRate.get("Wed"), "Amount", "Wed");
    dataset.setValue(completionRate.get("Thu"), "Amount", "Thu");
    dataset.setValue(completionRate.get("Fri"), "Amount", "Fri");
    dataset.setValue(completionRate.get("Sat"), "Amount", "Sat");
    dataset.setValue(completionRate.get("Sun"), "Amount", "Sun");

    //create a bar chart
    JFreeChart chart = ChartFactory.createBarChart("Task Completion","Day","Amount",
            dataset, PlotOrientation.VERTICAL, false,true,false);

    //make the chart looks flatter
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    categoryPlot.setRangeGridlinePaint(Color.BLUE);
    categoryPlot.setBackgroundPaint(Color.WHITE);
    BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
    Color color = new Color(31, 139, 217);
    renderer.setSeriesPaint(0, color);
    renderer.setBarPainter(new StandardBarPainter()); // Disable gradient effect
    renderer.setMaximumBarWidth(0.08);

    //draw the x axis labels (the day name)  by iterating over the key
    for(int i=1; i<= completionRate.size(); i++){
        for(String key: completionRate.keySet()){
            if (completionRate.get(key) == 0){
                NumberAxis rangeAxis = (NumberAxis) categoryPlot.getRangeAxis();
                rangeAxis.setTickUnit(new NumberTickUnit(1));
                rangeAxis.setRange(0.0, 8.0);
            }
        }
    }

    //create the chart
    ChartPanel barChart = new ChartPanel(chart);
    barChart.setPreferredSize(new Dimension(1000, 400));
    //remove the old chart (if exists) and set the new chart inside the panel
    barChartPanel.removeAll();
    barChartPanel.add(barChart, BorderLayout.CENTER);
    barChartPanel.validate();
}
```

The showBarChart method is used to show the bar chart in the chart panel based on the task completion amount that is populated before in the completionRate HashMap. It will set the dataset based on the task amount in the HashMap, then it creates a flat blue chart and sets the x-axis labels as the day name (the completionRate HashMap keys). After that, it will create the bar chart in the panel. But, before that, all the old charts inside the panel will be removed, so that only the new chart exists.

```
private void setAnalysisTxt(){
    //set text about how many tasks that the user has completed today
    //get today day name
    String day = LocalDate.now().getDayOfWeek().name();
    String day_name = day.charAt(0) + day.substring(1,3).toLowerCase();
    int todayCompleted = completionRate.get(day_name); //get task amount

    //set the message
    if (todayCompleted == 0){
        analyze_task_txt.setText("You haven't done any task today, Nara.");
    }else if (todayCompleted ==1){
        analyze_task_txt.setText("You have completed " + todayCompleted + " task today, Nara.");
    }
    else{
        analyze_task_txt.setText("You have completed " + todayCompleted + " tasks today, Nara.");
    }
}
```

The setAnalysisTxt method is used to set the text below the bar chart panel about the amount of task completion that the user has completed today. It will first get today's day of name and get the completion rate based on the day name. Then, it will display it in the JLabel.

```
private void setTodayTasks(){
    //set upcoming task for today
    String task_name = "";
    String task_date_from = "";
    String task_date_to = "";
    String task_type = "";
    LocalDate currentDate = LocalDate.now(); //get today date

     try{
        //query from the task table in the database
        Connection con = ConnectionProvider.getCon();
        String query = "SELECT * FROM tasks WHERE userID = ? ORDER BY completed ASC";

        PreparedStatement pst = con.prepareStatement(query);
        pst.setString(1, this.userID);

        try (ResultSet rs = pst.executeQuery()) {
            while (rs.next()) {
                String name = rs.getString("name");
                String type = rs.getString("type");
                String dateFrom = rs.getString("timeFrom");
                String dateTo = rs.getString("timeTo");
                boolean comp = rs.getBoolean("completed");
```

The setTodayTasks method is used to set the upcoming task for today. It will first get today's date and query from the task table in the database ascending by completed, so that the uncompleted task will be displayed first. It will get all the data about the task.

```
    //check if it is completed or not, if it is not completed yet
    if (!comp){
        if (type.equals("One-day event")){ //if it is one day event
            //check the date from, if same, then set the task that want to be displayed
            if (convertStrDate(dateFrom).equals(currentDate)){
                task_name = name;
                task_type = "One-day event";
                task_date_from = dateFrom;
                break; //break the loop
            }
         //multiple day event
        }else{
            //check the date range
            //if the date is inside the range, then set this task to be displayed
            if (currentDate.isAfter(convertStrDate(dateFrom).minusDays(1))
                    && currentDate.isBefore(convertStrDate(dateTo).plusDays(1))){
                task_name = name;
                task_type = "Multiple-day event";
                task_date_from = dateFrom;
                task_date_to = dateTo;
                break; //break the loop
            }
        }
    }
}

//set the nearest task and nearest time label based on the result
//convertDate method is used to change 2024-05-06 to 6 May 2024
if (task_type.equals("One-day event")){
    nearest_task.setText(task_name);
    nearest_time.setText(convertDate(task_date_from));
}else if (task_type.equals("Multiple-day event")){
    nearest_task.setText(task_name);
    String time = convertDate(task_date_from) + " to " + convertDate(task_date_to);
    nearest_time.setText(time);
}else{
    nearest_task.setText("No task available today.");
    nearest_time.setText("Try checking out other tasks in the calendar!");
}
```

If it is not completed yet, is a one-day event, and the dateFrom equals today's date, then set this task as the upcoming task and break the unnecessary loop. Otherwise, if it is not completed yet, is a multiple-day event, and today's date is in range of the dayFrom and dayTo range, then set this task as the upcoming task and break the unnecessary loop. After that, display the task name and time based on its type. If there is no uncompleted task, then set the label to say that there is no task for today.

Other than these methods, there are also some methods to control the navigation of each button. There are five buttons in the side navigation bar, which are the home, add workflow menu, calendar, Aranara menu, and logout button. The home button is not clickable as this is on the homepage, the add workflow menu button will direct you to the add workflow menu, the calendar to the calendar page, the Aranara to the Aranara

menu, and the logout button to the Welcome Page and play the Welcome Page's background music. Other buttons on the homepage are the view more button below the upcoming tasks for today label and the enlarge button beside the default Aranara picture. The view more button will direct you to the calendar page, while the enlarge or new window button to the Aranara menu.

e. AddWorkflowMenu

Same as HomePage, the Add Workflow Menu also has a side navigation bar that contains the same five buttons. However, this time, the one that can not be clicked is the ass workflow menu button. Other buttons can be clicked and will redirect the user to their corresponding menu. Hence, the hoverButton method inside this menu is also the same as the one on the homepage.

```
private void myinit(){
    //query all the workflow
    queryWorkflow();
    // Create the content pane
    contentPane = new JPanel() {
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            // Load the background image
            ImageIcon bgImage = new ImageIcon("src/App/img/default_page.png");
            // Draw the background image
            g.drawImage(bgImage.getImage(), 0, 0, getWidth(), getHeight(), null);
        }
    };
    contentPane.setLayout(null); // Use absolute layout
    setContentPane(contentPane);

    // Create the scroll pane
    scrollPane = new JScrollPane();
    scrollPane.setBounds(180, 180, 1054, 455); // Set bounds for the scroll pane
    scrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
    scrollPane.setBorder(null);
    scrollPane.getVerticalScrollBar().setUnitIncrement(16);
    contentPane.add(scrollPane);

    // Create the cloneable panel
    cloneablePanel = new JPanel(); // The initial panel inside scroll pane
    cloneablePanel.setLayout(null); // Use absolute layout
    cloneablePanel.setPreferredSize(new Dimension(400, 200)); // Set initial size
    cloneablePanel.setBounds(180, 200, 1200, 1500); // Set bounds for the initial panel
    cloneablePanel.setBackground(Color.white);
    scrollPane.setViewportView(cloneablePanel); // Set this panel as viewport's view

    //create the cloneable panels and the add panel
    createClonedPanels(workflowList, workflowList.size());
    createAddPanel();

    initDesign(); //initialize all the design components
    initHover(); //initialize the hovering method for buttons

    //set the addworkflowbutton to be focused when opening this page
    //so that the textfield is not focused first
    SwingUtilities.invokeLater(() -> addWorkflowBtn.requestFocusInWindow());
```

The myinit method will first call the queryWorkflow method. The queryWorkflow method is basically a method that queries all the workflow inside the database and inserts it into a Linked List. The myinit method will then initialize a content pane for the frame and create a scroll pane inside the content pane. The scroll pane will be used to scroll the workflow panels. After that, myinit will create all cloned panels based on the workflow list and also create the add panel on the first order inside the scroll pane. It will also initialize the design and hovering methods and set addWorkflowBtn to be focused, so that the initial focus is not on the search field, so that user knows what is the search field for.

```
public void createClonedPanels(LinkedList<Workflow> list, int totalElement){
    //get this frame as home or parent frame
    AddWorkflowMenu home = (AddWorkflowMenu) SwingUtilities.getRoot(this);
    int row=0, column=0;
    for(int i=0; i<totalElement;i++){
        //for every workflow in the list
        String id = list.get(i).getId();
        String title = list.get(i).getTitle();
        int checkpoint = list.get(i).getCheckpoint();

        // Create a new cloned panel
        CloneablePanelWorkflow clonedPanel = new CloneablePanelWorkflow(20, Color.white, 2 ,id, title, checkpoint, home);
        // Set width and height for the cloned panel
        int panelWidth = 292;
        int panelHeight = 278;

        // Calculate the row and column indices
        //because each row only contains three workflows and the first row has the add panel
        if (i == 0 || i == 1){
            column = (i % 3) +1; //+1 because of the add panel
            row = 0;
        }else{
          column = (i-2) % 3;  //-2 is for the panel0 and panel1
          row = (i+1)/3; //+1 because the first row is for the add panel, panel0, and panel1
        }

        // Calculate the x and y positions based on row and column indices
        int x = 10 + column * (panelWidth + 50);
        int y = 10 + row * (panelHeight + 50);

        // Set the bounds for the cloned panel with your custom size
        clonedPanel.setBounds(x, y, panelWidth, panelHeight);
        clonedPanel.setBackground(Color.white);

        // Add the cloned panel to the initial panel
        cloneablePanel.add(clonedPanel);
        // Adjust preferred size of initial panel to include new panel
        Dimension newSize = new Dimension(cloneablePanel.getWidth(), y + panelHeight + 10); // Adjusted size
        cloneablePanel.setPreferredSize(newSize);
        // Ensure the scroll pane updates its viewport
        scrollPane.revalidate();
        scrollPane.repaint();
        // Scroll to show the new panel
        scrollPane.getVerticalScrollBar().setValue(0);
    }
}
```

The createClonedPanels method can be seen on other menus when they also use cloneable panels. In this case, it will create the CloneablePanelWorkflow. First, the method will iterate over the list of workflow and create a cloned panel based on the workflow. After that, because in this menu, one row has only three workflows and the

first row also contains the add panel, we need to calculate the indices first. After that, the method will calculate the x and y positions based on the column and row positions for the cloned panel. Finally, the cloned panel will be added to the cloneable panel that contains the cloned panel and the scroll pane will be revalidated.

```
public void createAddPanel(){
    //intialize the add panel,
    JPanel add_panel = new JPanel();
    //add panel width, height, x, and y
    int panelWidth = 292;
    int panelHeight = 278;
    int add_panel_x = 10;
    int add_panel_y = 10;
    add_panel.setBounds(add_panel_x, add_panel_y, panelWidth, panelHeight);
    add_panel.setLayout(null);

    //add_panel color and border
    add_panel.setBackground(new Color(246, 252, 254));
    float[] dashPattern = {10, 10}; // 10 pixels on, 10 pixels off
    BasicStroke dashedStroke = new BasicStroke(3, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND, 10, dashPattern, 0);
    add_panel.setBorder(BorderFactory.createStrokeBorder(dashedStroke, new Color(31,139,217)));

    //label inside the add panel
    JLabel createtxt = new JLabel();
    createtxt.setFont(new java.awt.Font("Montserrat Medium", 0, 28)); // NOI18N
    createtxt.setForeground(new java.awt.Color(167, 204, 231));
    createtxt.setText("Create New");
    createtxt.setBounds(57, 172, 235, 34);
    add_panel.add(createtxt);

    //the add icon inside the add panel
    JLabel add_icon = new JLabel();
    add_icon.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/add_workflow_icon.png")));
    add_icon.setBounds(102, 77, 79, 78);
    add_panel.add(add_icon);
```

```
//if add_panel get selected
AddWorkflowMenu home = (AddWorkflowMenu) SwingUtilities.getRoot(this);
add_panel.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        //if there is no window ipen
        if (open == 0){
            //open create new workfloe menu
           AddWorkflowMenu.open = 1;
            new CreateNewWorkflow(userID, home).setVisible(true);
        }
        else{
         JOptionPane.showMessageDialog(getContentPane(), "One window is already open.");
        }
    }
    @Override
    public void mouseEntered(MouseEvent e) { //when hovered
        createtxt.setForeground(new java.awt.Color(31, 139, 217));
        add_icon.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/add_workflow_icon_hover.png")));
    }

    @Override
    public void mouseExited(MouseEvent e) { //when not hovered
        createtxt.setForeground(new java.awt.Color(167, 204, 231));
        add_icon.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/add_workflow_icon.png")));
    }
});

//add the add panel to the cloneable panel
cloneablePanel.add(add_panel);
scrollPane.revalidate();
scrollPane.repaint();
// Scroll to show the new panel
scrollPane.getVerticalScrollBar().setValue(0);
```

The createAddPanel method is used to create the add panel on the leftmost and topmost positions of the scroll pane. The add panel has a dashed border. It also contains

two labels which are the text and the + icon that can change color when the add panel is hovered. If the add panel is clicked, then it will show a new frame on top of the Add Workflow Menu frame to let the user create a new workflow. The add panel is then added to the scroll pane based on its position the position of the add panel is always constant because it will be always on the top-left of the scroll pane.

