2) Application Description and Rationale

To improve climate change education, knowledge, and capability, we have selected SDG Target 13.3. As recent events have demonstrated, climate change is a serious threat to the health of our world, and immediate action is needed to lessen its effects and prepare for the changes that are now occurring. The Sustainable Development Goals (SDGs) underline in Target 13.3 the critical role that institutional capacity, education, and awareness have in addressing climate change challenges. With the following essential characteristics, our application is intended to directly aid in the achievement of SDG Target 13.3:

2.1 Carbon Footprint Calculator:

Description

Using user data, this tool determines the annual carbon footprint output of the user. Not only will it print the users' carbon footprint, this program also introduces a credit system that will award users with more 'Carbon Credit Score' if their annual carbon score is low. This approach shows users their negative impact towards the environment, how to reduce their "Carbon Credit Score" and also encourages participation in reducing climate change. As an incentive to users to reduce their "Carbon Credit Score" this program rewards users with government sponsored titles on their profiles. This system raises awareness towards climate change and its carbon footprint output but also promotes collaborative work between the public and government enabling a nation to be more environmentally responsible.

Rational

Our carbon footprint measures carbon footprints and promotes awareness of environmental impact. Furthermore, it emphasizes the importance of reducing one's carbon footprint by encouraging user's responsibility and engagement. In line with Target 13.3, users will actively contribute to the development of environmental responsibility and awareness.

2.2 Climate Info System:

Description

This system provides information to help a user understand what climate action is, the consequences of climate change and how we can take initiative and help reduce the effects of climate change on our environment. In our climate information system, we have included:

- **Climate News Feed**
  This blog consists of the temperature, energy breakthrough, and plastic waste news that were carefully curated to inform users of our climate's current situation.
- **Interactive Info System**
  Our Interactive Information System provides users suggestions on how to have a positive impact on our environment.
- **Infographics and Videos**
  The Infographics and Videos help users visualize the effects of climate change. It will help them gain insights, awareness, and discover practical ways to contribute to the solution.

Rational

This system provides users with complete information on climate change mitigation and adaptation. It directly contributes to the aim of SDG 13.3 by improving education, awareness, and institutional capacity.

2.3 Early Warning System:

Description

The Early Warning System provides up to date updates on any climate related issue, it includes localized hazards, weather alerts, and climate-related events. With a focus on enhancing early warning efforts, this program not only keeps you updated but also provides valuable insights on preventive measures and immediate actions to take in response to early warnings.

Rational

SDG 13.3 especially addresses the need to increase education, knowledge, and human and institutional capacity on climate change mitigation, adaptation, impact reduction, and early warning. By providing users with current information, preventive measures, and warnings on localized dangers, the Early Warning System directly contributes to the objectives specified in SDG 13.3.

2.4 Interactive Educational Game:

Description

An interactive instructional game that places you in the shoes of an average person on their day off. Navigate through a day full of choices, each of which has an impact on the environment and climate change. Players can experience the effects of their actions in a simulated setting, and choose actions that will benefit the environment. With this, players will gain education insight into the connection between their daily actions and into helping the environment.

Rational

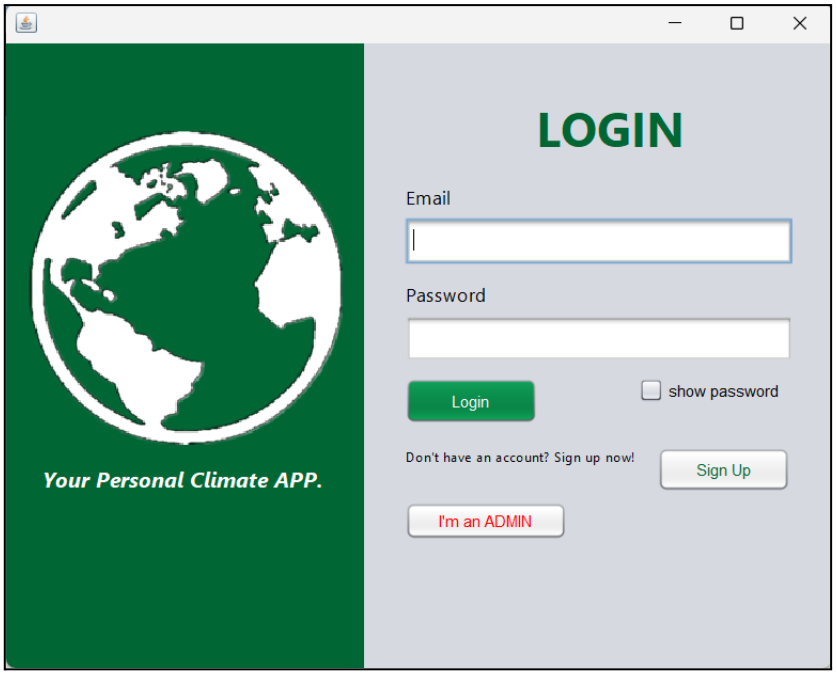
This aligns with SDG target 13.3 through gamification, the game engages and educates users, especially the younger audience, on the complexities of climate change, developing a deeper awareness and personal responsibility.

In conclusion, our key features directly address the objectives outlined in SDG Target 13.3. By improving education, awareness, and institutional capacity on climate change mitigation, adaptation, impact reduction, and early warning, our program aims to empower individuals, communities, and institutions to actively contribute to the global effort in combating climate change. Through innovative tools and engaging educational resources, we aspire to foster a culture of sustainability and climate resilience, paving the way for a more resilient and informed society.