```
private void handleSearch(){
    //get the text of the search field
    String searchStr = search_field.getText();
    LinkedList<Workflow> resultList= new LinkedList<>();
    boolean exist = false; //track whether exists the result or not

    for (int i = 0; i < workflowList.size(); i++){
        //iterate over the workflow list
        String want_to_be_check = workflowList.get(i).getTitle().toLowerCase();
        //if the searched string equals to the title of the workflow
        if (want_to_be_check.contains(searchStr.toLowerCase())){
            //append it to the result list
            resultList.add(workflowList.get(i));
            exist = true;
        }
    }

    if (exist == true){
         //reset the cloneable panel that contains the cloned panel
        cloneablePanel.removeAll();
        //create the add panel and the cloned panels based on the result
        createClonedPanels(resultList, resultList.size());
        createAddPanel();
    }
    else{
        //reset the cloneable panel and only create the add panel
        cloneablePanel.removeAll();
        createAddPanel();
    }
}
```

Meanwhile, the handleSearch method is used to display only the result of the searched String. It will first get the text from the search field and check every workflow in the workflow list whether they have the same name as the searched String or not. If yes, then it will display only the workflows that have the searched String in their title and the add panel. If no result exists, then it will only create the add panel.

```
public void goToEdit(String workflowID){
    //go to edit workflow menu
    setVisible(false);
    new EditWorkflow(workflowID, userID, player).setVisible(true);
}

public void reload(){
    //reload the add workflow menu
    setVisible(false);
    new AddWorkflowMenu(this.userID, this.player).setVisible(true);
}
```

The goToEdit is used to direct the user to the Edit Workflow Menu, while the reload method is used to refresh the Add Workflow Menu frame.

f. CreateNewWorkflow

The CreateNewWorkflow frame is supposed to be a child frame that is on top of the Add Workflow Menu frame. It only consists of a field, some labels, and an OK button. This frame's purpose is to let the user add a new workflow to the existing workflows.

```
//set the close operation
setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
addWindowListener(new WindowAdapter() {
    @Override
    public void windowClosing(WindowEvent e) {
        // when pressing the X button, ask for xonfirmation first
        int option = JOptionPane.showConfirmDialog(getContentPane(),
                "Do you really want to go back?", null, JOptionPane.YES_NO_OPTION);
        if (option == JOptionPane.YES_OPTION) {
            //if yes, then set unvisible and
            //make the opened window in AddWorkflowMenu to become 0 again
            setVisible(false);
            AddWorkflowMenu.open=0;
        }
    }
});
```

Because it appears on top of the add workflow menu frame and is considered a child frame, not an independent frame, then when the user presses the 'X' button on the frame, it will not exit the program. It will then set itself to be invisible again and the user can see the add workflow menu.

```
private void OKbuttonActionPerformed(java.awt.event.ActionEvent evt) {
    //get the ID and name of the new workflow
    String idInput = idtxt.getText();
    String titleInput = nameField.getText();

    if(titleInput.trim().isEmpty()){ //if title is empty
        JOptionPane.showMessageDialog(getContentPane(), "Your workflow name is still empty");
    }
    else{
        try{
            Connection con = ConnectionProvider.getCon();
            Statement st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);

            //insert the ID and name of the new workflow to the database
            PreparedStatement ps = con.prepareStatement("INSERT INTO workflow VALUES(?,?,?,?)");
            ps.setString(1, idInput);
            ps.setString(2, titleInput);
            ps.setInt(3, 0);
            ps.setString(4, userID);
            ps.executeUpdate();

            //display success message
            JOptionPane.showMessageDialog(getContentPane(), "Successfully created a new workflow!");
            //set invisible again and the opened window become 0
            AddWorkflowMenu.open=0;
            setVisible(false);

            home.queryWorkflow();
            home.cloneablePanel.removeAll();
            home.createClonedPanels(home.workflowList, home.workflowList.size());
            home.createAddPanel();

        }catch(Exception e){
            e.printStackTrace();
        }
```

The above method specifies what happens when the OK button is pressed. The method will first get the ID and the name of the new workflow. If the name field is still empty, display the error message. The ID is obtained from the ID label, whereupon initializing this frame, it will first query the last ID of the workflow in the database and generate the ID added by 1 from the one in the database. If all the conditions are met, then the method will insert the new workflow into the database and close this frame.

g. EditWorkflow

The Edit Workflow menu is not in the main menu, like the homepage, add workflow menu, calendar page, and Aranara menu, so it does not have a sidebar. However, it is an independent frame, unlike the CreateNewworkflow frame. It has two buttons that redirect the user to other frames, which are the back button to redirect the user to the add workflow menu and the generate text button to show the text result frame on top of the Edit Workflow frame. Its display is also divided into two parts, the left side which is the edit panel, and the right side which is a view of the current existing flows.

```
private void queryFlow(){
    flowList.clear(); //clear all the existing old flows (if exist)
    try{
        //query flows from the database based on the current workflow ID
        Connection con = ConnectionProvider.getCon();
        String query = "SELECT * FROM flow WHERE workflowID = ? ORDER BY dayFrom ASC";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, this.workflowID);

        ResultSet rs = pstmt.executeQuery();
        while(rs.next()){
            String fID = rs.getString("id");
            String fname = rs.getString("name");
            String ftype = rs.getString("type");
            int fdayFrom = rs.getInt("dayFrom");
            int fdayTo = rs.getInt("dayTo");
            String fnotes = rs.getString("notes");
            String fcolor = rs.getString("color");

            //add it to the flow list
            Flow flow = new Flow(fID, this.workflowID, fname, ftype, fdayFrom, fdayTo, fnotes, fcolor);
            flowList.add(flow);
        }
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(getContentPane(), e);
    }
}
```

The queryFlow method is used to query all the flows in the database into the flow list based on the user ID. It is also sorted ascending based on the dayFrom, so that when the cloned panel is displayed, it will be displayed from the first flow that has to

be done before the D-day until the last flow that has to be done after the D-day. Due to also having cloned panels (CloneablePanelFlow), the Edit Workflow menu also has the myinit and createClonedPanels similar to the one in the Add Workflow menu. The difference is that in this menu, there is no add panel and a row only consists of one panel.

```
private void createClonedPanels(LinkedList<Flow> list, int size){
    //iterate over all the flows inside the flow list
    for(int i=0; i<size;i++){
        String id = list.get(i).getId();
        String name = list.get(i).getNameInput();
        String type = list.get(i).getTypeInput();
        int dayFrom = list.get(i).getDayFromInput();
        int dayTo = list.get(i).getDayToInput();
        String note = list.get(i).getNoteInput();
        String color = list.get(i).getColorInput();

        // Create a new cloned panel
        CloneablePanelFlow clonedPanel = new CloneablePanelFlow(20,
                Color.white, 2 ,id, name, type, dayFrom, dayTo, note, color);
        // Set custom width and height for the cloned panel
        int panelWidth = 288;
        int panelHeight = 146;

        // Calculate the x and y positions
        int x = 110;
        int y = 10 + i * (panelHeight + 50); //count based on panel height

        // Set the bounds for the cloned panel with your custom size
        clonedPanel.setBounds(x, y, panelWidth, panelHeight);
        clonedPanel.setBackground(new Color(246,252,254));
```

```
    //the cloned panel can be selected
    clonedPanel.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            //if it is clicked and the initial current panel is itself
            if (currentPanel == clonedPanel){
                //set the current panel to become null
                currentPanel.setClicked(false);
                currentPanel = null;
                //clear all fields and deactivate delete btn
                clearAllFields();
                deactivateDeleteBtn();
            }
            else{
                //if the selected one is not itself
                if (currentPanel != null) {
                    //set the selected one to be false
                    currentPanel.setClicked(false);
                }
                //set itself to be clicked and be the new current panel
                clonedPanel.setClicked(true);
                currentPanel = clonedPanel;
                handleEditPanel(); //set up the fields inside the edit panel
            }
        }
    });
```

```
        // Add the cloned panel to the initial panel
        cloneablePanel.add(clonedPanel);

        // Adjust preferred size of initial panel to include new panel
        Dimension newSize = new Dimension(cloneablePanel.getWidth(), y + panelHeight + 10); // Adjusted size
        cloneablePanel.setPreferredSize(newSize);
        // Ensure the scroll pane updates its viewport
        scrollPane.revalidate();
        scrollPane.repaint();
        // Scroll to show the new panel
        scrollPane.getVerticalScrollBar().setValue(0);
```

The createClonedPanels is very similar to the one in the Add Workflow Menu, but instead, it takes a Linked List with Flow objects as the input. It also does not calculate the column and row as it is unnecessary. Additionally, the cloned panel can be clicked to be set as the current panel. When the panel is clicked (the current panel is not null), then we set up the fields in the edit panel to contain the information about the Flow that is represented by the panel. If we click a panel that is selected, the current panel will become null again. When this happens, all the fields in the edit panel will be cleared and the delete button will be deactivated.

```
private void handleEditPanel(){
    //set name field
    nameField.requestFocus();
    nameField.setText(currentPanel.getNameInput());

    //set radio button
    if (currentPanel.getTypeInput().equals("One-day event")){
        //if it's one day event, disable the to "fields"
        oneDay.setSelected(true);
        toField.setEnabled(false);
        toComboBox.setEnabled(false);
        repaint();
    }else{
        //otherwise, enable them
        multipleDay.setSelected(true);
        toField.setEnabled(true);
        toComboBox.setEnabled(true);
        repaint();
        //set dayTo field
        handleDayField(currentPanel.getDayToInput(), toComboBox, toField);
    }

    //set day from field
    handleDayField(currentPanel.getDayFromInput(), fromComboBox, fromField);
    //set notes text area and color combo box
    notesArea.setText(currentPanel.getNoteInput());
    colorComboBox.setSelectedItem(currentPanel.getColorInput());

    //activate delete button
    activateDeleteBtn();
}
```

The handleEditPanel method is used to set up the fields inside the edit panel when a panel is selected. It will also activate the delete button.

```
private void deleteFlow(){
    //delete the flow
    //ask for confirmation
    String question = "Do you really want to delete " + currentPanel.getNameInput() + "?";
    int a = JOptionPane.showConfirmDialog(getContentPane(), question, "SELECT", JOptionPane.YES_OPTION);

    if (currentPanel != null && a == 0){ //if user says yes and there is a selected panel
        String id = currentPanel.getId(); //get the id

        //delete flow from database based on the id
        try{
            Connection conn = ConnectionProvider.getCon();
            String query = "DELETE FROM flow WHERE id = ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, id);

            ps.executeUpdate();
            currentPanel = null; //set the current panel to be null

            //display success message
            String message = "Flow/checkpoint deleted successfully.";
            JOptionPane.showMessageDialog(getContentPane(), message);

        }catch(SQLException se){
            JOptionPane.showMessageDialog(getContentPane(), se);
        }

        //deduct the checkpoint by 1 and query to refresh workflow
        updateCheckpoint(-1);
        queryWorkflow();
        //set the checkpoint text
        checkpoint.setText("Total: " + current_workflow.getCheckpoint() + " checkpoints");

        //refresh the flow view on the right
        queryFlow();
        cloneablePanel.removeAll();
        createClonedPanels(flowList, flowList.size());

        //deactivate delete button and clear all fields
        deactivateDeleteBtn();
        clearAllFields();
    }
}
```

The delete flow method is called when the user clicks the activated delete button in the edit panel. It will first ask the user for confirmation. After that, it will delete the flow from the database and set the current panel as null (no selected panel). After that, it will update the number of checkpoints inside the workflow and refresh the flow view on the right side. It will also refresh the edit panel by clearing all the fields and deactivating the delete button.

```
private void saveBtnActionPerformed(java.awt.event.ActionEvent evt) {
    //if the current panel is null (there is no selected panel)
    if (currentPanel == null){
        //when save button clicked, it will insert a new flow
        insertNewFlow();
        deactivateDeleteBtn();
    }
    else{
        //edit the selected panel flow attributes
        editFlow();
    }
}
```

When the edit panel is empty, then when the user clicks the save button, it will insert the data as a new flow. Meanwhile, if the edit panel is not empty (there is a selected panel), then update the flow.