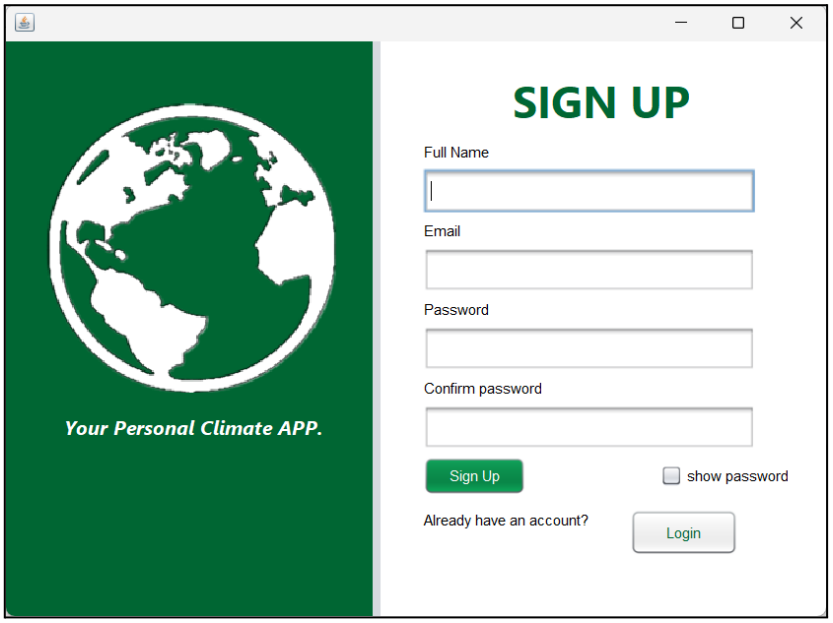
3) User Interface

3.1 User Pages

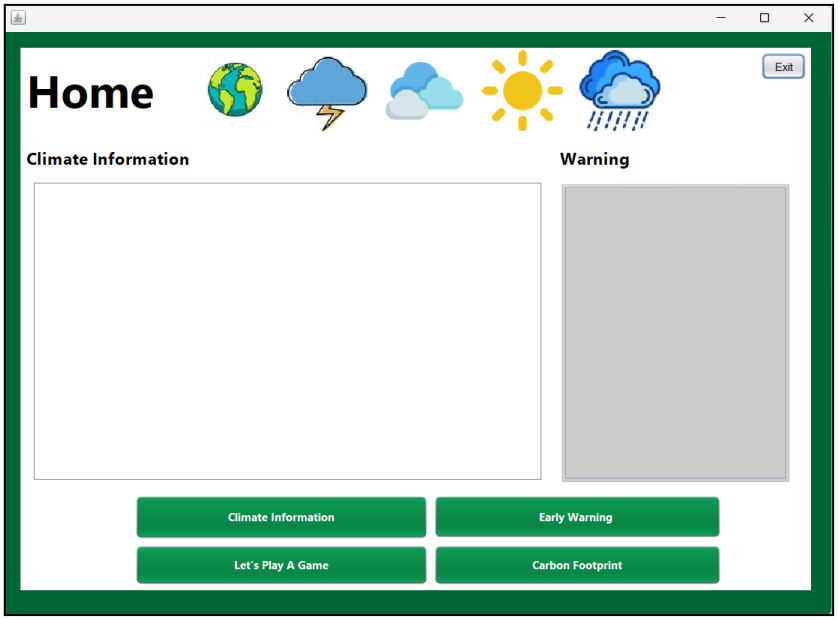
Login Page



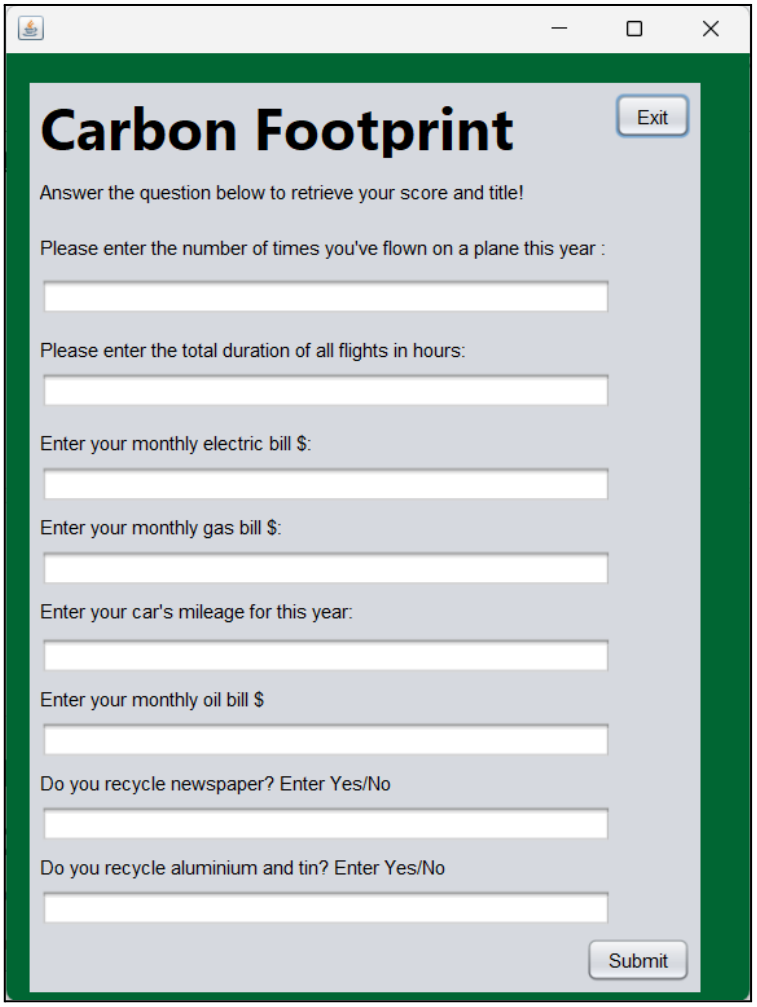
Sign Up Page



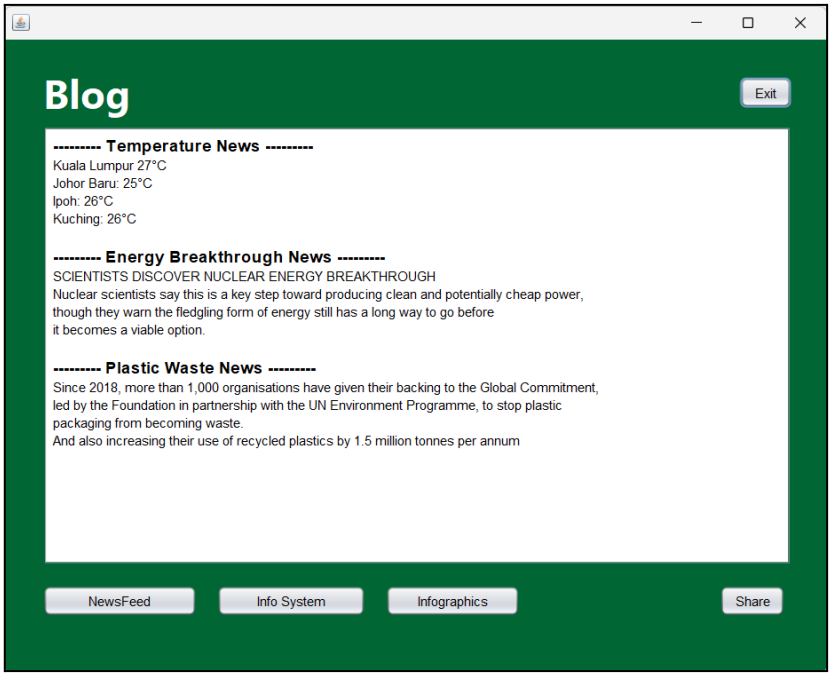
User Homepage



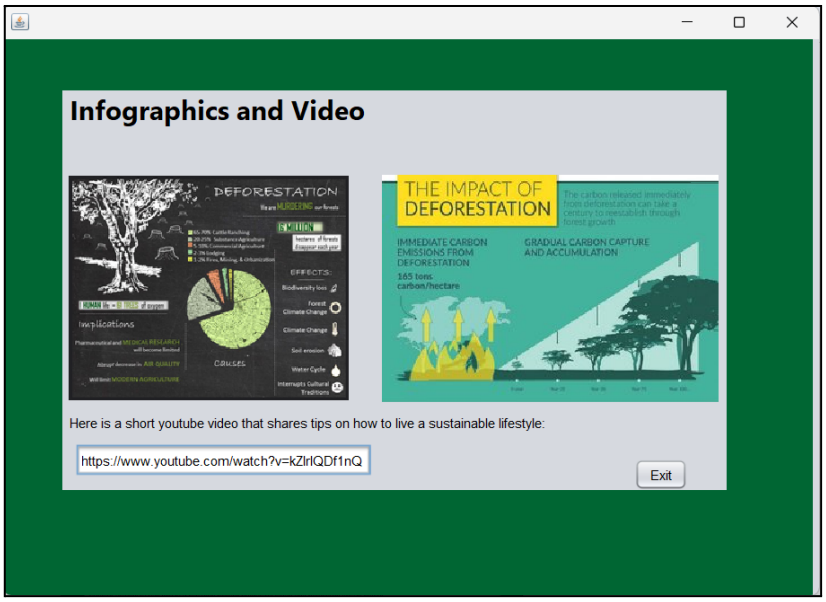
Carbon Footprint Calculator



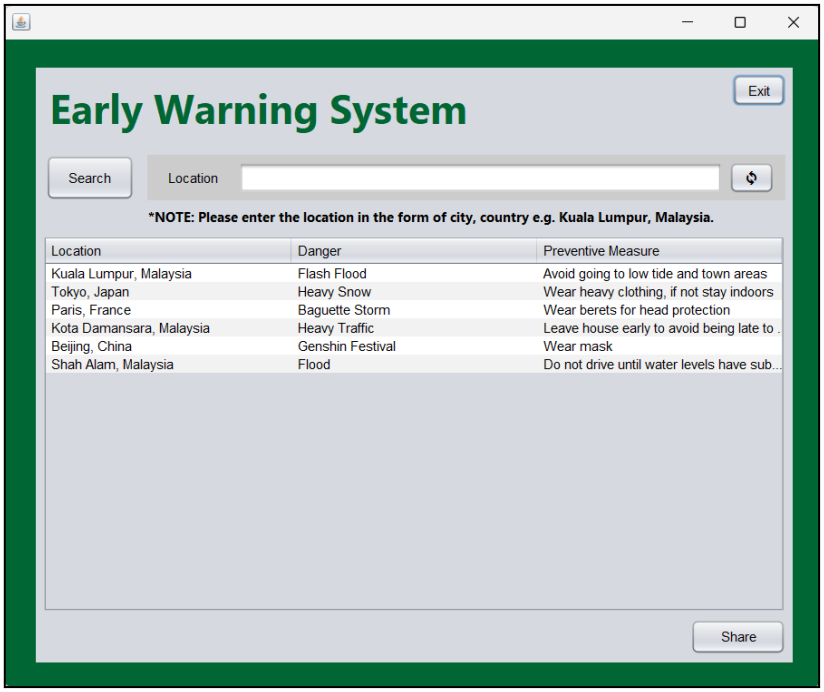
Climate Info System



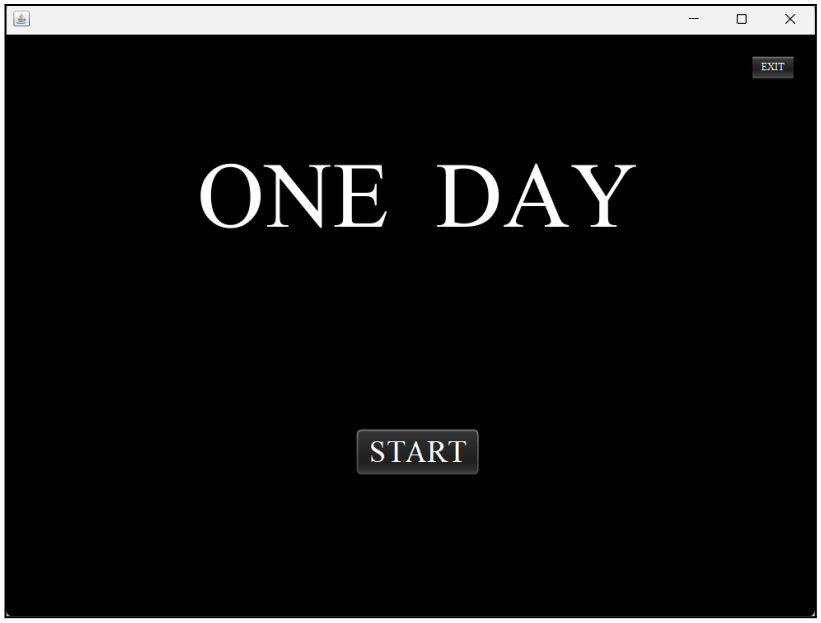
Infographics and Video



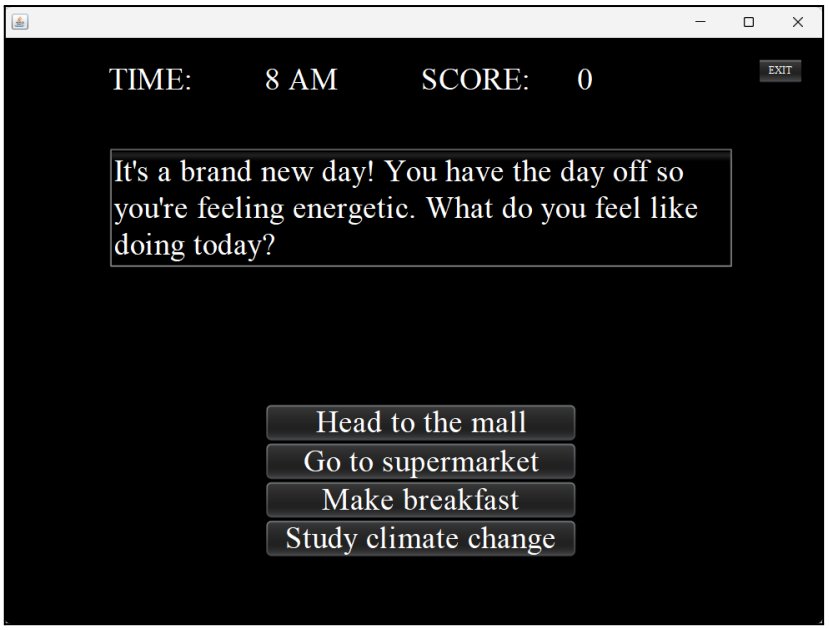
Early Warning System



Game Start Screen



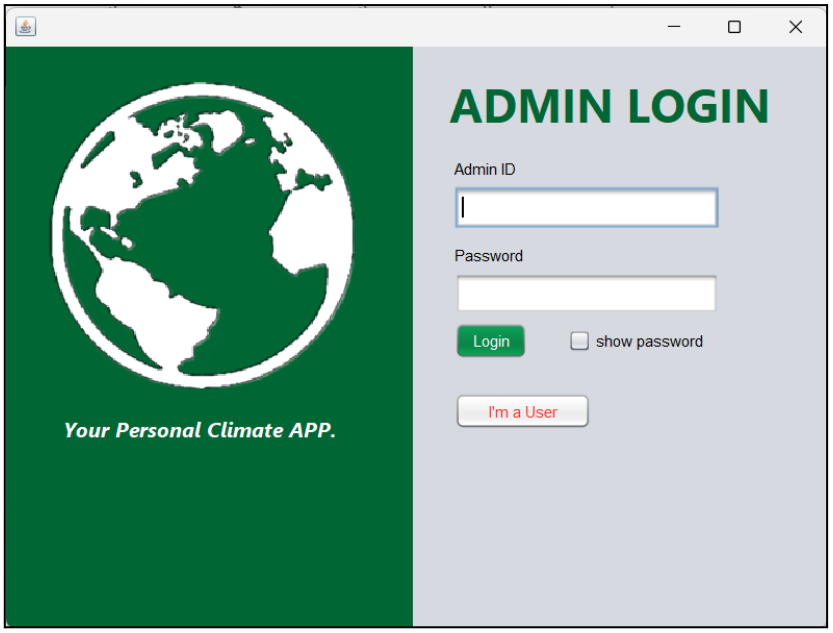
Gameplay Screen



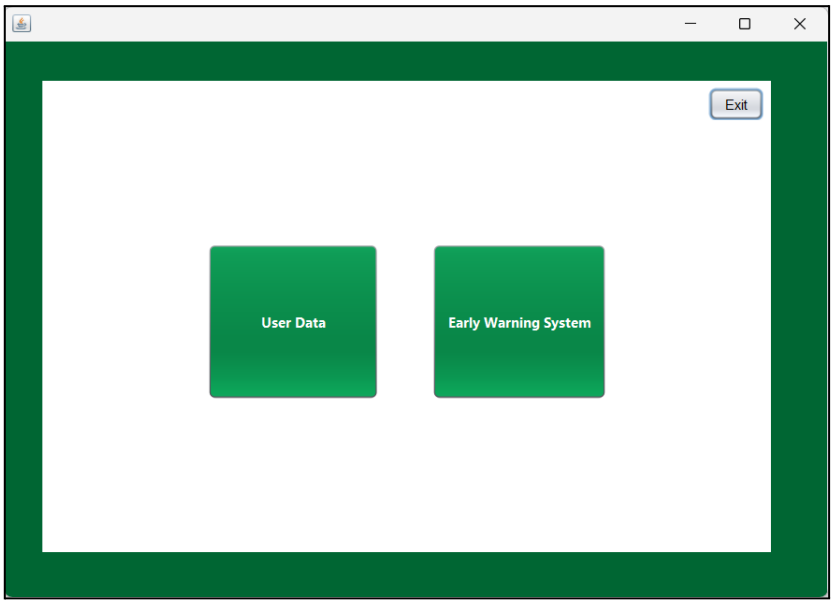
Game Over Screen

3.2 Admin Pages

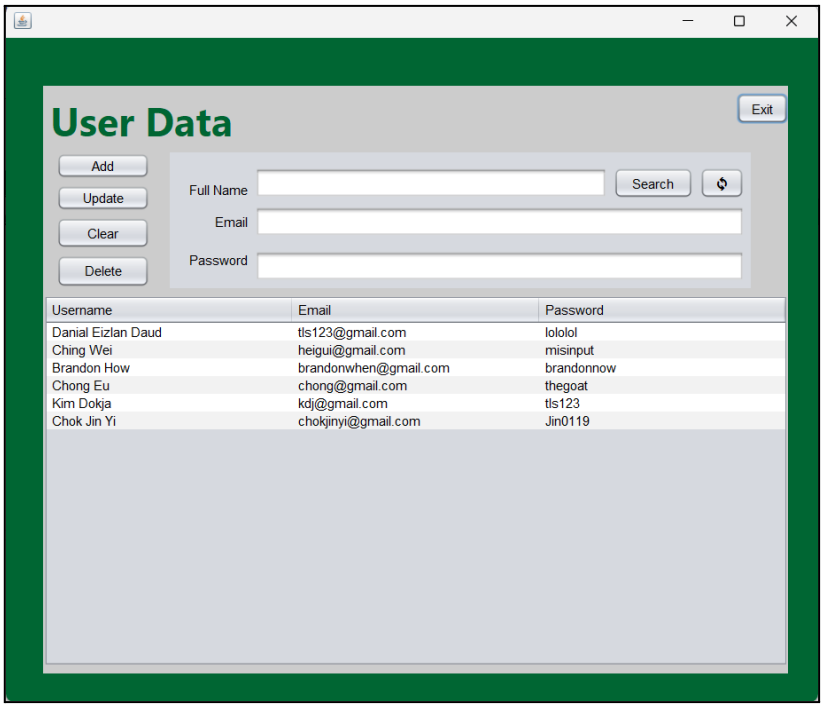
Admin Login Page



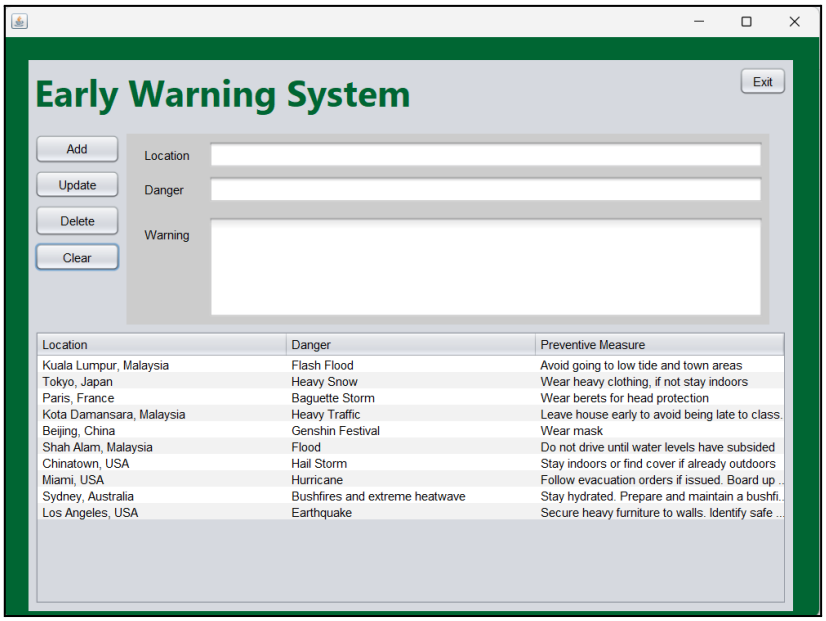
Admin Homepage



User Data (Admin)



Early Warning System (Admin)



4) Lessons Learned

4.1 Lessons learned about Java Programming

While developing this project, we essentially learned the basic separation of concerns by solely having an “Admin” class to represent the admin user and a separate class “AdminControlUserData” to the graphical user interface and user data operations.This allows us to practice on ensuring our project is in terms of good readability and maintainability whereby the graphical user interface is designed with a user-friendly layout, making it clear and intuitive for users to interact with the application.

Furthermore, we collectively as a group had learned to ensure the code uses the appropriate access modifiers to define the scope of variables and methods accordingly. For instance, the consistent use of access modifiers such as “private” and “public” are used consistently, making it easier to understand and navigate the code.Moreover, we implemented the use of JScrollPane effectively in the code to enable scrolling within text areas, which is especially useful when dealing with large amounts of data.

In addition, we primarily learned that nested classes are used to create a hierarchy and a better-organized code. In this case, the nested classes are used to represent a hierarchical structure of warnings. This approach provides a clear and understandable structure for the code.

In conclusion, we had captured a good understanding of the effective use of object-oriented programming principles and adherence to Java coding standards. We had also faced challenges as a group while developing this project involving errors in our codes and difficulty in the right codes. However, this was a lesson to use to convey the journey of discovery while working with Java. This was a lesson to reflect on the evolution of understanding, problem-solving, and the art of crafting software solutions with Java.

4.2 Lessons learned about SDG 13.3

Constructing a Java Program that is solely revolving around the SDG target 13.3, involves us researching and learning about the target goals and its achievements. So what is SDG target 13.3? Target 13.3 has a main goal of improving education, and raising awareness for climate change. The ‘campaign’ educates people on how to help in the mitigation of climate change and how to have a more sustainable lifestyle which would help reduce greenhouse gas emissions as carbon footprint would decrease. It also helps people adapt and learn how to face the effects of climate change; build resilience and respond appropriately to climate-related hazards and natural disasters.

As the effects of climate change are worsening day by day it is pivotal for modern society to know how to respond. As the temperature and more hot days and frequent heat waves are striking all lands, heat-related illnesses are to be considered by most nations and health workers. The effort to raise awareness is a global effort by all nations. They have set a goal that by 2030 all nations should build resilience towards the drawbacks that may come from the effects of climate change. Climate Scientists are also a big contributor to this movement, with some incredible and innovative breakthrough discoveries that would help reduce carbon footprint by a large volume.

Some countries have started implementing SDG 13 in their citizens' lifestyles, recycling bins, public transport, reducing plastic usage, electric cars, and eco-friendly and reusable products are all an effort to encourage a more sustainable lifestyle for everyone. With the support of breakthrough technologies and discoveries, with the help of the Government, we will see drastic progress to the SDG 13.3 Target.

4.3 Lessons learned about Teamwork

As this project is a group assignment we all learned something new, mainly about Java Programming and SDG, but beyond that, we all need to put in effort working as a group. We implemented collaboration skills, communication skills, responsibility, empathy, and awareness. We learned to listen and communicate with each other to accomplish our end goal; the Climate Program. Without our communication skills, we would all be lost and would never accomplish our objective.

With our diverse perspectives and backgrounds, we all expanded our understanding through our peers, which could be from different methodologies and how we structure our algorithm. We also learned the importance of division of labor in a group, this gives each group member a responsibility to fulfill. Encouraging open communication was also a huge factor in the completion of our project, as all group members were encouraged to ask openly about things that they lacked the understanding of, and the empathy of group members to step in and help them.

5) References

5.1 Carbon Footprint Calculator

- Calculating your carbon footprint. (2013, May 23). Just Energy. https://justenergy.com/blog/how-to-calculate-your-carbon-footprint/
- The Nature Conservancy. (2022). What is your carbon footprint? The Nature Conservancy. https://www.nature.org/en-us/get-involved/how-to-help/carbon-footprint-calculator/#:~:text=A%20 carbon%20 footprint%20is%20the

5.2 Climate Info System

- Global efforts show progress on plastic pollution is possible – but world remains off track. (2023, October 31). UN Environment. https://www.unep.org/news-and-stories/press-release/global-efforts-show-progress-plastic-pollution-possible-world
- ACCIONA. (2015). Sustainability in everyday life | Sustainability. In YouTube. https://www.youtube.com/watch?v=kZIrIQDf1nQ
- Magazine, S., & Osborne, M. (2023, August 9). Scientists Repeat Nuclear Fusion Breakthrough in a Step toward More Clean Energy. Smithsonian Magazine. https://www.smithsonianmag.com/smart-news/scientists-repeat-nuclear-fusion-breakthrough-in-a-step-toward-more-clean-energy-180982683/
- AccuWeather. (2023). Kuala Lumpur, Kuala Lumpur, Malaysia Current Weather. Accuweather.com; AccuWeather. https://www.accuweather.com/en/my/kuala-lumpur/233776/current-weather/233776
- AccuWeather. (2023). Johor Bahru, Johor, Malaysia Weather Forecast. Accuweather.com; AccuWeather. https://www.accuweather.com/en/my/johor-bahru/228029/weather-forecast/228029
- AccuWeather. (2023). Ipoh, Perak, Malaysia Weather Forecast. Accuweather.com; AccuWeather. https://www.accuweather.com/en/my/ipoh/234818/weather-forecast/234818
- AccuWeather. (2023). Kuching, Sarawak, Malaysia Weather Forecast. Accuweather.com; AccuWeather. https://www.accuweather.com/en/my/kuching/230204/weather-forecast/230204

5.3 Early Warning System

- Developing Early Warning Systems: A Checklist EWC III Third International Conference on Early Warning From concept to action. (2006). https://www.unisdr.org/files/608_10340.pdf
- CeleneB. Milanés, Batista, Á. R., Núñez, R. A., & Yero, H. T. (2021). Development of a mobile application for Early Warning Systems and risk management in Cuba. IOP Conference Series, 1154(1), 012005–012005. https://doi.org/10.1088/1757-899x/1154/1/012005

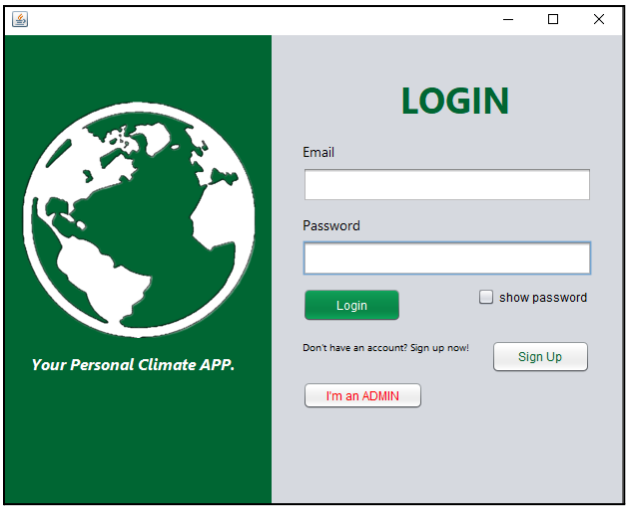
5.4 Interactive Educational Game

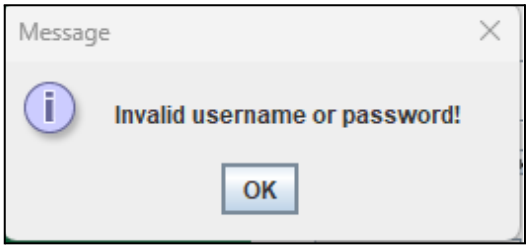
- Carter, B. (2019, June 7). What Is the Most Environmentally Friendly Meat? Eco & Beyond. https://www.ecoandbeyond.co/articles/most-environmentally-friendly-meat/
- Your Habit Of Buying A New Phone Every 2 Years Is Killing The Planet. (2018, November 14). HuffPost. https://www.huffpost.com/archive/ca/entry/smartphone-carbon-emissions-study_ca_5cd576d4e4b07bc729785397
- How do your favourite fashion brands score on climate card? Quite bad. (n.d.). Www.downtoearth.org.in. https://www.downtoearth.org.in/news/climate-change/how-do-your-favourite-fashion-brands-score-on-climate-card-quite-bad-78625
- 7 Reasons Why Supporting Local Business Is Important. (n.d.). 7 Reasons Why Supporting Local Business Is Important. https://www.strikingly.com/blog/posts/7-reasons-why-supporting-local-business-is-important#:~:text=When%20we%20support%20local%20business
- Ranking Milk Based on Sustainability. (2021, October 13). Whole Earth. https://wholeearthsea.com/blog/ranking-milk-based-on-sustainability/
- Eat less hard cheese. (n.d.). Cambridge Carbon Footprint. https://cambridgecarbonfootprint.org/actions/eat_less_hard_cheese/#:~:text=Soft%20cheese%20and%20plant%2Dbased
- Zheng, S., & Krol, A. (2023, February 21). Public Transportation. MIT Climate Portal. https://climate.mit.edu/explainers/public-transportation#:~:text=Public%20transportation%20gets%20people%20where
- If you’re worried about climate change, shop these eco-friendly brands. (2022, October 1). The Manual. https://www.themanual.com/fashion/sustainable-clothing-brands-fighting-climate-change/
- The Top 10 Most Climate Friendly Fast Food Chains, Ranked. (n.d.). Www.foodbeast.com. https://www.foodbeast.com/news/climate-friendly-fast-food-chains/

6) User Manual

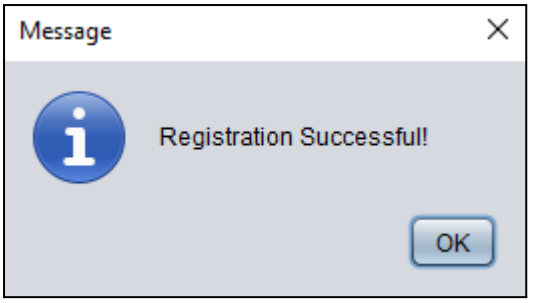
6.1 Login / Sign Up / Admin System

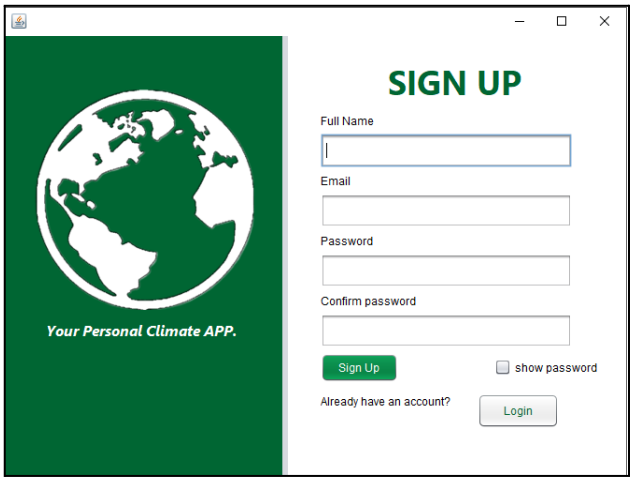
Once you are on the LOGIN page, you are required to enter the registered Email and Password for the account you had created. If you enter the wrong credentials, an error message will pop up.