```
private void insertNewFlow(){
    //get the name field
    String nameStr = nameField.getText();
    String typeStr;
    String dayToStr;

    //get the type and dayTo
    if(oneDay.isSelected()){
        typeStr = "One-day event";
        dayToStr = "0";
    }
    else{
        typeStr = "Multiple-day event";
        dayToStr = toField.getText();
    }

    //get the dayFrom, note, and color
    String dayFromStr = fromField.getText();
    String noteStr = notesArea.getText();
    if (noteStr.equals("Notes")){
        //if equals to placeholder, then the note is still empty
        noteStr = "";
    }
    String colorStr = (String) colorComboBox.getSelectedItem();

    //handle user validation
    if (nameStr.equals("Name") || nameStr.equals("")){//empty name
        JOptionPane.showMessageDialog(getContentPane(), "Name is still empty.");
    }
    else if (typeStr.trim().isEmpty()){ //empty type
        JOptionPane.showMessageDialog(getContentPane(), "Type is still empty.");
    }
    else if (dayFromStr.equals("Day") || dayFromStr.equals("")){ //empty day from
        JOptionPane.showMessageDialog(getContentPane(), "Day 'from' is still empty.");
    }
    else if (!isInteger(dayFromStr)){ //day from not integer
        JOptionPane.showMessageDialog(getContentPane(), "Day 'from' is not valid.");
    }
    else if (colorStr.trim().isEmpty()){ //emty color
        JOptionPane.showMessageDialog(getContentPane(), "Color is still empty.");
    }
```

```
//only for multiple day event
else if ((dayToStr.equals("Day") || dayToStr.equals("")) && typeStr.equals("Multiple-day event")){
    JOptionPane.showMessageDialog(getContentPane(), "Day 'to' is still empty."); //empty day to
}
else if (!isInteger(dayToStr) && typeStr.equals("Multiple-day event")){ //day to not integer
    JOptionPane.showMessageDialog(getContentPane(), "Day 'to' is not valid.");
}
else{
    //set day from day to becomes integer
    int dayFromInt = Integer.parseInt(beforeAfterDay(dayFromStr, fromComboBox));
    int dayToInt = Integer.parseInt(beforeAfterDay(dayToStr, toComboBox));

    if ((dayFromInt > dayToInt) && typeStr.equals("Multiple-day event")){
        //if it's multiple day and the date range is not valid
        JOptionPane.showMessageDialog(getContentPane(), "Day range is not valid.");
    }
    else{
        getFlowLastID(); //get last ID
        Flow new_flow = new Flow(flowIDTemp, this.workflowID, nameStr, typeStr, dayFromInt, dayToInt, noteStr, colorStr);

        try{
            Connection con = ConnectionProvider.getCon();

            //insert the new flow to the flow table in the database
            PreparedStatement ps = con.prepareStatement("insert into flow values(?,?,?,?,?,?,?,?)");
            ps.setString(1, new_flow.getId());
            ps.setString(2, new_flow.getWorkflowID());
            ps.setString(3, new_flow.getNameInput());
            ps.setString(4, new_flow.getTypeInput());
            ps.setInt(5, new_flow.getDayFromInput());
            ps.setInt(6, new_flow.getDayToInput());
            ps.setString(7, new_flow.getNoteInput());
            ps.setString(8, new_flow.getColorInput());
            ps.executeUpdate();

            //success message
            String message = "Flow added successfully.";
            JOptionPane.showMessageDialog(getContentPane(), message);

            //clear all the field
            clearAllFields();

            //add the checkpoint by 1 and query to refresh workflow
            updateCheckpoint(1);
            queryWorkflow();
            checkpoint.setText("Total: " + current_workflow.getCheckpoint() + " checkpoints");

            //refresh the flow view on the right
            queryFlow();
            cloneablePanel.removeAll();
            createClonedPanels(flowList, flowList.size());

        }catch(Exception e){
            JOptionPane.showMessageDialog(getContentPane(), e);
        }
    }
```

The insertNewFlow method is used to insert the new flow into the database. It will first get all the inputted data from fields, radio buttons, and combo boxes and set it to become Strings. After that, it will check or validate all the data and display the message if the data is not correct or not valid. After all requirements are met, it will get the last ID of Flow and insert the new flow into the database. After that, it will clear all fields in the edit panel, update the checkpoint by adding it with 1, and refresh the flow view on the right side.

```
else{
    //if all conditions met
    flowIDTemp = currentPanel.getId(); //get the current panel ID

    try{
        Connection con = ConnectionProvider.getCon();

        //update the flow attributes in the database
        String query = "UPDATE flow SET name = ?, type = ?, dayFrom = ?, dayTo = ?, notes = ?, color = ? WHERE id = ?";
        PreparedStatement ps = con.prepareStatement(query);
        ps.setString(1, nameStr);
        ps.setString(2, typeStr);
        ps.setInt(3, dayFromInt);
        ps.setInt(4, dayToInt);
        ps.setString(5, noteStr);
        ps.setString(6, colorStr);
        ps.setString(7, flowIDTemp);
        ps.executeUpdate();

        //show success message
        String message = "Flow edited successfully.";
        JOptionPane.showMessageDialog(getContentPane(), message);

        //clear all the field
        clearAllFields();

        //refresh the flow view on the right
        queryFlow();
        cloneablePanel.removeAll();
        createClonedPanels(flowList, flowList.size());

    }catch(Exception e){
        JOptionPane.showMessageDialog(getContentPane(), e);
    }
```

The above code is a part of the editFlow method. The edit flow method is very similar to the previous insertNewFlow method. It will first get all the inputted data and validate it. The differences are in the code that is presented above. After all requirements are met, it will instead get the current panel ID and update the database with the new data based on the ID. After that, it will clear all the fields and refresh the flow view on the right side. No need for updating checkpoints because the number is still the same.

```
private void clearAllFields(){
    //clear all the fields in the edit panel
    //set all the combo box to the first index
    currentPanel = null;
    nameField.setText("");
    oneDay.setSelected(true); //by default, the type is one day
    fromField.setText("");
    fromComboBox.setSelectedIndex(0);
    toField.setText("");
    toComboBox.setSelectedIndex(0);
    toField.setEnabled(false);
    toComboBox.setEnabled(false);
    notesArea.setText("");
    colorComboBox.setSelectedIndex(0);
}
```

The clearAllFields method is used to clear all the fields in the edit panel after adding, deleting, or editing a Flow. It will set all the fields to be empty again, combo boxes to select their first index, and the type to be a one-day event. The to fields are also disabled.

h. TextResult

The text result frame is shown when the user clicks the generate workflow to text button on the top right of the edit workflow frame. The text result frame is also not independent. When we click the 'X' button in it, it will only close itself and the user will then back to see the Edit Workflow frame.

```
private void createLabels(){
    //create the result labels inside the resultPane
    int yPos = 0;
    int space0;
    int space_iter =1;
    int labelHeight = 30;

    String currentDate = dateList.getFirst(); //get the first date of the result
    for (int i = 0; i < flowlist.size(); i++) {
        //if it is the first index or the date is not the same
        if (i == 0 || (!dateList.get(i).equals(currentDate))){
            currentDate = dateList.get(i);
            space_iter = 1;

            //if the the date is new, add the adder space0
            if (i == 0) space0 = 30;
            else space0 = 0;

            yPos += space0;
            //set the date label
            JLabel date_label = new JLabel(dateList.get(i));
            date_label.setFont(new java.awt.Font("Montserrat Semibold", 0, 18));
            date_label.setBounds(10, yPos, 460, labelHeight);
            resultPane.add(date_label);
            yPos += labelHeight + 10;

            //set the flow label
            JLabel label = new JLabel(flowlist.get(i).getNameInput());
            label.setFont(new java.awt.Font("Montserrat", 0, 18));
            label.setBounds(10, yPos, 460, labelHeight);
            resultPane.add(label);
            yPos += labelHeight + 10;
        }
        else{
            //if the date is not new or the flow has the same date as previous flow
            space_iter +=1;
            //only set the flow label
            JLabel label = new JLabel(flowlist.get(i).getNameInput());
            label.setFont(new java.awt.Font("Montserrat", 0, 18));
            label.setBounds(10, yPos, 460, labelHeight);
            resultPane.add(label);
            yPos += labelHeight + 10;
        }
        //set the size of the result pane based on the space iter or number of elements
        Dimension newSize = new Dimension(resultPane.getWidth(), 400+space_iter *50);
        resultPane.setPreferredSize(newSize);

        //revalidate and repaint result pane and scroll pane
        resultPane.revalidate(); resultPane.repaint();
        scrollPane.revalidate(); scrollPane.repaint();

        // Scroll to show the new panel
        scrollPane.getVerticalScrollBar().setValue(0);
    }
```

The createLabels method is used to display the result (labels of the date and flow based on the inputted date or D-day). It will first get the first date of the result and iterate over the flow list to generate the flows and their dates. If it is a new date, then there is the space0 adder, so that there is more space when switching dates. The date label will also be initialized with the flow label. If it is not a new date, then, it will only initialize the flow label (flow name). After all are done, the method will calculate the result pane width based on the space iterations or number of elements. Then, it will revalidate the scroll pane and the result pane to display the labels.

```
private void setBtnActionPerformed(){
    //ensure that all the old labels and date are empty
    resultPane.removeAll();
    dateList.clear();

    //get the data of the day, month, and year
    String dateStr = dateField.getText();
    String monthStr = (String) monthComboBox.getSelectedItem();
    String yearStr = yearField.getText();

    //create the String in the format of YYYY-MM-dd
    String modify = yearStr + "-" + monthMap.get(monthStr) + "-" + dateStr ;
    if (modify.length() != 10){
        //add a 0 to the date if the date is only a number
        //for example: 2024-02-9 will become 2024-02-09
        modify = modify.substring(0, 8) + "0" + modify.charAt(8);
    }
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    //check if the String is a valid date or not
    if (isValid(modify, formatter)){
        //if valid, then parse it to LocalDate
        LocalDate settedDate = LocalDate.parse(modify, formatter);

        for (Flow result : flowlist){
            //iterate over every flow in the flow list
            //if the flow is one day event
            if (result.getTypeInput().equals("One-day event")){
                //count the date of the flow and add it to the date list
                LocalDate day = settedDate.plusDays(result.getDayFromInput());
                dateList.add(convertDatetoStr(day));
            }
            else{
                //if multiple day event
                //count the date range of the flow and add it to the date list
                LocalDate dayFrom = settedDate.plusDays(result.getDayFromInput());
                LocalDate dayTo = settedDate.plusDays(result.getDayToInput());
                String dateFix = convertDatetoStr(dayFrom) + " - " +convertDatetoStr(dayTo);
                dateList.add(dateFix);
            }
        }
        createLabels(); //create the labels based on the flow and the setted d-day
    }else{
        JOptionPane.showMessageDialog(getContentPane(), "Date is not valid.");
    }
```

This method specifies what the program will do when the set button is performed or pressed. It will first remove all the dates in the date list and labels in the result pane. After that, it will change the inputted date to String and check if it is a valid date or not. If it is valid, then it will count the date for all flows inside the flow list. After that, it will create the labels based on the flow list and date list.

```
private void copyBtnActionPerformed(){
    //when the user clicks the copy button
    String result = "";
    //use hash set, so that only unique date inside it
    HashSet<String> dateSet = new HashSet<>();
    dateSet.addAll(dateList); //get all the date

    //for every component inside the result pane
    for (Component comp : resultPane.getComponents()){
        //get the JLabel Component
        JLabel lab = (JLabel) comp;
        if (dateSet.contains(lab.getText())){
            //if it is a date, than plus one enter before appending the text (the date)
            result += "\n";
        }
        //append the text and end with an enter
        result += lab.getText() + "\n";
    }
    //copy the result
    StringSelection stringSelection = new StringSelection (result);
    Clipboard clpbrd = Toolkit.getDefaultToolkit ().getSystemClipboard ();
    clpbrd.setContents (stringSelection, null);
    JOptionPane.showMessageDialog(getContentPane(), "Successfully copied the result!");
}
```

This method specifies what the program does when the copy button is clicked. It will first create a HashSet based on the date list so that it can only contain unique dates. After that, it will get all the labels inside the result pane and append each text inside the label to the result. If the label is the date label, then it will append additional space before appending the date label. After the iterations are done, the result will be copied to the clipboard, and a success message will be shown.

i. CalendarPage

The CalendarPage is one of the fourth main menus, so it has a side navigation bar to navigate to other main menus. Therefore, it will also have the hoverButton method like the homepage and add workflow menu. The Calendar Page has two buttons that can let the user open new window on top of it, which are the contact button with the phone logo to open the ContactOthers frame and the insert workflow task button to open the InsertWorkflowTask frame.

The CalendarPage has a large calendar inside it which is the calendarCustom class. It has the calendarPanel and the taskPanel. There are cloned panels inside the task

panel, which uses the CloneablePanelTask. Therefore, the myinit and createClonedPanels method is almost the same.

```
public void queryCurrentTaskList(){
    currTasksList.clear(); //clear all remaining task
    queryTask(); //refresh the task list
    CalendarCell currCell = calendarCustom2.currentPanel.getCurrentCell(); //get the selected cell

    //get the selected cell's date
    LocalDate date_cell = currCell.getDate().toInstant().atZone(ZoneId.systemDefault()).toLocalDate();

    for (int i = 0; i < taskList.size(); i++){
        //get the date from of the task
        LocalDate date_from = convertStrDate(taskList.get(i).getTimeFromInput());
        //one-day event
        if (taskList.get(i).getTypeInput().equals("One-day event")){
            //if the current cell's date is the same with the date from of the task
            if (date_from.equals(date_cell)){
                currTasksList.add(taskList.get(i)); //add the task to the current task list
            }
        //multiple day event
        }else{
            //get the date to of the task
            LocalDate date_to = convertStrDate(taskList.get(i).getTimeToInput());
            LocalDate curr_iter = date_from;

            while (!curr_iter.isAfter(date_to)) {
                //iterate to check if the cell date is in range of the dateFrom to dateTo of the task
                if (curr_iter.equals(date_cell)){
                    //if yes, append the task to the current task list
                    currTasksList.add(taskList.get(i));
                    break;
                }
                // Increment date by one day
                curr_iter = curr_iter.plusDays(1);
            }
        }
    }
```

The queryCurrentTaskList method is used to query the task that happens in the selected cell's date. It will first delete all the remaining old current tasks and refresh all the tasks in the task list. After that, it will get the selected cell's date and check whether the task date is the same as the cell's date (if it is a one-day event) or the cell's date is in range of the task date (if it is a multiple-day event). If yes, then add it to the current task list.

The createClonedPanels logic is basically the same as the previous createClonedPanels method, only the size of the panel is different.

```
queryCurrentTaskList(); //query the current task list
//create the cloned panels in the task panel based on the current date task
createClonedPanels(currTasksList, currTasksList.size());
```

Meanwhile, in the myinit method, instead of creating the cloned panels based on the task list, it will create the cloned panels based on the current task list. The reason is that when the user selects the corresponding cell, it will only display the task for that date.

```
public void refresh(){
    //refresh the calendar page
    queryCurrentTaskList(); //get the selected date task
    cloneablePanel.removeAll(); //remove all the cloned panel
    //create cloned panels based on current task
    createClonedPanels(home.currTasksList, home.currTasksList.size());
    renewTaskText(); //renew the task text and the add task button
}
```

The refresh method is used to refresh the calendar page, specifically the task panel. It will first query the current task list to get the selected cell's date tasks and create cloned panels based on it. After that, it will renew the task text and the add task button.

j. AddNewTask

The add task button in the task panel of the Calendar Page can show a new window on top of the Calendar Page, which is the AddNewTask frame. Because the Add New Task frame is a child window, then when it is closed, it will only close itself, not the entire program. The Task is also similar to the Flow.

```
private void saveBtnActionPerformed(){
    //if save button is clicked
    //get all the data
    String nameStr = nameField.getText();
    String typeStr;
    String dateToStr, monthToStr, yearToStr;

    if(oneDay.isSelected()){
        typeStr = "One-day event";
        dateToStr = "0";
        monthToStr = "0";
        yearToStr = "0";
    }
    else{
        typeStr = "Multiple-day event";
        dateToStr = toDateField.getText();
        monthToStr = (String) monthToComboBox.getSelectedItem();
        yearToStr = toYearField.getText();
    }

    String dateFromStr = fromDateField.getText();
    String monthFromStr = (String) monthFromComboBox.getSelectedItem();
    String yearFromStr = fromYearField.getText();

    //if the notes is the same as the placeholder
    String noteStr = noteArea.getText();
    if (noteStr.equals("Notes")){
        noteStr = ""; //set it as empty
    }
    String colorStr = (String) colorComboBox.getSelectedItem();
```

When the user presses the save button, the method will first get all the data inside the field. Note that the field with the 'to' name is only obtained and put into use when the type is a multiple-day event.

```
    //validation
    if (nameStr.equals("Name") || nameStr.equals("")){ //empty name
        JOptionPane.showMessageDialog(getContentPane(), "Name is still empty.");
    }
    else if (dateFromStr.equals("Date") || dateFromStr.equals("")){ //empty dateFrom
        JOptionPane.showMessageDialog(getContentPane(), "Date 'from' is still empty.");
    }
    else if (!isInteger(dateFromStr)){ //dateFrom is not integer
        JOptionPane.showMessageDialog(getContentPane(), "Date 'from' is not valid.");
    }
    else if (yearFromStr.equals("Year") || yearFromStr.equals("")){ //yearFrom is empty
        JOptionPane.showMessageDialog(getContentPane(), "Year 'from' is still empty.");
    }
    else if (!isInteger(yearFromStr)){//yearFrom is not integer
        JOptionPane.showMessageDialog(getContentPane(), "Year 'from' is not valid.");
    }
```

```
//only for multiple day event
else if ((dateToStr.equals("Date") || dateToStr.equals("")) && typeStr.equals("Multiple-day event")){
    JOptionPane.showMessageDialog(getContentPane(), "Date 'to' is still empty."); //empty dateTo
}
else if (!isInteger(dateToStr) && typeStr.equals("Multiple-day event")){
    JOptionPane.showMessageDialog(getContentPane(), "Date 'to' is not valid."); //dateTo is not integer
}
else if ((yearToStr.equals("Year") || yearToStr.equals("")) && typeStr.equals("Multiple-day event")){
    JOptionPane.showMessageDialog(getContentPane(), "Year 'to' is still empty."); //empty yearTo
}
else if (!isInteger(yearToStr) && typeStr.equals("Multiple-day event")){
    JOptionPane.showMessageDialog(getContentPane(), "Year 'to' is not valid."); //yearTo is not integer
}
else{
    //set date from and date to become String YYYY-MM-dd
    String timeFrom = yearFromStr + "-"  + monthMap.get(monthFromStr) + "-" + dateFromStr;
    String timeTo = yearToStr + "-"  + monthMap.get(monthToStr) + "-" + dateToStr;
    boolean safe = false; //to track whether date range is valid or not

    if (typeStr.equals("Multiple-day event")){ //multiple day event
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        try {
            // Parse the date strings into Date objects
            Date dateFrom = dateFormat.parse(timeFrom);
            Date dateTo = dateFormat.parse(timeTo);

            // Compare the dates to see whether the date range is valid or not
            if (dateFrom.compareTo(dateTo) > 0) {
                JOptionPane.showMessageDialog(getContentPane(), "Date range is not valid.");
            } else if (dateFrom.compareTo(dateTo) < 0) {
                safe = true; //if valid
            } else {
                //if the date is the same
                JOptionPane.showMessageDialog(getContentPane(), "Date 'from' and 'to' are the same. Consider changing to one-day event.");
            }
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }else{
        //if it's one day event
        safe = true;
    }
```

After that, it will validate the user input from the name, date, and year. If the type is multiple-day event, then further validation is needed. It will first validate the dateTo and yearTo. After that, it will check whether the date range is valid or not. If it is valid, then it is safe. The one-day event is automatically safe after the first name, date, and year validations.

```
if (safe){ //if valid
    getNewID(); //get a new ID for the new task
    Task new_task = new Task(this.lastID, nameStr, typeStr, timeFrom, timeTo, noteStr, colorStr, this.userID, false);

    try{
       Connection con = ConnectionProvider.getCon();
       //insert the new task to the database
       PreparedStatement ps = con.prepareStatement("insert into tasks values(?,?,?,?,?,?,?,?,?)");
       ps.setString(1, new_task.getId());
       ps.setString(2, new_task.getNameInput());
       ps.setString(3, new_task.getTypeInput());
       ps.setString(4, new_task.getTimeFromInput());
       ps.setString(5, new_task.getTimeToInput());
       ps.setString(6, new_task.getNoteInput());
       ps.setString(7, new_task.getColorInput());
       ps.setString(8, new_task.getUserID());
       ps.setBoolean(9, new_task.getCompleted());
       ps.executeUpdate();

       //success message
       String message = "Task added successfully.";
       JOptionPane.showMessageDialog(getContentPane(), message);

       //go back to calendarpage
        CalendarPage.open=0;
        setVisible(false);

        //refresh Calendar page
        home.queryTask(); //query task list
        home.queryCurrentTaskList(); //query current task list
        //refresh the task panel
        home.cloneablePanel.removeAll();
        home.createClonedPanels(home.currTasksList, home.currTasksList.size());
        home.renewTaskText();

        //refresh the task dots in the cells
        home.calendarCustom2.refreshTaskDots();
        home.calendarCustom2.currentPanel.revalidate();
        home.calendarCustom2.currentPanel.repaint();

   }catch(Exception e){
       JOptionPane.showMessageDialog(getContentPane(), e);
```

If it is safe, then it will get the new ID for the new task and insert it into the database. After that, it will close the AddNewTask window and refresh the calendar page. It will refresh the task panel, task list, current task list, and calendar panel (task dots).

k. EditTask

The Edit Task frame is also a child window of the Calendar Page, so when it is closed, it will only close itself. The Edit Task frame consists of two major operations, which are the edit task and the delete task. Two of them are also similar to the one performed for flows. The edit task method is also very similar to the add task method. The difference is back again on the method to update it in the database.

```
private void setUpFields(){
    //set name field as the task name
    nameField.requestFocus();
    nameField.setText(thisTask.getNameInput());

    //set radio button
    if (thisTask.getTypeInput().equals("One-day event")){
        //if it is one day event
        oneDay.setSelected(true);
        //disable the to fields
        toDateField.setEnabled(false);
        monthToComboBox.setEnabled(false);
        toYearField.setEnabled(false);
        repaint();
    }else{
        //if it is multiple day event
        multipleDay.setSelected(true);
        //enable the to fields
        toDateField.setEnabled(true);
        monthToComboBox.setEnabled(true);
        toYearField.setEnabled(true);
        repaint();
        //set dateTo to the to fields
        handleDateMonthYear(thisTask.getTimeToInput(), toDateField, monthToComboBox, toYearField);
    }

    //set date from to the from fields
    handleDateMonthYear(thisTask.getTimeFromInput(), fromDateField, monthFromComboBox, fromYearField);
    noteArea.setText(thisTask.getNoteInput()); //set notes
    colorComboBox.setSelectedItem(thisTask.getColorInput()); //set color
}
```

The setUpFields method will set the fields inside the Edit Task frame based on the current task. If the task is a one-day event, then the to fields are disabled. Conversely, the to fields are enabled when the task is a multiple-day event. Meanwhile, the handleDateMonthYear method will handle the date field, month combo box, and year field based on the task date.

```
private void deleteBtnActionPerformed(){
    queryCurrentTask(); //get the information of the current task
    //ask for confirmation
    String question = "Do you really want to delete " + thisTask.getNameInput() + "?";
    int a = JOptionPane.showConfirmDialog(getContentPane(), question, "SELECT", JOptionPane.YES_OPTION);

    if (a == 0){
        String id = thisTask.getId(); //get task ID

        //delete from database
        try{
            Connection conn = ConnectionProvider.getCon();
            String query = "DELETE FROM tasks WHERE taskID = ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, id);

            ps.executeUpdate();
            String message = "Task deleted successfully.";
            JOptionPane.showMessageDialog(getContentPane(), message);

        }catch(SQLException se){
            JOptionPane.showMessageDialog(getContentPane(), se);
        }

        // go back to calendar page
        CalendarPage.open=0;
        setVisible(false);
```

```
        //refresh calendar page
        home.queryTask(); //refresh task list
        home.queryCurrentTaskList(); //refresh current task list

        //refresh task panel
        home.cloneablePanel.removeAll();
        home.createClonedPanels(home.currTasksList, home.currTasksList.size());
        home.renewTaskText();
        //refresh task dots in calendar panel
        home.calendarCustom2.refreshTaskDots();
```

```
//remove the task dot
if (thisTask.getTypeInput().equals("One-day event")){
    //if it is a one day event
    CalendarCell current = home.calendarCustom2.currentPanel.getCurrentCell(); //get current cell
    current.setTaskAmount(current.getTaskAmount() -1); //deduct the task amount
    if (current.getTaskAmount() <= 0){
        //false has task means that there is no other dots
       current.setHasTasks(false, ""); //set has task to be false if the task amount <= 0
    }
}
else{
    //get the range of date to and date from
    LocalDate date_to = home.convertStrDate(thisTask.getTimeToInput());
    LocalDate date_from = home.convertStrDate(thisTask.getTimeFromInput());

    //for every cell
    for (CalendarCell cell1 : home.calendarCustom2.currentPanel.getCells()){
        LocalDate date_cell = cell1.getDate().toInstant().atZone(ZoneId.systemDefault()).toLocalDate();
        //if the cell is in the task date range
       if (date_cell.isAfter(date_from.minusDays(1)) && date_cell.isBefore(date_to.plusDays(1))){
            cell1.setTaskAmount(cell1.getTaskAmount() -1); //deduct the task amount
            if (cell1.getTaskAmount() <= 0){
                //false has task means that there is no other dots
                 cell1.setHasTasks(false, ""); //set has task as false if the task amount  <= 0
            }
       }
    }
}
//revalidate and repaint
home.calendarCustom2.currentPanel.revalidate();
home.calendarCustom2.currentPanel.repaint();
```

If the delete button is pressed, then it will first get the information of the corresponding task. After that, it will ask the user for confirmation. If the user says yes, then it will delete the task from the database and go back to Calendar Page. At the same time, it will also refresh Calendar Page and remove the task dots. The method removes the task dots by getting all cells that have the date or date range of the task and deducting the cell task amount by 1. The hasTask attribute will be set to false if the cell does not have any other tasks anymore. If the hasTask is false, then there will be no color dot in the cell.

The edit task method is very similar to the add task method in the add new task frame. The validations and the way for it to get the data from the fields are all the same.