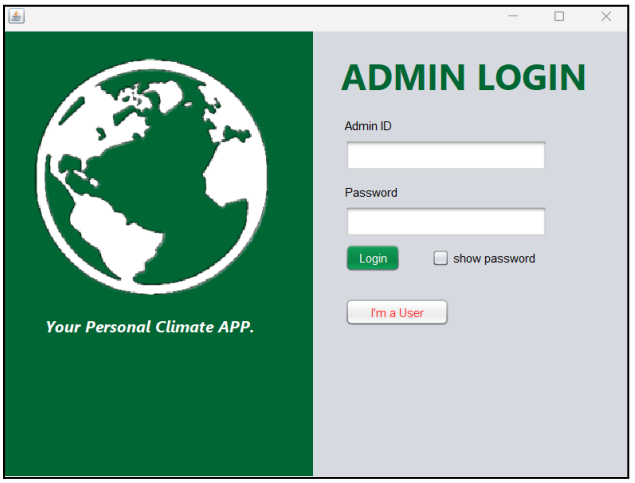
If you have yet to sign up for an account, you may click on the Sign Up button which will then bring you to the Sign Up page. You are required to fill in your Full Name, Email, Password and Confirm Password. Upon registering for a message will pop out stating registration successful.





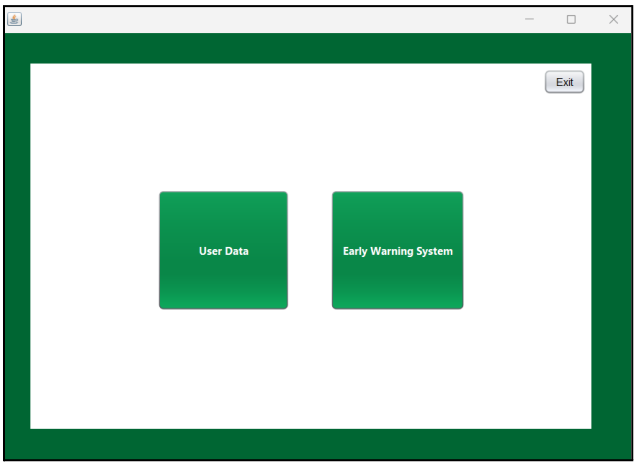
Login Page For Admin Users

If you are an admin user, select ‘I am an Admin’ and the ADMIN LOGIN page will pop up. Enter the fixed Admin ID and Password to enter. If you enter wrong credentials, an error message will pop up.



Home Page for Admin Users

The home page will pop up once you have entered the correct credentials and admin users can choose between User Data and Early Warning System. To exit this page, users can select the exit button on the top right corner.



User Data (Admin)

Upon selecting 'User Data,' admins are directed to the page shown. Admins can perform the following actions:

Add User Data:

- Required fields: 'Full Name,' 'Email,' 'Password.'
- Click 'Add' to add new user data.

Update User Data:

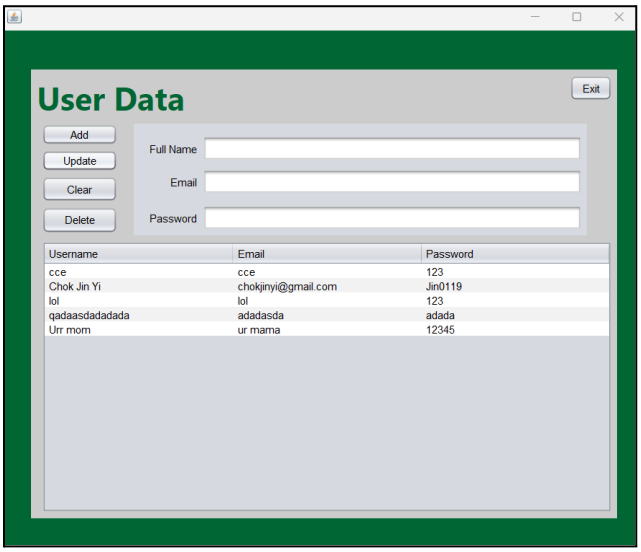
- Select a table row.
- Modify desired fields.
- Click 'Update' to save changes.

Clear User Data:

- Entered data in fields will be cleared

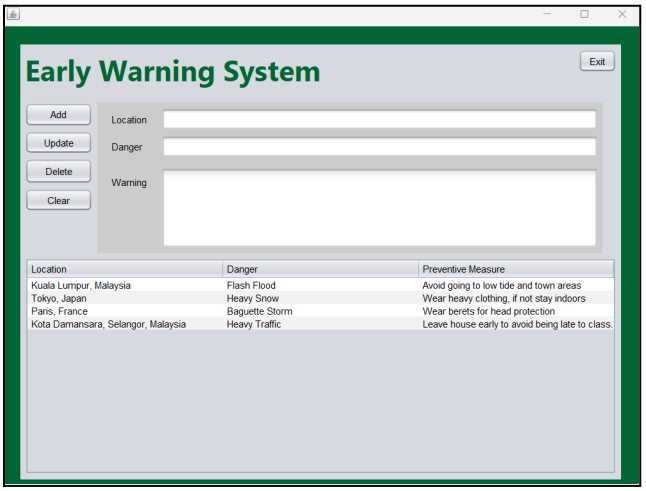
Delete User Data:

- Choose the row for deletion.
- Click 'Delete' to remove the selected user data.



To return to the home page, click 'Exit' in the top right corner.

Early Warning System (Admin)



Upon selecting 'Early Warning System', admins are directed to the page shown. Admins can perform the following options:

Add Warning:

- Required fields: 'Location,' 'Danger,' 'Warning.'
- Click 'Add' to include a new warning.

Edit Warning:

- Select a table row.
- Modify desired fields.
- Click 'Edit' to save changes.

Delete Warning:

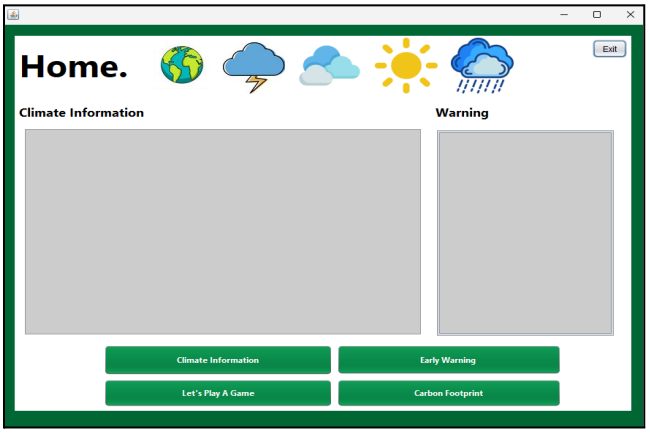
- Choose the row for deletion.
- Click 'Delete' to remove the selected warning.

Clear User Data:

- Entered data in fields will be cleared

To return to the home page, click 'Exit' in the top right corner.

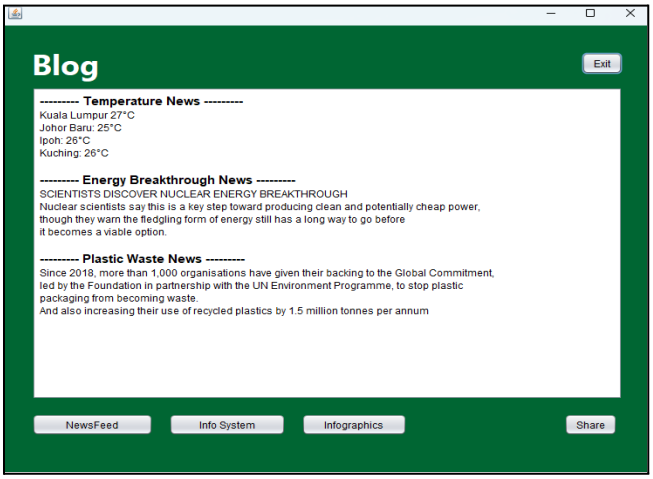
Home Page for Users



Upon logging in, users are welcomed by a Home Page that would provide a Climate Information Panel and Warning Panel that can be used to display information. Users can choose from Climate Information, Early Warning, Let's Play A Game, and Carbon Footprint; which would direct the users to their desired option. There is also an Exit Button if the user wants to log out or leave the program.

6.2 Climate Information System

NewsFeed Page

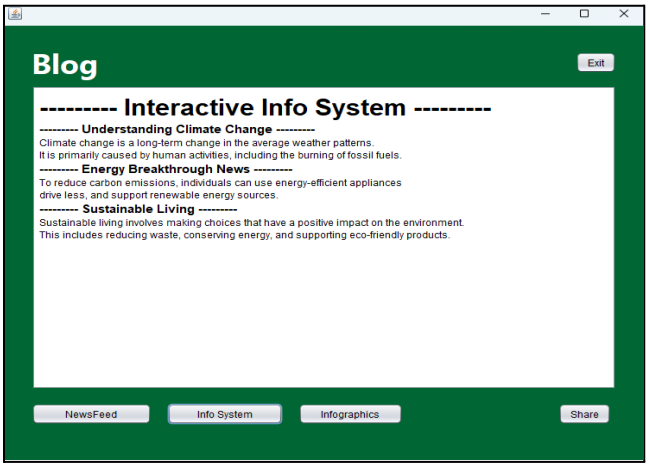


Upon clicking on Climate Information, the user will be greeted with the newsfeed of climate-related news, on the bottom left there are three buttons that the user can freely choose which include the Newsfeed, and the other two are Info System and Infographics. On the bottom right there is a share page where the User can showcase the desired Information on the Climate Information Panel on the Home page.

Located on the top right there is an Exit button which could be used to exit back to the home page.

Info System Page

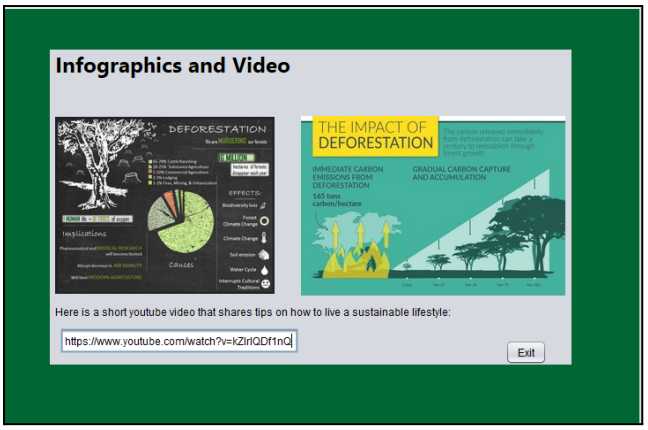
If the User decides to click on Info System, they will face a showcase of the basic necessary information on climate change, and are also able to share this information on the Climate Information Page which is located on the main Home Page.



Infographics & Videos Page

Another possible case is that if the User chooses the Infographics and Video Page, they will be greeted with two Infographics related to climate change and a link that the User can click on which will direct them to a YouTube video that shows ways on how to live a more sustainable lifestyle.