```
if (safe){//if valid
    try{
        Connection con = ConnectionProvider.getCon();
        //update the tasks table
        PreparedStatement ps = con.prepareStatement("UPDATE tasks SET name = ?, type = ?, timeFrom = ?, "
                + "timeTo = ?, notes = ?, color = ? WHERE taskID = ?");
        ps.setString(1, nameStr);
        ps.setString(2, typeStr);
        ps.setString(3, timeFrom);
        ps.setString(4, timeTo);
        ps.setString(5, noteStr);
        ps.setString(6, colorStr);
        ps.setString(7, this.thisID);
        ps.executeUpdate();

        //success message
        String message = "Task edited successfully.";
        JOptionPane.showMessageDialog(getContentPane(), message);

        //go back to calendarpage
        CalendarPage.open=0;
        setVisible(false);

        //refresh calendar page
        home.queryTask(); //query task list
        home.queryCurrentTaskList(); //query current task list

        //refresh the task panel
        home.cloneablePanel.removeAll();
        home.createClonedPanels(home.currTasksList, home.currTasksList.size());
        home.renewTaskText();

        //refresh the task dots
        home.calendarCustom2.refreshTaskDots();
        home.calendarCustom2.currentPanel.revalidate();
        home.calendarCustom2.currentPanel.repaint();

    }catch(Exception e){
        JOptionPane.showMessageDialog(getContentPane(), e);
    }
```

The difference is that here, it will update the database, not insert a new task into the database. After it updates the task in the database based on the task ID, it will refresh the calendar page (task list, current task list, task panel, and task dots).

l. InsertWorkflowTask

The InsertWorkflowTask is also a child window of the CalendarPage. It will close only itself when the close button is performed.

```
private String[] queryAllWorkflow(){
    workflowList.clear(); //clear the remaining workflow
    try{
        //query all workflow information from the database
        Connection con = ConnectionProvider.getCon();
        String query = "SELECT * FROM workflow WHERE userID = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, this.userID);
```

```
        ResultSet rs = pstmt.executeQuery();
        while(rs.next()){
            String wid = rs.getString("workflowID");
            String wtitle = rs.getString("title");
            int wcheckpoint = rs.getInt("checkpoint");
            //add it to the workflow list
            Workflow workflow = new Workflow(wtitle, wcheckpoint, wid, this.userID);
            workflowList.add(workflow);
        }
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(getContentPane(), e);
    }
    if (!workflowList.isEmpty()){
        //if there are available workflows
        String[] name = new String[workflowList.size()];
        for (int i = 0; i < workflowList.size(); i++){
            //set the array of the workflow name
            name[i] = workflowList.get(i).getTitle();
        }
        return name;
    }
    return null;
```

The queryAllWorkflows method is used to set up the workflow combo box inside the frame. It will first query all the workflows from the database and add them to a workflow list. After that, if the workflow list is not empty, it will create a new String array that contains the names of all of the workflows. This String array will be used to set the combo box.

```
String[] workflowOptions = queryAllWorkflow();
String[] optionIfEmpty = {"None"}; //if there is no workflow available
if (workflowOptions == null){
    workflowOptions = optionIfEmpty;
}
```

The above code is a part of the initDesign method for the combo box. If the result is null based on the previous method, then the combo box will only contain the option “None”.

```
private void getCurrentFlows(){
    flowList.clear(); //clear the flowlist

    //get the selected workflow
    Workflow currWorkflow = null;
    for (int i = 0 ; i<workflowList.size(); i++){
        if(workflowBox.getSelectedItem().equals(workflowList.get(i).getTitle())){
            currWorkflow = workflowList.get(i);
            break;
        }
    }
    //if there is no such workflow, display the message
    if (currWorkflow == null){
        JOptionPane.showMessageDialog(getContentPane(), "The selected workflow does not exist!");
```

```
    }else{
        try{
            //query all the flows of the eslected workflow
            Connection con = ConnectionProvider.getCon();
            String query = "SELECT * FROM flow WHERE workflowID = ? ORDER BY dayFrom ASC";
            PreparedStatement pstmt = con.prepareStatement(query);
            pstmt.setString(1, currWorkflow.getId());

            ResultSet rs = pstmt.executeQuery();
            while(rs.next()){
                String fID = rs.getString("id");
                String fname = rs.getString("name");
                String ftype = rs.getString("type");
                int fdayFrom = rs.getInt("dayFrom");
                int fdayTo = rs.getInt("dayTo");
                String fnotes = rs.getString("notes");
                String fcolor = rs.getString("color");
                //add it to the flow list
                Flow flow = new Flow(fID, currWorkflow.getId(), fname, ftype, fdayFrom, fdayTo, fnotes, fcolor);
                flowList.add(flow);
            }
        }
        catch(Exception e){
            JOptionPane.showMessageDialog(getContentPane(), e);
        }
```

The getCurrentFlows method will be used to get all the flows that a workflow has. It will iterate over the workflow list to see what is the current workflow. If the selected workflow is “None” or does not exist in the workflow list, then it will display the error message. If such workflow exists, then the method will query all the flows of the selected workflow from the database. After that, the flow will be inserted into the flow list.

```
private void OKbtnActionPerformed(){
    //get the date from the fields
    String dateStr = dateField.getText();
    String yearStr = yearField.getText();

    if (dateStr.trim().isEmpty()){ //empty date
       JOptionPane.showMessageDialog(getContentPane(), "Date is still empty.");
    }
    else if (yearStr.trim().isEmpty()){ //empty year
       JOptionPane.showMessageDialog(getContentPane(), "Year is still empty.");
    }
    else{
        getCurrentFlows(); //get the flows of the selected workflow

        //if the flow list is empty
        if (flowList.isEmpty()){
            JOptionPane.showMessageDialog(getContentPane(), "The workflow is not valid or does not have any flows yet.");
            return;
        }
        LocalDate datee = conditioningDate(); //get the LocalDate format of the inputted date

        for (Flow flow : flowList){
            //iterates for every flow inside the flow list
            getNewID(); //get new ID for the flow before inserting it as a task
            LocalDate dateFrom;
            LocalDate dateTo;
            String dateToStr = "0-null-0";

            //if one day event
            if (flow.getTypeInput().equals("One-day event")){
                //get the date from based on the amount of day from of the flow
                dateFrom = datee.plusDays(flow.getDayFromInput());
            }
            else{
                //if multiple day event
                //get the dateFrom and dateTo based on the amount of the dayFrom and dayTo and the inputted date
                dateFrom = datee.plusDays(flow.getDayFromInput());
                dateTo = datee.plusDays(flow.getDayToInput());
                dateToStr = dateTo.toString();
            }
```

```
        //create a new task for the flow
        Task new_task = new Task(this.lastID, flow.getNameInput(), flow.getTypeInput(),
                dateFrom.toString(), dateToStr, flow.getNoteInput(), flow.getColorInput(), this.userID, false);

        try{
            Connection con = ConnectionProvider.getCon();
            //insert the new task to the task database
            PreparedStatement ps = con.prepareStatement("insert into tasks values(?,?,?,?,?,?,?,?,?)");
            ps.setString(1, new_task.getId());
            ps.setString(2, new_task.getNameInput());
            ps.setString(3, new_task.getTypeInput());
            ps.setString(4, new_task.getTimeFromInput());
            ps.setString(5, new_task.getTimeToInput());
            ps.setString(6, new_task.getNoteInput());
            ps.setString(7, new_task.getColorInput());
            ps.setString(8, new_task.getUserID());
            ps.setBoolean(9, new_task.getCompleted());
            ps.executeUpdate();

        }catch(Exception e){
            JOptionPane.showMessageDialog(getContentPane(), e);
        }
    }
    //show success message
    JOptionPane.showMessageDialog(getContentPane(), "Workflow tasks added successfully.");
    //go back to calendarpage
    CalendarPage.open=0;
    setVisible(false);

    //refresh calendar page
    home.queryTask(); //query task list
    home.queryCurrentTaskList(); //query current task list

    //refresh the task panel
    home.cloneablePanel.removeAll();
    home.createClonedPanels(home.currTasksList, home.currTasksList.size());
    home.renewTaskText();

    //refresh the task dots
    home.calendarCustom2.refreshTaskDots();
    home.calendarCustom2.currentPanel.revalidate();
    home.calendarCustom2.currentPanel.repaint();
```

When the OK button is pressed, the method will get the full inputted date from the field and validate it. After that, it will get the flow of the selected workflow. If there is no flow yet in the selected workflow, then display the error message and return. If there are flows, then set the date String into LocalDate format. After that, it iterates over the flow list to set each of them as a task with their new ID and insert them into the database. After all is done, the frame will be closed and the calendar page will be refreshed as usual.

m. ContactOthers

The ContactOthers frame will be opened if the user clicks the contact button on top of the calendar panel. This frame can be linked to two other frames, which are the ContactsList frame and the MsgTemplate frame. When the user clicks the edit contact button, they will be redirected to the ContactsList frame, while when the user clicks the edit message button, they will be redirected to the MsgTemplate frame.

```
private void setUpFields(){
    queryContact(); //get all the contacts
    //set up the array for all the contacts name
    String[] nameArray = new String[contactList.size()];
    for (int i = 0 ; i < contactList.size(); i++){
       nameArray[i] = contactList.get(i).getName();
    }
    //set the box that contains the contacts name
    nameBox.setBackground(new java.awt.Color(234, 234, 234));
    nameBox.setFont(new java.awt.Font("Montserrat", 0, 14)); // NOI18N
    nameBox.setForeground(new java.awt.Color(155, 154, 154));
    nameBox.setMaximumRowCount(contactList.size());
    nameBox.setModel(new javax.swing.DefaultComboBoxModel<>(nameArray));
    nameBox.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Get the selected index
            int selectedIndex = nameBox.getSelectedIndex();
            // Update the label with the selected index
            phoneField.setText(contactList.get(selectedIndex).getPhone());
        }
    });
    getContentPane().add(nameBox, new org.netbeans.lib.awtextra.AbsoluteConstraints(93, 55, 257, 32));
```

```
    //automatically fill the phone number field based on the selected contact
    int selectedIndex = nameBox.getSelectedIndex();
    phoneField.setBackground(new java.awt.Color(234, 234, 234));
    phoneField.setFont(new java.awt.Font("Montserrat", 0, 14)); // NOI18N
    phoneField.setForeground(new java.awt.Color(155, 154, 154));
    phoneField.setEnabled(false);
    phoneField.setBorder(new EmptyBorder(new Insets(2, 15, 5, 10)));
    if (contactList.isEmpty()){
        phoneField.setText("None");
    }else{
        phoneField.setText(contactList.get(selectedIndex).getPhone());
    }
    getContentPane().add(phoneField, new org.netbeans.lib.awtextra.AbsoluteConstraints(93, 104, 257, 32));

    queryMsg();
    //set up the message array that contains all the messages
    String[] msgArray = new String[msgList.size()];
    for (int i = 0 ; i < msgList.size(); i++){
       msgArray[i] = msgList.get(i).getMsg();
    }
    //set the message box for all the messages
    messageBox.setBackground(new java.awt.Color(234, 234, 234));
    messageBox.setFont(new java.awt.Font("Montserrat", 0, 14)); // NOI18N
    messageBox.setForeground(new java.awt.Color(155, 154, 154));
    messageBox.setMaximumRowCount(msgList.size());
    messageBox.setModel(new javax.swing.DefaultComboBoxModel<>(msgArray));
    getContentPane().add(messageBox, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 212, 330, 40));
```

The setUpFields method is used to set up the contact and message template combo boxes and the phone number field. It will first get all the contacts inside the contact list and display their names in the combo box. If the user selects a particular name, then the corresponding contact phone number will be displayed in the phone field. After that, it will get all the message templates and display them inside the combo box.