On the bottom right there is an exit button that would direct the User to the NewsFeed Page.



6.3 Early Warning System

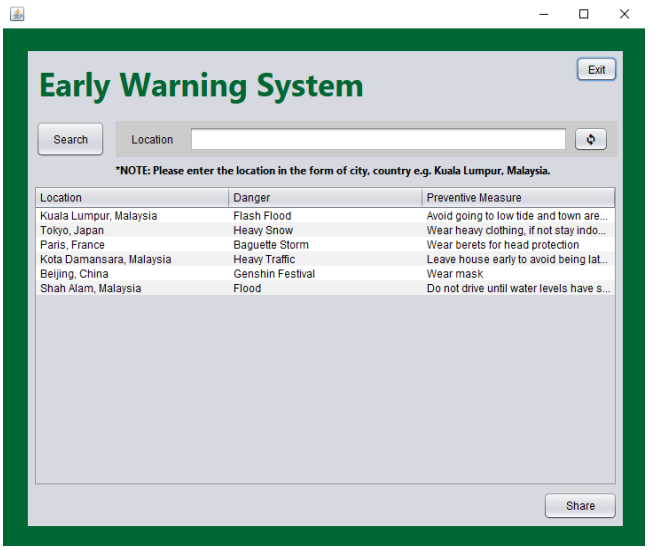
Early Warning System Page

Upon clicking on the Early Warning tab, the user shall be brought to the Early Warning System page, where the user would be able to asses the location,danger and preventive measure accordingly.

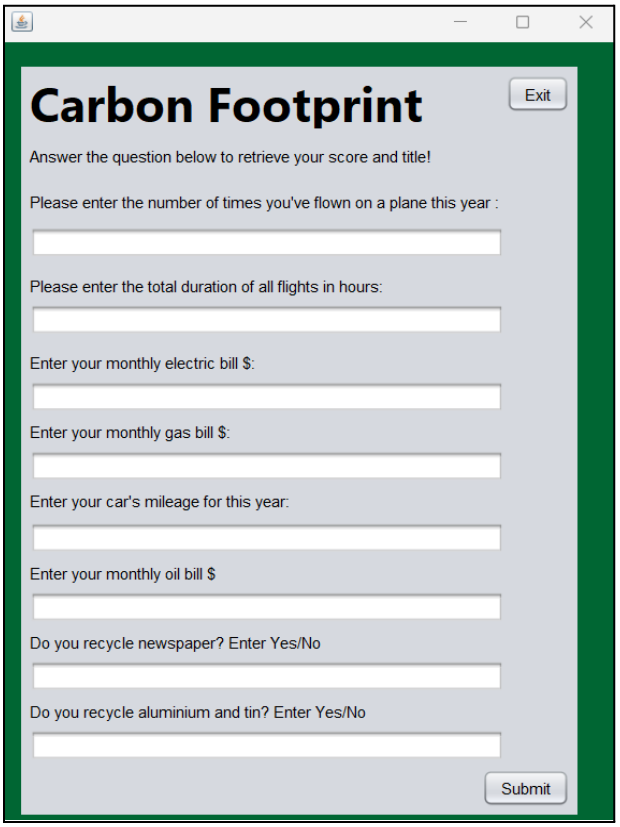
The user would be able to filter out the type of danger and preventive measure by searching the specific location.

The user would also be able to share the warning information by selecting the warning and click on the submit button on the bottom right which will direct the user to the HomePage with the information displayed on the Warning Page.

On the top right there is an exit button that would redirect the user back to the Home page.



6.4 Carbon Footprint Calculator



Calculator Page

To calculate your carbon footprint, follow these steps:

1. Navigate to 'Carbon Footprint Calculator' in the main menu.
2. Complete all required fields accurately.
3. Click 'Submit' at the bottom right corner to calculate.

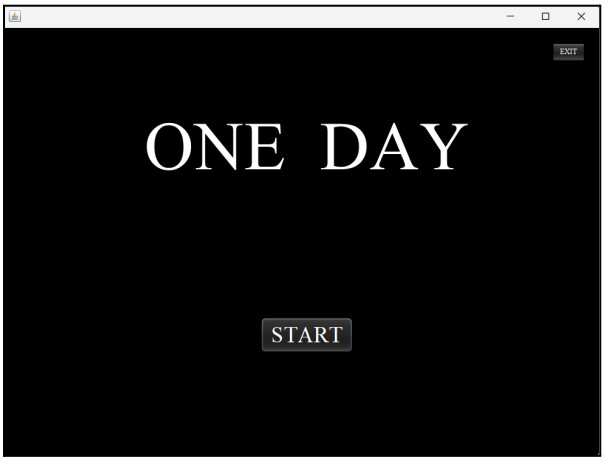
To exit the calculator, click 'Exit' at the top right corner of the screen.

6.5 Interactive Educational Game

Main Menu Screen

Players will be greeted with the main menu screen.

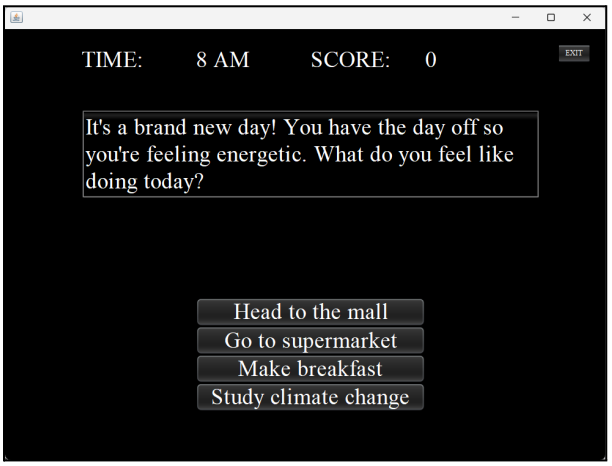
- To start the game, the user may click the START button centered near the bottom of the window.
- To quit the game, an EXIT button will always be present at the top right of the window, throughout the game.



Home Screen (Main)

Player is presented with a header containing the in-game time, the player's current score and a main text box. The text box narrates the context of the player's current position.

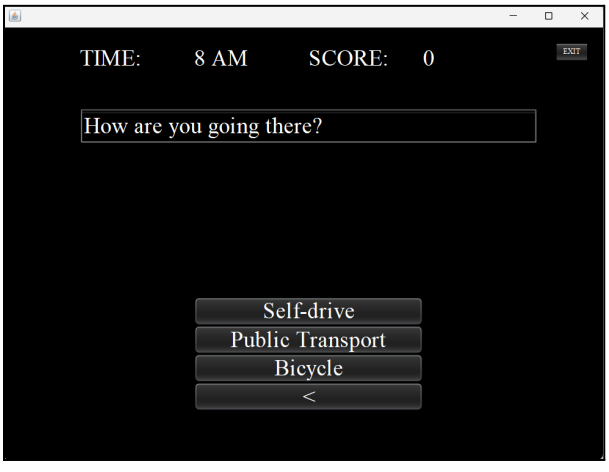
- The player is free to pick from 1 in 4 choices. Once an 'action' is fully completed, the virtual clock will be updated.
- Different choices reward different scores. It is up to the player to figure out how to get a higher score.

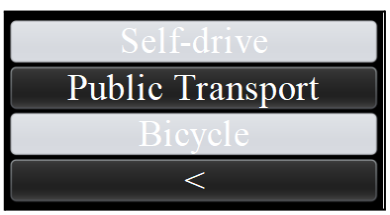


Transport Screen

Should the player choose to go to the Mall or Supermarket, they will first be shown the following screen.

In this case, 3 choices are actual choices while the last one is a 'back' button, allowing the player to go back to the previous screen. Should they pick public transport, that will be the only choice available when they return home.

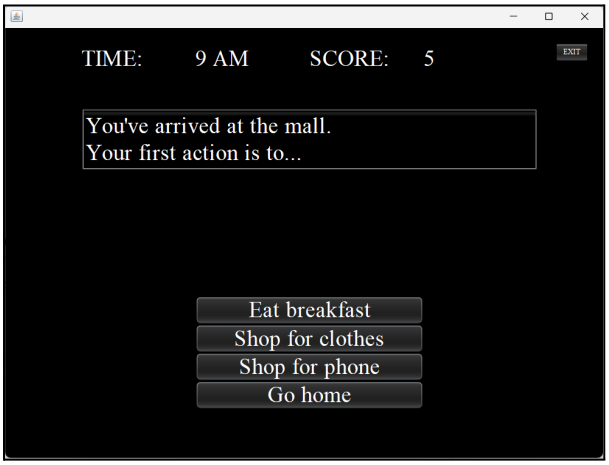




Mall Screen (Main)

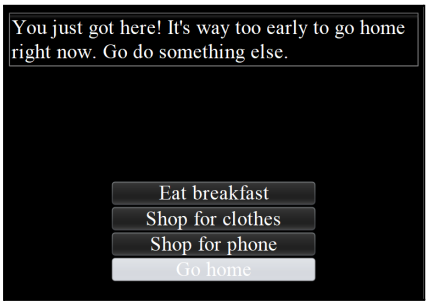
After a transport choice was chosen, they will arrive at the mall successfully. As shown in the image, the score and time has been updated accordingly. Here, they are free to do any of the following:

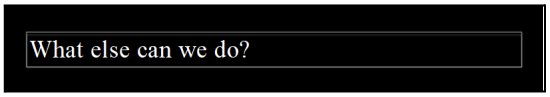
- Eat breakfast (varies with time)
- Shop for clothes
- Shop for phone



Should they try to go home before doing any one of the 3 choices, they will be told by the game and the button gets disabled.

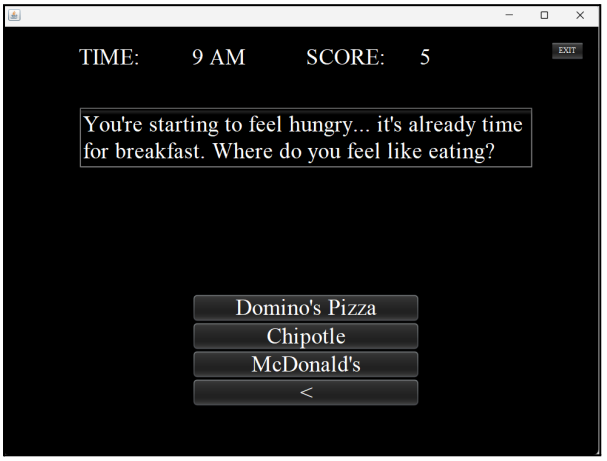
If the player has fully completed one of the choices before, the textbox's content will change accordingly.



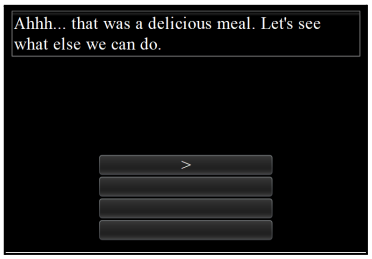


Mall Food Screen

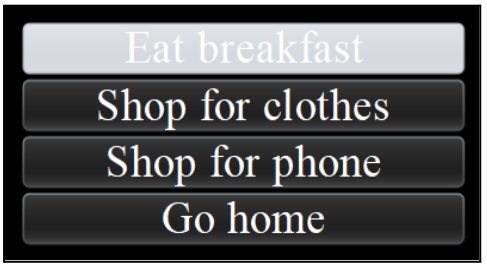
If the eating option is chosen, the user can pick what to eat at the mall between the 3 given fast food chains.



After, a filler screen is shown where the time, score and hunger status is updated.

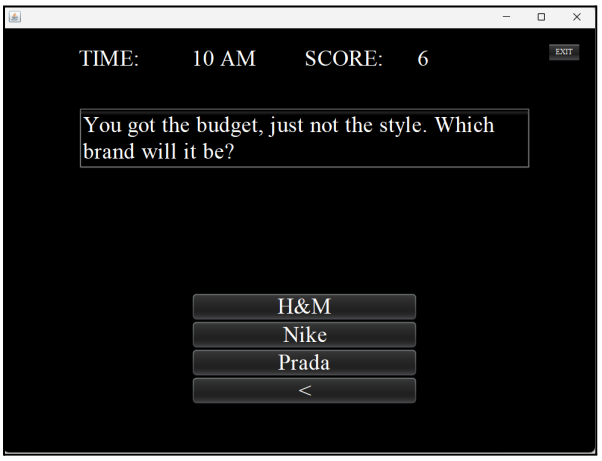


Returning the player to the Mall screen, where the eating option is now disabled until the next mealtime.

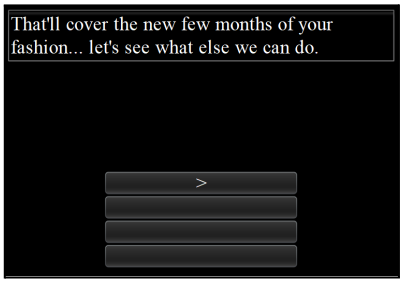


Mall Clothes Shopping Screen

If the clothes shopping option is chosen, the user can pick 1 of 3 clothing brands.



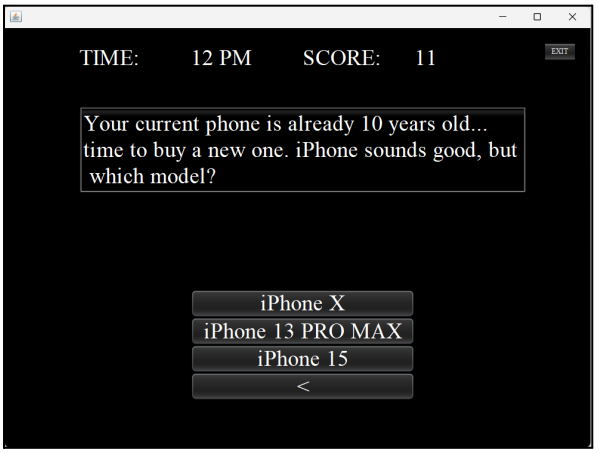
After, a filler screen is shown where the header is updated again.



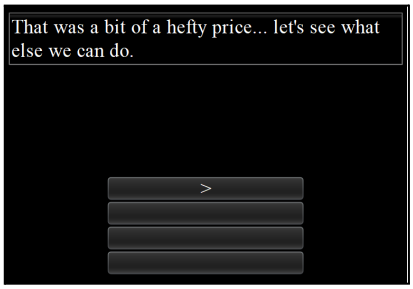
Returning the player to the Mall screen where the clothes shopping option is now disabled.

Mall Phone Shopping Screen

If the phone shopping option is chosen, the user can pick 1 of 3 iPhone models.



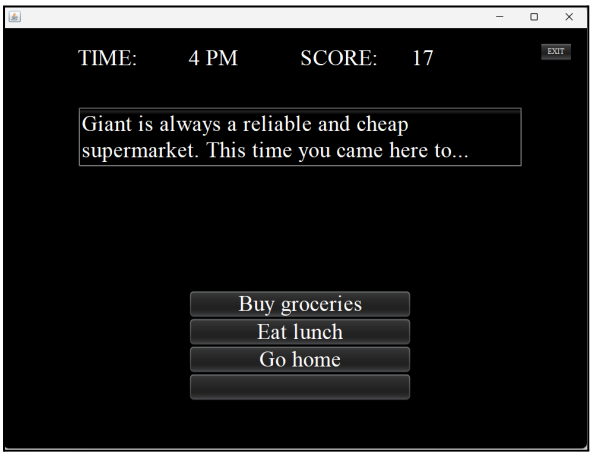
After, a filler screen is shown where the header is updated again.



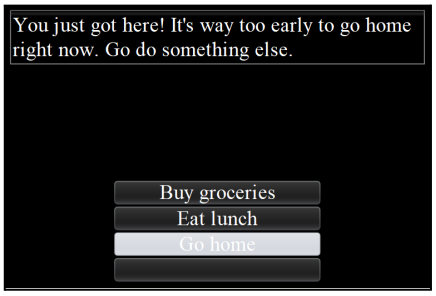
Returning the player to the Mall screen where the phone shopping option is now disabled.

Supermarket Screen (Main)

At the Supermarket, the player has two choices, to buy groceries or to eat. In the screenshot, since it is 4pm the eating choice mentions lunch rather than breakfast or dinner.



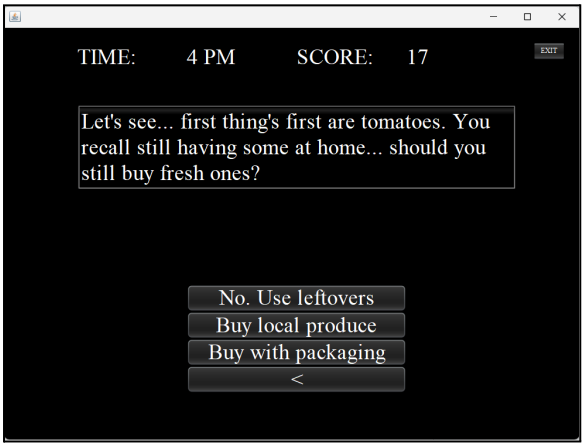
Attempting to go home before making any actions yields the same result as the one at the Mall screen.



Choosing Tomato Screen

The buying groceries option starts with buying tomatoes.

The back button allows the player to cancel buying groceries.



Choosing Cheese Screen

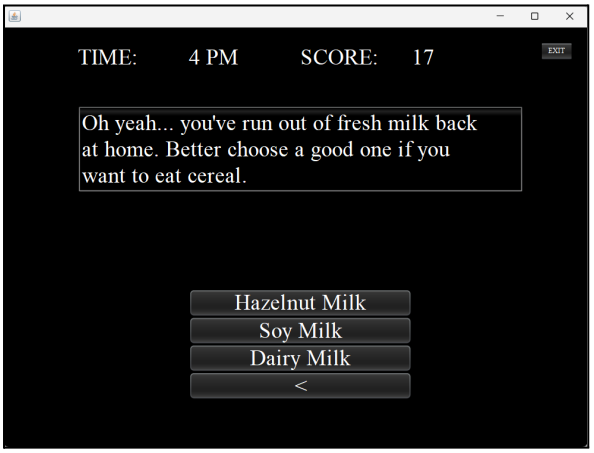
After the tomato choice, now the player has to choose from 3 different types of cheese.

The back button brings the player back to the Choosing Tomato Screen and allows them to redo their choice.

Choosing Milk Screen

After the cheese choice, now the player has to choose from 3 different types of cheese.

The back button brings the player back to the Choosing Cheese Screen and allows them to redo their choice.



Choosing Meat Screen

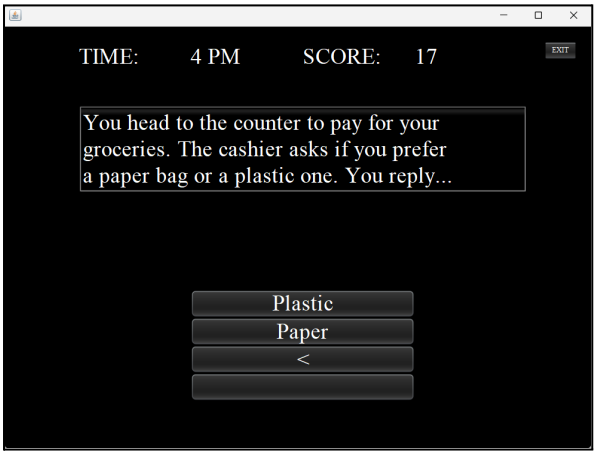
After the cheese choice, now the player has to choose from 3 different types of cheese.

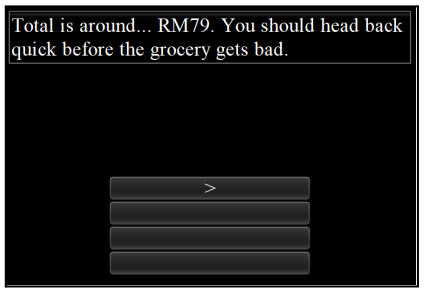
The back button brings the player back to the Choosing Milk Screen and allows them to redo their choice.

Choosing Bag Type Screen

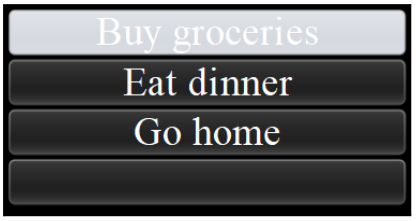
After the meat choice, now the player has to decide whether they want a plastic or paper bag for their groceries.

The back button brings the player back to the Choosing Meat Screen and allows them to redo their choice. This is the final choice of the ‘Buy groceries’ action.





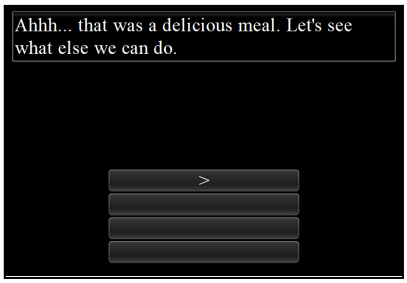
Above is the filler screen for having finished buying groceries, before returning the player to the Supermarket screen where ‘Buy groceries’ is now disabled.



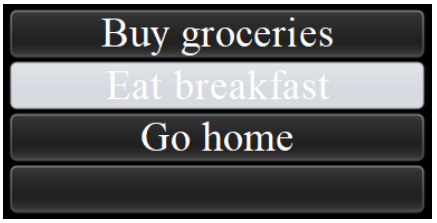
Supermarket Food Screen

If the eating option is chosen, the user can pick between a cafe, restaurant or going to the mall. Going to the mall from this screen will prompt the Transport screen.

Otherwise, a filler screen is shown where the time, score and hunger status is updated.

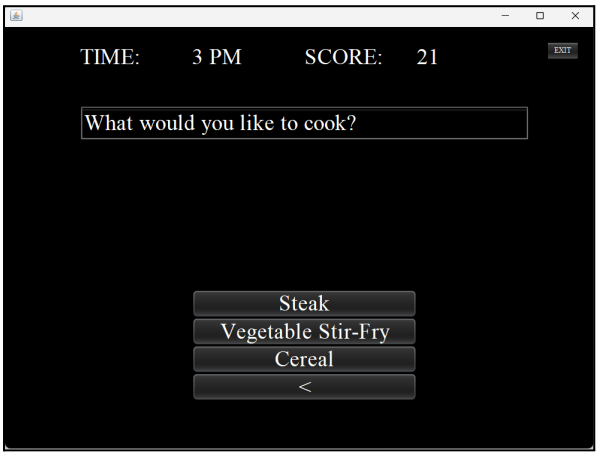


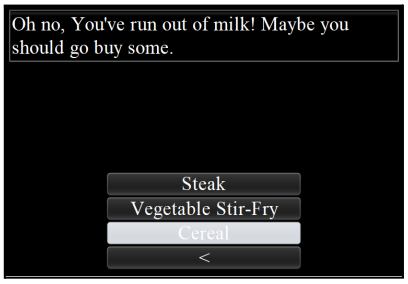
Returning the player to the Supermarket screen, where the eating option is now disabled until the next mealtime.



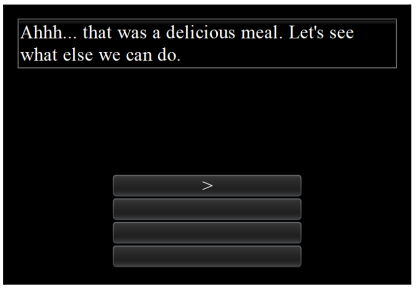
Home Food Screen

If the eating option is chosen, the user can pick between steak, vegetable stir-fry and cereal. Going to the mall from this screen will prompt the Transport screen.

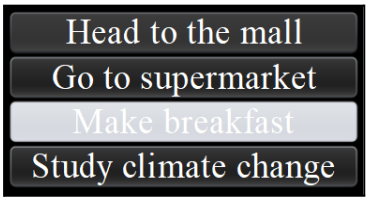




If the player has not bought the groceries and picks ‘Cereal’, the text area will prompt with the following and the option will be disabled.