```
private void OKbtnActionPerformed(){
    //if the contact and message list is still empty
    if (contactList.isEmpty() || msgList.isEmpty()){
        //display the error message and return
        JOptionPane.showMessageDialog(getContentPane(), "Invalid contact or message.");
        return;
    }
    //otherwise
    String phoneNumber = "+" + phoneField.getText(); // get the target phone number
    String message = (String) messageBox.getSelectedItem(); // get the chosen message template
```

```
// Encode the message
String encodedMessage = null;
try {
    encodedMessage = java.net.URLEncoder.encode(message, "UTF-8");
} catch (Exception ex) {
    ex.printStackTrace();
}

// Create the WhatsApp URL
String url = "https://web.whatsapp.com/send?phone=" + phoneNumber + "&text=" + encodedMessage;

// Use the Desktop class to open the URL
if (Desktop.isDesktopSupported()) {
    Desktop desktop = Desktop.getDesktop();
    try {
        desktop.browse(new URI(url));
        //the user will be directed to the WhatsApp chat with the message in the message box
        //user now only needs to check then send the message
    } catch (IOException | URISyntaxException ex) {
        ex.printStackTrace();
    }
} else {
    System.err.println("Desktop is not supported. Unable to open the URL.");
}
```

If the contact or message list is still empty (that means that no way a contact or a message can be chosen), then display the error message and return. Otherwise, get the phone number from the phone field and the message template from the combo box. After that, encode the message and create the WhatsApp URL for the phone number and message. Then, the method will use the Desktop class to open the URL and direct the user to the corresponding WhatsApp chat. The message field in the chat is also already filled with the template message, user only needs to check and press send.

n. ContactsList

The ContactsList is a child window of the ContactOthers frame. So when the ContactsList is closed, then the ContactOthers frame will become visible again. The ContactsList also utilizes the cloneable panel to display the info of the contacts, so the function is highly similar to other createClonedPanels methods. However, the same as the cloned panel in the Edit Workflow menu, the panel can be selected, so the cloned panels have the mouse listeners as below.

```
// Create a new cloned panel
CloneablePanelContact clonedPanel = new CloneablePanelContact(home, 20,
        Color.white, 2 ,id, name, phone);
// Set width and height for the cloned panel
int panelWidth = 255;
int panelHeight = 60;

// Calculate the x and y positions
int x = 10;
int y = 10 + i * (panelHeight + 20);

// Set the bounds and background for the cloned panel
clonedPanel.setBounds(x, y, panelWidth, panelHeight);
clonedPanel.setBackground(new Color(246,252,254));
```

One row only has one cloned panel, which is why there is no need to calculate the row and column. The cloneable panel used is also the cloneable panel for contact, which is the CloneablePanelContact.

```
//the cloned panel can be clicked
clonedPanel.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        //if clicked and the cloned panel is the current panel
        //then set the the current panel to be null
        if (currentPanel == clonedPanel){
            currentPanel.setIsClicked(false);
            currentPanel = null;
            clonedPanel.repaint();
            //clear all fields
            nameField.setText("");
            phoneField.setText("");
        }
        else{
            //if the current panel is not null
            if (currentPanel != null ) {
                currentPanel.setIsClicked(false);
                currentPanel.repaint();
            }
            //set the cloned panel to be the new current panel
            clonedPanel.setIsClicked(true);
            currentPanel = clonedPanel;
            clonedPanel.repaint();
            //set current panel info in the fields
            nameField.requestFocus();
            nameField.setText(currentPanel.getNameInput());
            phoneField.setText(currentPanel.getPhoneInput());
        }
    }
```

If the cloned panel is the current selected panel, then when it is clicked again, the current panel will become null again. Otherwise, if the current panel is not null, then switch the current panel to the newly selected cloned panel.

```
private void setBtnActionPerformed(){
    //if there is no selected panel
    if (currentPanel == null){
        handleAdd(); //add a new contact
    }else{
        handleEdit(); //edit or update the contact
    }
}
```

When the set button is pressed, then if the current panel is null, a new contact will be inserted, else the current data of current panel's contact will be updated.

```
private void handleAdd(){
    //get the data in the name and phone field
    String nameStr = nameField.getText();
    String phoneStr = phoneField.getText();

    if (nameStr.equals("New Name") || nameStr.equals("")){ //empty name
        JOptionPane.showMessageDialog(getContentPane(), "Name is still empty.");
    }
    else if (phoneStr.equals("Phone Number") || phoneStr.equals("")){ //empty phone number
        JOptionPane.showMessageDialog(getContentPane(), "Phone number is still empty.");
    }
    else if (!isPhoneInteger(phoneStr)){ //character in the phone number is not integer
        JOptionPane.showMessageDialog(getContentPane(), "Phone number is not valid.");
    }
    else{
        getLastID(); //get the last id for the new contact
        //add the new contact to the contact list
        Contact new_contact = new Contact(IDtemp, nameStr, phoneStr);
        contactList.add(new_contact);

        try{
            Connection con = ConnectionProvider.getCon();
            //insert the new contact to the database
            PreparedStatement ps = con.prepareStatement("INSERT INTO contact VALUES(?,?,?,?)");
            ps.setString(1, IDtemp);
            ps.setString(2, userID);
            ps.setString(3, nameStr);
            ps.setString(4, phoneStr);

            ps.executeUpdate();
            //success message
            String message = "Contact added successfully.";
            JOptionPane.showMessageDialog(getContentPane(), message);

            //clear all the field
            nameField.setText("");
            phoneField.setText("");

            //reload to show the cloned panels of the added contact
            reloadSelf();

        }catch(Exception e){
            JOptionPane.showMessageDialog(getContentPane(), e);
```

The handleAdd method will get the data from the name and phone number fields. After that, it will validate it or check whether the data is correct or not. If the data is correct, then it will get a new ID for the new contact and add it to the contact list. Then, the new contact will be inserted into the database.

```
public void reloadSelf(){
    queryContact(); //get the updated contact list
    cloneablePanel.removeAll(); //remove all cloned panels
    createClonedPanels(contactList, contactList.size()); //recreate the cloned panels
}
```

The reloadSelf method is used to refresh the cloned panels inside the frame based on the updated contact list.

```
IDtemp = currentPanel.getId(); //get ID of current selected panel

try{
    Connection con = ConnectionProvider.getCon();
    //update it in the database
    String query = "UPDATE contact SET name = ?, phone = ? WHERE id = ?";
    PreparedStatement ps = con.prepareStatement(query);
    ps.setString(1, nameStr);
    ps.setString(2, phoneStr);
    ps.setString(3, IDtemp);
    ps.executeUpdate();

    //success message
    String message = "Contact edited successfully.";
    JOptionPane.showMessageDialog(getContentPane(), message);

    //clear all the field
    nameField.setText("");
    phoneField.setText("");

    //refresh the cloned panels
    reloadSelf();
```

The handleEdit method is similar to the handleAdd method. The difference is only when all the conditions are met. When the data is correct, the handleEdit method will get the current panel's ID and then update the name and phone inside the database based on the current panel's ID. It will then clear all the fields and reload itself because there is new information in the cloned panels.

o. MsgTemplate

The MsgTemplate frame is opened when the user clicks the edit message button inside the ContactOthers frame. This frame is almost the same as the ContactsList frame. The only difference is that the MsgTemplate frame takes and display Message objects, not Contact. There is also only one field for the user to fill in the message. It also uses the cloned panels. The cloned panels are from the CloneablePanelMsg class, while the createClonedPanels method is the same as the one in the ContactsList (but it takes list of Message instead).

The set button when performed also behaves the same. When there is no selected panel, the data will be inserted as a new message, while if there exists a selected panel, then the data will become the new data for the selected panel.

```
private void handleAdd(){
    //get the message
    String messageStr = newMsgField.getText();

    if (messageStr.equals("Add new message") || messageStr.equals("")){ //if still empty
        JOptionPane.showMessageDialog(getContentPane(), "Message is still empty.");
    }
    else{
        getLastID(); //get the last ID
        //add the new message to the message list
        Message new_msg = new Message(IDtemp, messageStr);
        msgList.add(new_msg);

        try{
            Connection con = ConnectionProvider.getCon();
            //insert the message to the database
            PreparedStatement ps = con.prepareStatement("INSERT INTO message_template VALUES(?,?,?)");
            ps.setString(1, IDtemp);
            ps.setString(2, userID);
            ps.setString(3, messageStr);
            ps.executeUpdate();

            //success message
            String message = "Message added successfully.";
            JOptionPane.showMessageDialog(getContentPane(), message);

            //clear all the fields
            currentPanel = null;
            newMsgField.setText("");
            reloadSelf(); //reload to show the new panel

        }catch(Exception e){
            e.printStackTrace();
        }
```

The handleAdd method will first get the message from the message field and then checks whether the result is empty or not. If not, then it will get the last ID of the message and add it to the message list and insert it into the database. All fields and selected panel will be cleared and the cloned panels will be refreshed.

```
IDtemp = currentPanel.getId(); //get current panel's ID

try{
    Connection con = ConnectionProvider.getCon();
    //update the message database
    String query = "UPDATE message_template SET message = ? WHERE id = ?";
    PreparedStatement ps = con.prepareStatement(query);
    ps.setString(1, msgStr);
    ps.setString(2, IDtemp);
    ps.executeUpdate();

    //success message
    String message = "Message edited successfully.";
    JOptionPane.showMessageDialog(getContentPane(), message);

    //clear all the field
    currentPanel = null;
    newMsgField.setText("");
    reloadSelf(); //refresh the panel view

}catch(Exception e){
    e.printStackTrace();
```

The handleEdit method is also the same. After all the validations, it will get the current ID and update the message in the database based on the ID and the new

inputted data. After that, it will clear all fields and selected panel and also refresh the cloned panels.

p. AranaraMenu

Last but not least, we have the last main menu, which is the Aranara menu. Due to it being a main menu, the Aranara menu also has a side navigation bar, where users can access other main menus or log out from there. The hoverButton method is also the same.

```
private void queryCurrentAffection(){
    try{ //get the current affections of all aranaras from the database
        Connection con = ConnectionProvider.getCon();
        String query = "SELECT * FROM user WHERE userID = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, this.userID);

        ResultSet rs = pstmt.executeQuery();
        if(rs.next()){
            int aff_arama = rs.getInt("aff_arama");
            int aff_ararycan = rs.getInt("aff_ararycan");
            int aff_arabalika = rs.getInt("aff_arabalika");

            //index 0 = arama, 1 = ararycan, 2 = arabalika
            affections.add(aff_arama);
            affections.add(aff_ararycan);
            affections.add(aff_arabalika);
        }
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(getContentPane(), e);
    }
}
```

The queryCurrentAffection method is used to query the current affections of all Aranaras and insert them into a Linked List. The first index will be Arama's affection, the second is Ararycan's, and the third is Arabalika's. The affection is used to determine whether an Aranara is unlocked or not.

```
queryCurrentAffection(); //query the affections of the Aranaras

//set panel arama
AranaraDropShadowPanel panel_arama;
panel_arama = new AranaraDropShadowPanel("Arama",affections.get(0), this.userID, home, player);
panel_arama.setBackground(Color.white);
panel_arama.setLayout(null); // Ensure DropShadowPanel uses null layout for its children
```

```
//set panel ararycan
AranaraDropShadowPanel panel_ararycan;
if (affections.get(0) >= 60){ //if the total affection greater than 60
    if (affections.get(1) == 0){ //unlock panel ararycan
        affections.set(1, affections.get(1)+1);
    }
   panel_ararycan = new AranaraDropShadowPanel("Ararycan",affections.get(1),this.userID, home, player);
}else{
    //this panel will still be locked
   panel_ararycan = new AranaraDropShadowPanel("Ararycan",affections.get(1),this.userID, home, player);
}
panel_ararycan.setLayout(null);

//set panel arabalika
AranaraDropShadowPanel panel_arabalika;
if ((affections.get(0) + affections.get(1)) >= 120){ //if total affection greater than 120
    if (affections.get(2) == 0){ //unlock panel arabalika
        affections.set(2, affections.get(2)+1);
    }
    panel_arabalika = new AranaraDropShadowPanel("Arabalika",affections.get(2),this.userID, home, player);
}else{
    //the panel is still locked
   panel_arabalika = new AranaraDropShadowPanel("Arabalika",affections.get(2),this.userID, home, player);
}
panel_arabalika.setLayout(null);
```

The above code is a part of the initDesign method. In there, after getting all of Aranaras current affection, it will initialize the Aranara panel based on the affections. Because Arama is the default Aranara, then it will automatically be unlocked. Meanwhile, Ararycan is unlocked after the current total affection is greater than 60 and greater than 120 for Arabalika. The user can not access the locked Aranaras as explained in the AranaraDropShadowPanel. When an Aranara is unlocked (Ararycan or Arabalika), their default affection is 1.

q. EditAranara

The EditAranara menu is used as a page to interact with the Aranara. It is also an independent page.

```
public void setDialogText(String s){
    //refresh the dialogue box if a button is pressed continuously
    //so that the dialog box is still visible and display other text
    dialog_box.setVisible(false);
    dialog_text.setVisible(false);

    //set the dialog box and text to be visible
    dialog_box.setVisible(true);
    dialog_text.setVisible(true);
    dialog_text.setText(s);

    //set the timer
    Timer timer = new Timer(5000, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            //after 5 seconds, the box and text will be invisible again
            dialog_box.setVisible(false);
            dialog_text.setVisible(false);
        }
    });
    // Start the timer
    timer.setRepeats(false); // Make sure the timer only runs once
    timer.start();
}
```

The setDialogText method is used to set the dialog box and the desired dialog text to be visible for 5 seconds using the timer.