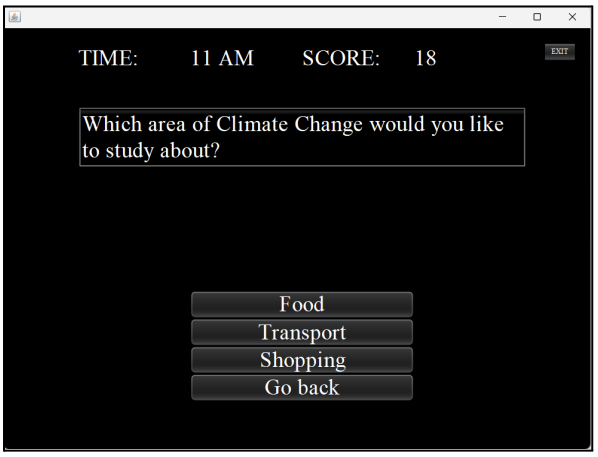


Otherwise, if any of the food choices are available and the action goes through, the filler screen is displayed where the header is updated again. Clicking the arrow option returns the player to the Home screen, where the eating option is now disabled until the next mealtime.

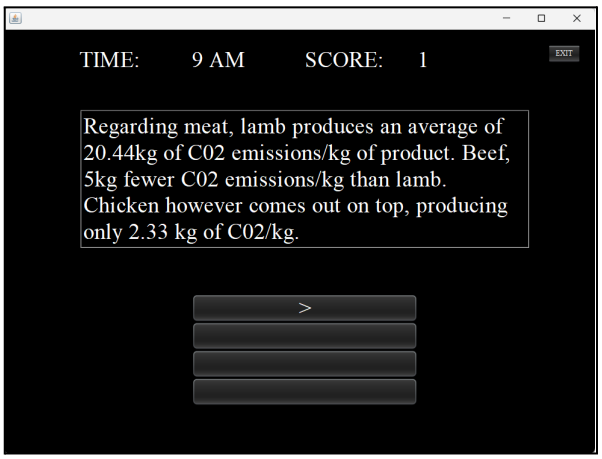


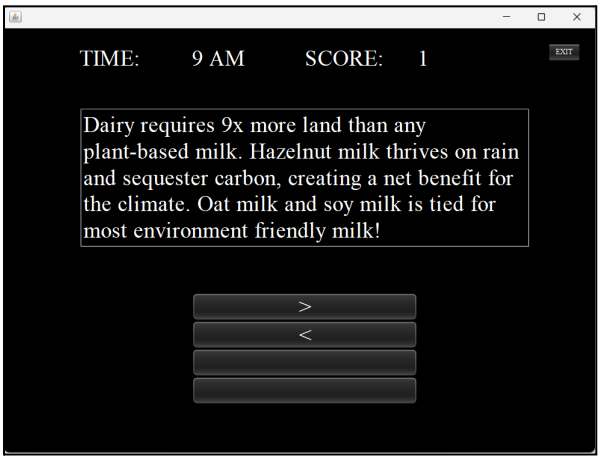
Study Topic Screen (Main)

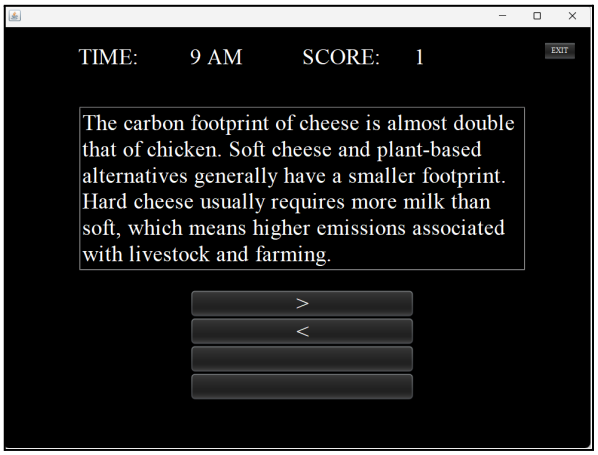
With the ‘Study climate change’ option, the player has 3 topics to study from. Each of them has hints to the choices that yield the highest score throughout the game. Once any of them are chosen, the user must go through the topic until the end.

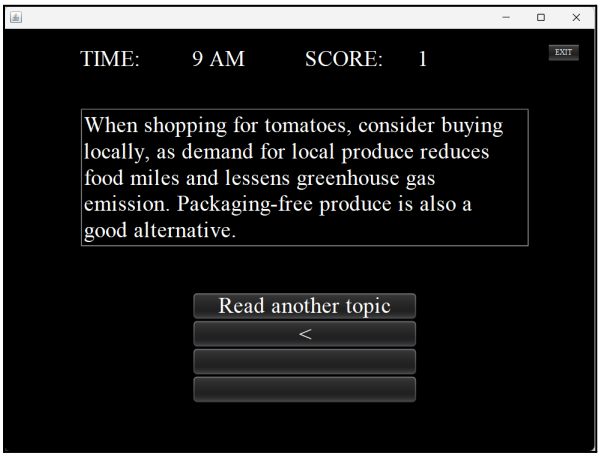


Study Food Screen





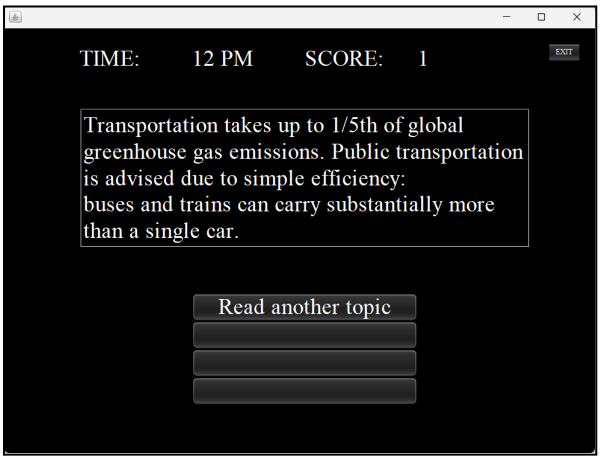




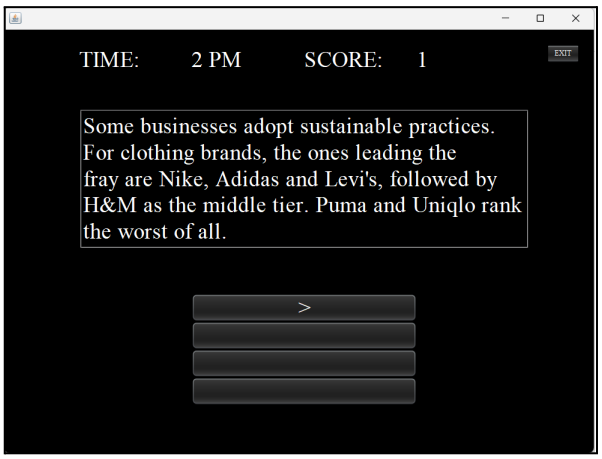
The player is able to switch between these 4 screens while studying about the Food topic. ‘Read another topic’ will send the player back to the Study screen, with the time incremented.

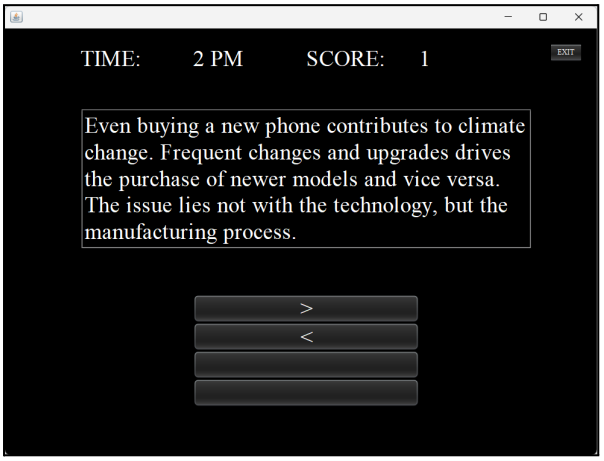
Study Transport Screen

The player is displayed the following screen while studying about the Transport topic. ‘Read another topic’ will send the player back to the Study screen, with the time incremented.

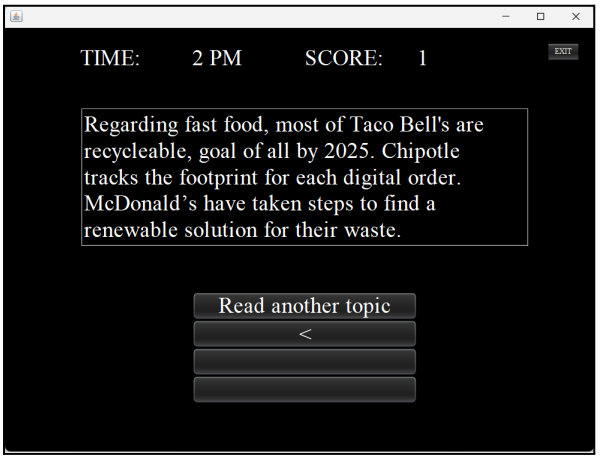


Study Shopping Screen





The player is able to switch between these 3 screens while studying about the Shopping topic. ‘Read another topic’ will send the player back to the Study screen, with the time incremented.



Game Over Screen

Once the time has gone past 7 PM, the game finishes and the Game Over screen will be displayed. Here, the final score is presented and the player has the choice to restart the game or return to the Home Page.

7) Source Code & Requirements Highlighted

7.1 Admin Class

```
package com.mycompany.ClimateEducationApp;

public class Admin {
    // Data fields(username and password)
    private String username;
    private String password;

    // Class constructor
    public Admin(String username,String password) {
        this.username = username;
        this.password = password;
    }

    // Username getter method
    public String getUsername() {
        return username;
    }

    // Password getter method
    public String getPassword() {
        return password;
    }
}
```