```
public void setGameDialogIcon(String label_path){
    //the game dialog is a special smaller dialog box when playing game with the Aranara
    //set the dialog box icon to be other icon
    dialog_box.setVisible(true);
    dialog_box.setIcon(new javax.swing.ImageIcon(getClass().getResource(label_path)));

    Timer timer = new Timer(2000, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            //after 2 seconds, the box will be invisible and the dialog box icon will be default one again
            dialog_box.setVisible(false);
            dialog_box.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/dialog_box.png")));
        }
    });
    // Start the timer
    timer.setRepeats(false); // Make sure the timer only runs once
    timer.start();
}

private void setHeartIcon(){
    //heart icon is visible when user pats the Aranara
    heart_icon.setVisible(true);
    Timer timer = new Timer(2000, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            //after 2 seconds, the heart will be invisible again
            heart_icon.setVisible(false);
        }
    });
    // Start the timer
    timer.setRepeats(false); // Make sure the timer only runs once
    timer.start();
}
```

Meanwhile, the set game dialog icon is used to set the dialog box to be the special dialog box that is used when playing the game with the Aranara. The special dialog box will be visible only for 2 seconds, and then the dialog box will revert back to the default dialogue box. The setHeartIcon is used to set the heart icon when the Aranara is patted. The heart icon will only be visible for 2 seconds.

```
private void patBtnActionPerformed(){
    //set heart icon regardless the requirements
    setHeartIcon();
    if (affection < 100){ //if affection is not full
        if (checkValidPat()){ //check whether the pat is valid
            //if valid, then affection for the current aranara + 1
            affection += 1;
            String col = "aff_" + aranaraName.toLowerCase();
            try{
                Connection con = ConnectionProvider.getCon();
                //update the affection in the database
                PreparedStatement ps = con.prepareStatement("UPDATE user SET "+ col +" = ? WHERE userID = ?");
                ps.setInt(1, affection);
                ps.setString(2, this.userID);
                ps.executeUpdate();

                //dialog message when successfully patted
                String[] arama_msg = {"Good day Nara!", "Arama is happy today!", "Hehe, thanks Nara!"};
                String[] ararycan_msg = {"Ahh! Ararycan is startled.", "Nara is friend of Aranara.",
                    "Nara is the friend of forest."};
                String[] arabalika_msg = {"Hmph.", "Don't pat Arabalika like that, Nara.",
                    "Hmph. don't disturb Arabalika."};

                //randomize the dialog message
                Random random = new Random();
                int index = random.nextInt(3);
```

```
            //choose the message based on the aranara name
            String[] proper = null;
            switch (aranaraName) {
                case "Arama" -> proper = arama_msg;
                case "Ararycan" -> proper = ararycan_msg;
                case "Arabalika" -> proper = arabalika_msg;
                default -> {
                }
            }
            setDialogText(proper[index]); //set the dialogue text

            //refresh progress bar based on the affection
            affProgressBar.setValue(affection);
            affectiontxt.setText(affection + "/100");

        }catch(Exception e){
            e.printStackTrace();
        }
```

When the pat button is pressed, then regardless of the affections and pat conditions, set the heart icon. After that, the method will check whether the pat is valid or not. If the pat is valid and the affection is lesser than 100, then increase the corresponding Aranara affection and set the random dialogue text based on the Aranara. After that, set the value of the progress bar and the progress label.

```
private boolean checkValidPat(){
    //one user can only pat all the aranaras up to three times a day
    String day = LocalDate.now().getDayOfWeek().name(); //get the day name
    String day_name = day.charAt(0) + day.substring(1,3).toLowerCase();

    if (!day_name.equals(patDay)){
        //if last patted is not today
        //set the pat amount to 1 because it is patted and update the pat database
        patDay = day_name;
        patAmount = 1;
        updatePatDatabase();
        return true; //can be patted
    }
    else{
        if (patAmount < 3){
            //if the last patted is today but have not patted 3 times
            patAmount += 1;
            updatePatDatabase();
            return true;
        }else{
            //if the user already pat 3 times
            setDialogText("Today you have patted the Aranaras enough, Nara.");
            return false;
        }
    }
}
```

The checkValidPat will check the amount of pat that the user has done for today. Every user can only pat the Aranaras up to three times a day. The method will check if the last patted is today or not, if it is not today, pat the Aranara 1 time and update the database. If it is today, it has to be less than 3 if the user wants to pat the Aranara.

```
private void setDefaultBtnActionPerformed(){
    //get the default aranara name in lowercase
    String new_default_aranara = aranaraName.toLowerCase();
    //check whether the default aranara is the same as the one in
    if (defaultAranara.equals(new_default_aranara)){
        setDialogText("I'm already the default one, Nara! :]");
        return; //return if yes
    }
    try{
        Connection con = ConnectionProvider.getCon();
        //update the default aranara in the database
        PreparedStatement ps = con.prepareStatement("UPDATE user SET default_aranara = ? WHERE userID = ?");
        ps.setString(1, new_default_aranara);
        ps.setString(2, this.userID);
        ps.executeUpdate();

        //get the background music corresponding to the new default aranara
        String bgmPath = "src/App/sound/";
        if (new_default_aranara.equals("arama")){
            bgmPath += "MelodyofHiddenSeeds.wav";
        }
        else if (new_default_aranara.equals("ararycan")){
            bgmPath += "IveNeverForgotten.wav";
        }else if (new_default_aranara.equals("arabalika")){
            bgmPath += "ForRiddlesForWonders.wav";
        }

        player.stop(); //stop the initial music
        player.loadMusic(bgmPath); //load the new music
        player.play(); //play the new music

        //success message
        JOptionPane.showMessageDialog(getContentPane(), "Default aranara updated successfully.");

    }catch(Exception e){
        JOptionPane.showMessageDialog(getContentPane(), e);
    }
```

If the set default button is pressed, then check whether the default Aranara is the same as the current Aranara or not. If it is the same, then return. If not, then update the database and set the background music based on the Aranara. Every Aranara has their own background music.

```
private void initBasedOnAranara(){
    try{
        //query from the user table in the database
        Connection con = ConnectionProvider.getCon();
        String query = "SELECT * FROM user WHERE userID = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, this.userID);

        ResultSet rs = pstmt.executeQuery();
        if(rs.next()){
            //get the current aranara name
            String name = aranaraName.toLowerCase();
            //get the affection of the current aranara
            int aff = rs.getInt("aff_"+name);
            affection = aff;
            //set it in the label and progress bar
            affectiontxt.setText(aff + "/100");
            affProgressBar.setValue(aff);

            //set the icon of the aranara based on the aranara name
            aranara.setIcon(new javax.swing.ImageIcon(getClass().getResource("/App/img/"+ name +"_animated.gif")));

            //get pat amount and day
            patDay = rs.getString("pat_day");
            patAmount = rs.getInt("pat_amount");
            //get the default aranara of that user
            defaultAranara = rs.getString("default_aranara");

            //set the dialog box and text to be initially invisible
            dialog_box.setVisible(false);
            dialog_text.setVisible(false);
        }
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(getContentPane(), e);
        e.printStackTrace();
    }
```

The initBasedOnAranara method is used to initialize the Edit Aranara frame based on the current Aranara that is active on the page. It will set the label and progress bar based on the current Aranara's affection in the database and set the Aranara icon based on the current Aranara name. It will also query the pat day, pat amount, and default Aranara from the database.

There is also a chat button so that when the user presses it, the AranaraChatMenu will be open. Only when the AranaraChatMenu is open, the back chat button is set to be visible. This button is used to collapse the opened AranaraChatMenu.

r. AranaraChatMenu

The AranaraChatMenu consists of eight different buttons that has their own operations if it is pressed. Note that eight of them does not increase the Aranara affection.

```
private void hiBtnActionPerformed(){
    //when the user presses the Hi button
    String[] arama_msg = {"Hello Nara! I am Arama.",
        "Nice to meet you, Nara " + username + ".", "Oh welcome, good Nara!"};

    String[] ararycan_msg = {"Hello, Nara. I am Ararycan.",
        "Ahhh! Nara " + username + ".", "You are good Nara. Ararycan is not afraid."};

    String[] arabalika_msg = {"Hmph. I am Arabalika.",
        "Arabalika want to see how strong Nara " + username + " is.", "Hmph. Arabalika wants to practice."};

    //randomly generate the message
    Random random = new Random();
    int index = random.nextInt(3);

    String[] proper = null;
    switch (parent.aranaraName) { //the message is based on the aranara
        case "Arama" -> proper = arama_msg;
        case "Ararycan" -> proper = ararycan_msg;
        case "Arabalika" -> proper = arabalika_msg;
        default -> {
        }
    }
    //set the dialog text
    parent.setDialogText(proper[index]);
    open = 0; //opened window become 0
}
```

The first one is the hi or greet button. When the button is clicked, it will generate a message on the dialogue box randomly based on the Aranara. Every Aranara has three possible responses.

```
private void taskBtnActionPerformed(){
    queryTask(); //query all the tasks information
    String[] options = {"today", "tomorrow", "random date"}; //options in joptionpane
    // Show the initial option dialog
    String choice = (String) JOptionPane.showInputDialog(
            null,   "Choose a date option:",
            "Date Options",  JOptionPane.PLAIN_MESSAGE,
            null,   options,  options[0]
    );

    if (choice == null) {
        // User pressed cancel or closed the dialog
        JOptionPane.showMessageDialog(parent.getContentPane(), "No options selected.");
        open = 0;
        return;
    }

    switch (choice) {
        case "today":
            // get task for today
            String today = new SimpleDateFormat("yyyy-MM-dd").format(new Date());
            getSpecificTaskDate(today);
            break;

        case "tomorrow":
            // get task for tomorrow's date
            Date tomorrowDate = new Date(System.currentTimeMillis() + (1000 * 60 * 60 * 24));
            String tomorrow = new SimpleDateFormat("yyyy-MM-dd").format(tomorrowDate);
            getSpecificTaskDate(tomorrow);
            break;

        case "random date":
            // Prompt the user to enter a random date
            String randomDate = JOptionPane.showInputDialog("Please enter a date (yyyy-MM-dd):");
            if (randomDate != null) {
                // Validate the date format
                try {
                    //get the task for the random date
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
                    sdf.setLenient(false);
                    Date date = sdf.parse(randomDate);
                    getSpecificTaskDate(randomDate);
                } catch (Exception e) {
                    JOptionPane.showMessageDialog(null, "Invalid date format. Please use yyyy-MM-dd.");
                }
            } else {
                // User pressed cancel or closed the dialog
                JOptionPane.showMessageDialog(parent.getContentPane(), "No date entered.");
            }
            break;
```

The second one is the task button. It will first query all the available tasks that the user has. After that, it will show a JOptionPane to let the user choose what date they want to know the tasks of. The choices are today, tomorrow, and random date. If random date is entered, there is another JOptionPane to prompt the user to enter the date. The method will check the format of the date. If today or tomorrow is chosen or the random date is valid, then the method getSpecifisTaskDate will be called with the inputted date.

```
private void getSpecificTaskDate(String date){
    //set the inputted date
    LocalDate input_date = convertStrDate(date);
    LinkedList<String> result = new LinkedList<>();

    //iterate the taskList
    for (int i = 0; i < taskList.size(); i++){
        LocalDate date_from = convertStrDate(taskList.get(i).getTimeFromInput());
        if (taskList.get(i).getTypeInput().equals("One-day event")){
            //if it is one day event and the date is equal to the inputted date
            if (date_from.equals(input_date)){
                result.add(taskList.get(i).getNameInput()); //append task to result
            }
        //if it is multiple day event
        }else{
            LocalDate date_to = convertStrDate(taskList.get(i).getTimeToInput());
            LocalDate curr_iter = date_from;

            //if the inputted date is in range of the task date range
            while (!curr_iter.isAfter(date_to)) {
                if (curr_iter.equals(input_date)){
                    //append to the result
                    result.add(taskList.get(i).getNameInput());
                    break;
                }
                // Increment date by one day
                curr_iter = curr_iter.plusDays(1);
            }
        }
    }
    //show the result in dialogue text
    if (result.isEmpty()){ //if empty
        parent.setDialogText("You don't have any tasks for " + date + ", Nara.");
    }
    else{
        //if not empty
        String message = "For " + date + " task(s), you have ";
        for (int i = 0; i < result.size(); i++){
            message += result.get(i);
            //iterate over all the task inside the result
            if (i != result.size()-1){
                message += ", ";
            }else{
                message += ", Nara.";
            }
        }
        parent.setDialogText(message);
    }
```

The getSpecificTaskDate is used to get the tasks based on the inputted date. It will iterate over the task list and check whether the inputted date is the same or in the range of the task date. If yes, then append the task name to the result. After that, the method will set up a message in the dialog for the user about the task on that date.

```
private void openMsApplication(String app, String name) {
    //is used to open specific ms app on Windows
    try {
        // Command to open Microsoft app on Windows
        String command = "cmd /c start " + app;
        // Execute the command
        Runtime.getRuntime().exec(command);
        //set the dialogue box text
        parent.setDialogText("Opening Microsoft " + name + "... Please wait a moment, Nara!");

    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error opening " + name + e.getMessage());
    }
    open = 0; //the opened window become 0
}
```

The third, fourth, and fifth buttons consecutively are Microsoft Word, Excel, and PowerPoint. The three of them use the same method which is the openMsApplication method. This method will command to open the Microsoft app based on the specific command. For Word, the command (the app variable) is “winword”, for Excel, it’s “excel”, while for PowerPoint, it’s “powerpnt”. After that, it will execute the command and set the dialogue text of the Aranara based on the app name.

```
private void timerBtnActionPerformed(){
    //prompt the user to input the time
    String timeStr = JOptionPane.showInputDialog(null, "Please enter minutes and seconds "
            + "in the format of minute:second.", "Time Input", JOptionPane.PLAIN_MESSAGE);

    if (!timeStr.contains(":")){ //check the formet
        JOptionPane.showMessageDialog(null, "The inputted time format is not correct.");
        open = 0;
        return;
    }

    String[] arrOfStr = timeStr.split(":"); //split the minute and second
    //validation
    if (!isDigit(arrOfStr[0]) || !isDigit(arrOfStr[1])){ //if one of them is not digit
        JOptionPane.showMessageDialog(null, "The inputted time is not valid.");
    }else{
        if (Integer.parseInt(arrOfStr[0]) < 0 || Integer.parseInt(arrOfStr[1]) < 0){
            //if the minute and second is smaller than 0
            JOptionPane.showMessageDialog(null, "The minute and second has to be greater or equal to 0.");
        }
        else if (Integer.parseInt(arrOfStr[1]) >= 60){
            //if the second is greater than or equal to 60
            JOptionPane.showMessageDialog(null, "The second has to be smaller than 0.");
        }
        else{
            //all conditions met
            setReminder(arrOfStr[0], arrOfStr[1]);
        }
    }
    open =0; //opened window become 0 again
}
```

The sixth button is the timer button. When the time button is pressed, it will initialize a JOptionPane to prompt the user to input the minutes and seconds in the format of minute : second. After that, the method will validate the user inputted time. If all conditions are met, it will set the reminder.

```
private void setReminder(String min, String sec){
    //get the reminder time
    int timeSec = Integer.parseInt(min) * 60 + Integer.parseInt(sec);
    parent.setDialogText("Timer starts!"); //set the dialog text

    //set the timer based on the inputted time
    Timer timer = new Timer(timeSec * 1000, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            showCustomDialog("Time's up, Nara!");
        }
    });
    // Start the timer
    timer.setRepeats(false); // Make sure the timer only runs once
    timer.start();
}
```

The setReminder method will get the time based on the inputted minutes and seconds. It will then start a timer based on the time. After the time passes, there will be a custom JDialog that is set on the top to say that the time is up.

```
private void weatherBtnActionPerformed(){
    //prompt the user to enter the city
    String city = JOptionPane.showInputDialog("Enter your city: ");
    getWeather(city);
    open = 0;
}

private void getWeather(String city) {
    try {
        //set the url string for calling the API
        String urlString = String.format(WEATHER_URL, city, API_KEY);
        URL url = new URL(urlString);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");

        //get the response code
        int responseCode = connection.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK) {
            BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuilder response = new StringBuilder();

            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();
            //parse the response
            parseWeatherResponse(response.toString());
        } else {
            //if the city is not available or error
            parent.setDialogText(city + " is not available, Nara!");
            System.out.println("GET request failed. Response code: " + responseCode);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

The seventh button is the weather button. When the weather button is pressed, it will prompt the user to input a city. Then, it will get the weather of the city by calling the OpenWeatherMap API. It will then parse the response. If the city is not available or other errors, then set the error message.

```
private void parseWeatherResponse(String response) {
    //init a json object to store the api call response
    JSONObject jsonObject = new JSONObject(response);
    //get the city name
    String cityName = jsonObject.getString("name");
    JSONObject main = jsonObject.getJSONObject("main");
    //get the temperature
    double temperature = main.getDouble("temp");
    double tempInCelsius = temperature - 273.15; // Convert from Kelvin to Celsius

    //get the weather description
    JSONArray weatherArray = jsonObject.getJSONArray("weather");
    JSONObject weatherObj = weatherArray.getJSONObject(0);
    String weatherMain = weatherObj.getString("main");
    String weatherDesc = weatherObj.getString("description");

    //set the temperature string to 2 decimal points
    String temper = String.format("%.2f", tempInCelsius) + " °C";
    //get the reccomendation of the aranara based on the main weather description
    String suppMsg = getAranaraMsg(weatherMain);
    //set up the message
    String msg = "The weather in " + cityName + " today is " + weatherMain
            + " (" + weatherDesc + ") with the temperature of "
            + temper + ". " + suppMsg;

    parent.setDialogText(msg); //set the dialog text of the message
}
```

The parseWeatherResponse is used to store the response as a JSON object and parse it. This method will get the temperature, weather main, and the weather description. After that, it will customize the message with some Aranara messages about the weather. After that, the completed message will be set to the dialogue box.

```
private void gameBtnActionPerformed(){
    createGamePanel();
    open = 0;
}
```

The last button is the game button. The game is rock, paper, scissors. When the game button is pressed, the method will create a game panel that consists of the rock, paper, and scissors button. User can choose between those three.

```
private void aranaraTurn(){
    //the aranara turn in playing the game
    String[] move = {"rock", "paper", "scissors"};
    //the move will be randomized
    Random random = new Random();
    int index = random.nextInt(3);
    String label_icon_path = "/App/img/dialog_game_" + move[index] + ".png";
    aranara_choice = move[index];
    //set the move in the dialog box
    parent.setGameDialogIcon(label_icon_path);
}
```

After the user moves, the Aranara will randomly choose between rock, paper, or scissors. After that, the Aranara move will be visible on the special dialogue box for 2 seconds.

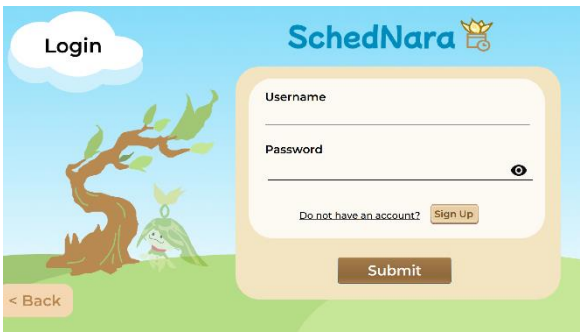
```
private void judgeGame(){
    //judge the game result
    String message = "";
    if (aranara_choice.equals(game_choice)){
        //if its a draw
        message = getGameMsg("draw");
    }else{
        //user rock case
        if (game_choice.equals("rock")){
            if (aranara_choice.equals("paper")){
                //user lose if aranara use paper
                message = getGameMsg("lose");
            }else{
                //user win if aranara use scissors
                message = getGameMsg("win");
            }
        }
        //user paper case
        else if (game_choice.equals("paper")){
            if (aranara_choice.equals("scissors")){
                //user lose if aranara use scissors
                message = getGameMsg("lose");
            }else{
                //user win if aranara use rock
                message = getGameMsg("win");
            }
        }
        //user scissors case
        else if (game_choice.equals("scissors")){
            if (aranara_choice.equals("rock")){
                //user lose if aranara use rock
                message = getGameMsg("lose");
            }else{
                //user win if aranara use paper
                message = getGameMsg("win");
            }
        }
    }
    parent.setDialogText(message); //set the dialog text
}
```

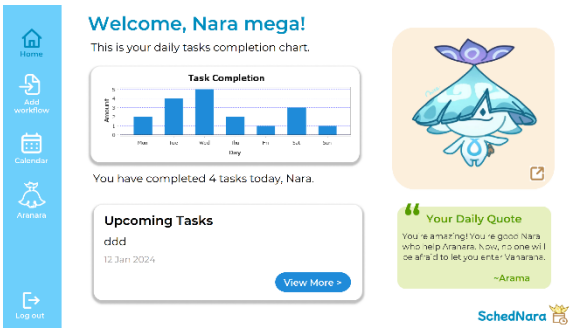
After that, the judgeGame method will be called to determine the result of the game. There are three kinds of results which are the draw, win, and lose. Every result has its corresponding sound effects and every Aranara has its own dialogue about the result. The judgeGame logic is based on the regular rock, paper, and scissors game, where rock is triumphant over scissors, paper over rock, and scissors over the paper.

E. Evidence of Working Program

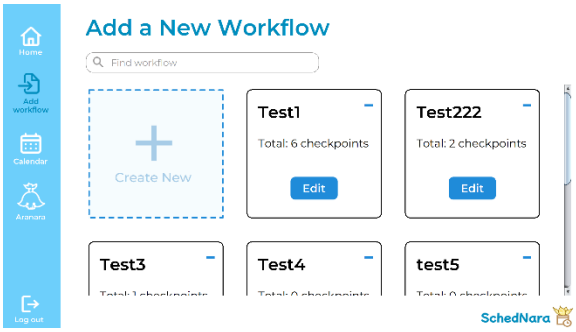
These are only some samples of SchedNara frames with dummy content.

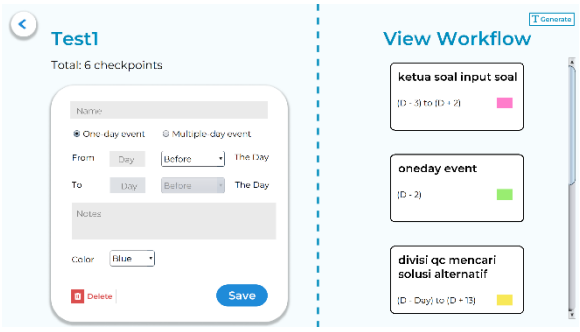
The login page and the homepage



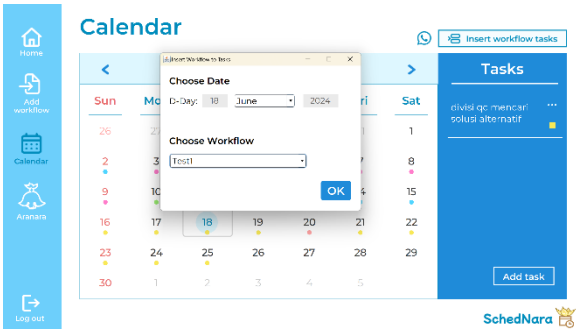


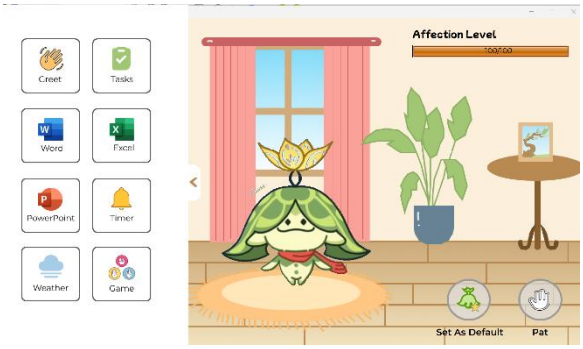
The add workflow menu and the edit workflow page





The calendar page and the edit Aranara menu (Aranara activity page)





F. Lesson Learnt (Reflection)

From this Object Oriented Programming final project, I learn a lot of technical and non-technical skills. First of all, I would like to thank Sir Jude for giving us this project, so that I can learn the Object Oriented Programming in Java. When I first got this project, I immediately thought of the problem and minor inefficiency that I have to do with my organizational job every month, in which I have to count the date of the organizational task based on the monthly workflow. Hence, I created SchedNara. The design is also based on my favorite Aranara so that I can be more motivated in doing this project.

I learned a lot of technical skills from this project, from creating a GUI application to calling the weather API in Java. Because this project idea was originally from me, I did not watch many tutorials, therefore I had to debug a lot along the way. I learned that Java Swing is not as flexible as Python. One example is that it does not allow the resizing of the image icon on the label. Hence, this is a new experience for me.

Meanwhile, the non-technical skill that I have learned from this project is that we should not rush the report less than a week before the submission date. Due to having more time, I focus more on adding features to SchedNara, resulting in less amount of time to do the project report. The thing that surprises me is that I have to spend a day full only for creating the class diagram due to SchedNara having 42 classes. I realized that I should not focus on adding too many features when the deadline is near and should focus on the report. I did not regret creating SchedNara as I can expect myself to still use it in the future. I also realized that SchedNara can be further improved and I will do so in the future if I have the time.

Lastly, I am satisfied with this project more than the previous semester's Algorithm and Programming Final Project due to SchedNara being more original. Besides that, most of the features that I needed and wanted have already been added to SchedNara, so I have no regrets anymore. Although my time management is still bad, I feel that it is better than in the previous semester and I am happy with my skill advancement.

G. Resources

Learning resources:

- Calendar panel display in Java : https://youtu.be/YivaMCfichQ?si=nsv0AYzm3s4dWicD
- OpenWeatherMap API : https://openweathermap.org/forecast
- Debug : https://stackoverflow.com/ and https://chatgpt.com/

Image, font, and audio resources are stated in part C.