7.2 AdminControlUserData Class

```
package com.mycompany.ClimateEducationApp;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.io.*;
import java.util.ArrayList;

public class AdminControlUserData extends javax.swing.JFrame {
    private javax.swing.JButton addbutton;
    private javax.swing.JButton clearButton;
    private javax.swing.JButton deleteButton;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel25;
    private javax.swing.JLabel jLabel26;
    private javax.swing.JLabel jLabel27;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton refreshBtn;
    private javax.swing.JButton updateButton;
    private javax.swing.JTable userDataTable;
    private javax.swing.JTextField userEmail;
    private javax.swing.JTextField userFullName;
    private javax.swing.JTextField userPassword;
```

```
    private String username;
    private String email;
    private String password;
    private DefaultTableModel model;

    // Class constructor, loads window with User Data
    public AdminControlUserData() {
        initComponents();
        LoadUserFromFile();
        setUpListSelectionListener();
    }

    // Exit button method
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        AdminInterface adminUI = new AdminInterface();
        adminUI.setVisible(true);
        adminUI.pack();
        adminUI.setLocationRelativeTo(null);

        this.dispose();
    }

    // Add user data button method
    private void addbuttonActionPerformed(java.awt.event.ActionEvent evt) {
        String username = userFullName.getText();
        String email = userEmail.getText();
        String password = userPassword.getText();
        if (username.trim().isEmpty() || email.trim().isEmpty() || password.trim().isEmpty()) {
            JOptionPane.showMessageDialog(this, "There is no data to be added!");
        } else {
            model.addRow(new Object[]{username, email, password});
            appendToFile(username, email, password);
            loadFromFile();
        }
    }

    // Writes new user data to userDetails.txt file
    private void appendToFile(String username, String password, String email) {
        String filePath = "src/com/mycompany/ClimateEducationApp/userDetails.txt";
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath, true))) {
            writer.write(username + "," + password + "," + email + "\n");
        } catch (IOException e) {
            System.err.println("Error writing to file: " + e.getMessage());
        }
    }

    // Loads user data from userDetails.txt file
    private void loadFromFile() {
        model.setRowCount(0);
        String filePath = "src/com/mycompany/ClimateEducationApp/userDetails.txt";
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] userData = line.split(",");
                if (userData.length == 3) {
                    model.addRow(userData);
                }
```

```
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
    }

    // Updates selected table row with new user data
    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedRow = userDataTable.getSelectedRow();
        if (selectedRow != -1) {
            String username = userFullName.getText();
            String email = userEmail.getText();
            String password = userPassword.getText();

            model.setValueAt(username, selectedRow, 0);
            model.setValueAt(email, selectedRow, 1);
            model.setValueAt(password, selectedRow, 2);
            updateFile(selectedRow, username, email, password);
            loadFromFile();
        }
    }

    // Delete button method
    private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedRow = userDataTable.getSelectedRow();
        if (selectedRow != -1) {
            deleteRow(selectedRow);
        } else {
            JOptionPane.showMessageDialog(null, "Please select a row to delete.");
        }
    }

    // Clear input fields method
    private void clearButtonActionPerformed(java.awt.event.ActionEvent evt) {
        userFullName.setText("");
        userEmail.setText("");
        userPassword.setText("");
    }

    // Refresh button method
    private void refreshBtnActionPerformed(java.awt.event.ActionEvent evt) {
        userFullName.setText("");
        userEmail.setText("");
        userPassword.setText("");
        LoadUserFromFile();
    }

    // Search button method
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        String targetUser = userFullName.getText();
        searchDataByUser(targetUser);
    }

    // Searches user data in userDetails.txt
    public void searchDataByUser(String targetUser) {
        String filePath = "src/com/mycompany/ClimateEducationApp/userDetails.txt";
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
```

```
            String line;
            while ((line = reader.readLine()) != null) {
                String[] userData = line.split(",");
                if (userData.length == 3 && userData[0].equals(targetUser)) {
                    model.setRowCount(0);
                    // Location found, assign data to variables
                    username = userData[0];
                    email = userData[1];
                    password = userData[2];
                    model.addRow(new Object[]{username, email, password});
                    return; // Optional: If you want to stop searching after the first match
                }
            }
            // Location not found
            JOptionPane.showMessageDialog(null, "User '" + targetUser + "' not found.");
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
        userDataTable.setModel(model);
        userDataTable.getSelectionModel().addListSelectionListener(e -> {
            int selectedRow = userDataTable.getSelectedRow();
            if (selectedRow != -1) {
                displaySelectedRowData(selectedRow);
            }
        });
    }

    // Loads user data from userDetails.txt file into program
    private void LoadUserFromFile() {
        String[] columnNames = {"Username", "Email", "Password"};
        model = new DefaultTableModel(columnNames, 0);
        String filePath = "src/com/mycompany/ClimateEducationApp/userDetails.txt";
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] userData = line.split(",");
                if (userData.length == 3) {
                    model.addRow(userData);
                }
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
        userDataTable.setModel(model);
        userDataTable.getSelectionModel().addListSelectionListener(e -> {
            int selectedRow = userDataTable.getSelectedRow();
            if (selectedRow != -1) {
                displaySelectedRowData(selectedRow);
            }
        });
    }

    // Update file after removing the row
    private void deleteRow(int rowIndex) {
        model.removeRow(rowIndex);
        deleteRowFromFile(); // Update file after removing the row
        loadFromFile();
```

```
    }

    // Deletes user data of selected row from userDetails.txt file
    private void deleteRowFromFile() {
        String filePath = "src/com/mycompany/ClimateEducationApp/userDetails.txt";
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {
            for (int row = 0; row < model.getRowCount(); row++) {
                StringBuilder rowData = new StringBuilder();
                for (int col = 0; col < model.getColumnCount(); col++) {
                    if (col > 0) {
                        rowData.append(",");
                    }
                    rowData.append(model.getValueAt(row, col));
                }
                writer.write(rowData.toString() + "\n");
            }
        } catch (IOException e) {
            System.err.println("Error writing to file: " + e.getMessage());
        }
    }

    // Method to initialize text fields with selected row data
    private void displaySelectedRowData(int rowIndex) {
        userFullName.setText(model.getValueAt(rowIndex, 0).toString());
        userEmail.setText(model.getValueAt(rowIndex, 1).toString());
        userPassword.setText(model.getValueAt(rowIndex, 2).toString());
    }

    // Updates userDetails.txt file with new user data
    private void updateFile(int rowIndex, String username, String email, String password) {
        try {
            ArrayList<String> lines = new ArrayList<>();
            String filePath = "src/com/mycompany/ClimateEducationApp/userDetails.txt";
            try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
                String line;
                while ((line = reader.readLine()) != null) {
                    lines.add(line);
                }
            }

            String selectedLine = lines.get(rowIndex);
            String[] userData = selectedLine.split(",");
            userData[0] = username;
            userData[1] = email;
            userData[2] = password;
            lines.set(rowIndex, String.join(",", userData));

            try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {
                for (String line : lines) {
                    writer.write(line + "\n");
                }
            }
        } catch (IOException e) {
            System.err.println("Error updating file: " + e.getMessage());
        }
    }
```

```
    // List selection method
    private void setUpListSelectionListener() {
        userDataTable.getSelectionModel().addListSelectionListener(e -> {
            int selectedRow = userDataTable.getSelectedRow();
            if (selectedRow != -1) {
                displaySelectedRowData(selectedRow);
            }
        });
    }
}
```

7.3 AdminInterface Class

```
package com.mycompany.ClimateEducationApp;

public class AdminInterface extends javax.swing.JFrame {
    // Creates new form AdminInterface
    public AdminInterface() {
        initComponents();
    }

     // Exit button method
    private void userLoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        LoginPage LoginFrame = new LoginPage();
        LoginFrame.setVisible(true);
        LoginFrame.pack();
        LoginFrame.setLocationRelativeTo(null);
        // Closes AdminLoginFrame
        this.dispose();
    }

    // Toggles password text visibility
    private void showPwordActionPerformed(java.awt.event.ActionEvent evt) {
        if (showPword.isSelected()) {//show entered password
            AdminPword.setEchoChar((char) 0);
        } else {//hide entered password
            AdminPword.setEchoChar('*');
        }
    }

    // Authenticate admin login credentials
    private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String email = AdminId.getText();
        String password = new String(AdminPword.getPassword());
        Admin admin1 = new Admin("admin001","passadmin001");
        if (admin1.getUsername().equals(email) && admin1.getPassword().equals(password)) {
            AdminInterface adminUI = new AdminInterface();
            adminUI.setVisible(true);
            adminUI.pack();
            adminUI.setLocationRelativeTo(null);
            this.dispose();
        } else {
             JOptionPane.showMessageDialog(this, "Invalid username or password!");
        }
    }

    // Variables declaration - do not modify
    private javax.swing.JTextField AdminId;
```

```
    private javax.swing.JPasswordField AdminPword;
    private javax.swing.JButton LoginButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel left;
    private javax.swing.JCheckBox showPword;
    private javax.swing.JButton userLoginButton;
    // End of variables declaration
}
```

7.4 AdminLogin Class

```
package com.mycompany.ClimateEducationApp;

import javax.swing.JOptionPane;

public class AdminLogin extends javax.swing.JFrame {
    // Creates new form AdminLogin
    public AdminLogin() {
        initComponents();
    }

    // Exit button method
    private void userLoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        LoginPage LoginFrame = new LoginPage();
        LoginFrame.setVisible(true);
        LoginFrame.pack();
        LoginFrame.setLocationRelativeTo(null);
        // Closes AdminLoginFrame
        this.dispose();
    }

    // Toggles password text visiblity
    private void showPwordActionPerformed(java.awt.event.ActionEvent evt) {
        if (showPword.isSelected()) {//show entered password
            AdminPword.setEchoChar((char) 0);
        } else {//hide entered password
            AdminPword.setEchoChar('*');
        }
    }

    // Authenticate admin login credentials
    private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String email = AdminId.getText();
        String password = new String(AdminPword.getPassword());
        Admin admin1 = new Admin("admin001","passadmin001");
        if (admin1.getUsername().equals(email) && admin1.getPassword().equals(password)) {
            AdminInterface adminUI = new AdminInterface();
            adminUI.setVisible(true);
            adminUI.pack();
            adminUI.setLocationRelativeTo(null);

            this.dispose();
```

```
        } else {
            JOptionPane.showMessageDialog(this, "Invalid username or password!");
        }
    }

    // Variables declaration - do not modify
    private javax.swing.JTextField AdminId;
    private javax.swing.JPasswordField AdminPword;
    private javax.swing.JButton LoginButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel left;
    private javax.swing.JCheckBox showPword;
    private javax.swing.JButton userLoginButton;
    // End of variables declaration
}
```

7.5 CF_UserUI Class

```
package com.mycompany.ClimateEducationApp;

import javax.swing.JOptionPane;
import javax.swing.*;
import java.util.List;
import java.util.ArrayList;

public class CF_UserUI extends javax.swing.JFrame {
    int numberOfFlights;
    double totalFlightDuration, electricBill, gasBill, oilBill, mileage, totalFootprint, creditScore;
    String title, choiceNewspaper, choiceAluminium;
    private List<JTextField> textFields;

    // Creates new form CF_UserUI
    public CF_UserUI() {
        initComponents();

        textFields = new ArrayList<>();
        textFields.add(flighttakenTF);
        textFields.add(flightDurationTF);
        textFields.add(electricBillTF);
        textFields.add(gasBillTF);
        textFields.add(carMileageTF);
        textFields.add(oilBillTF);
        textFields.add(NewspaperTF);
        textFields.add(ALTF);
    }

    // InputMismatchException error handling for Integer input fields
    private class IntegerInputVerifier extends InputVerifier {
        @Override
        public boolean verify(JComponent input) {
```

```
            JTextField textField = (JTextField) input;
            String text = textField.getText().trim();

            try {
                // Attempt to parse the input as an integer
                int intValue = Integer.parseInt(text);
                // Perform additional validation if needed (e.g., range checks)
                if (intValue >= 0) {
                    return true; // Input is valid
                } else {
                    showErrorDialog("Please enter a non-negative integer value.");
                    return false; // Input is invalid
                }
            } catch (NumberFormatException e) {
                showErrorDialog("Please enter a valid integer value.");
                return false; // Input is invalid
            }
        }
    }

    // InputMismatchException error handling for Double input fields
    private class DoubleInputVerifier extends InputVerifier {
        @Override
        public boolean verify(JComponent input) {
            JTextField textField = (JTextField) input;
            String text = textField.getText().trim();

            try {
                // Attempt to parse the input as a double
                double doubleValue = Double.parseDouble(text);
                // Perform additional validation if needed (e.g., range checks)
                if (doubleValue >= 0) {
                    return true; // Input is valid
                } else {
                    showErrorDialog("Please enter a non-negative double value.");
                    return false; // Input is invalid
                }
            } catch (NumberFormatException e) {
                showErrorDialog("Please enter a valid double value.");
                return false; // Input is invalid
            }
        }
    }

    // InputMismatchException error handling for Boolean input fields
    private class BooleanInputVerifier extends InputVerifier {
        @Override
        public boolean verify(JComponent input) {
            JTextField textField = (JTextField) input;
            String text = textField.getText().trim().toLowerCase(); // Convert to lowercase for
case-insensitivity

            if (text.equals("yes") || text.equals("no")) {
                return true; // Input is valid
            } else {
                showErrorDialog("Please enter either 'Yes' or 'No'.");
                return false; // Input is invalid
```

```
        }
    }
}

// Checks if no fields are empty
private boolean validateTextFields() {
    for (JTextField textField : textFields) {
        if (textField.getText().trim().isEmpty()) {
            showErrorDialog("Please fill in all information to retrieve your SCORE and TITLE.");
            return false;
        }
    }
    return true;
}

// Displays error message for input error
private void showErrorDialog(String errorMessage) {
    JOptionPane.showMessageDialog(null, errorMessage, "Input Error",
JOptionPane.ERROR_MESSAGE);
}

// Exit button method
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    UserInterface UI = new UserInterface();
    UI.setVisible(true);
    UI.pack();
    UI.setLocationRelativeTo(null);

    this.dispose();
}

// Submit button method
private void submitBtnActionPerformed(java.awt.event.ActionEvent evt) {
    if (validateTextFields() == true &&
        new IntegerInputVerifier().verify(flighttakenTF)==true&&
        new DoubleInputVerifier().verify(flightDurationTF)==true&&
        new DoubleInputVerifier().verify(electricBillTF)==true&&
        new DoubleInputVerifier().verify(gasBillTF)==true&&
        new DoubleInputVerifier().verify(carMileageTF)==true&&
        new DoubleInputVerifier().verify(oilBillTF)==true&&
        new BooleanInputVerifier().verify(NewspaperTF)==true&&
        new BooleanInputVerifier().verify(ALTF)==true
        ) {
            numberOfFlights = Integer.parseInt(flighttakenTF.getText());
            totalFlightDuration = Double.parseDouble(flightDurationTF.getText());
            electricBill = Double.parseDouble(electricBillTF.getText());
            gasBill = Double.parseDouble(gasBillTF.getText());
            oilBill = Double.parseDouble(oilBillTF.getText());
            mileage = Double.parseDouble(carMileageTF.getText());
            choiceNewspaper = NewspaperTF.getText();
            choiceAluminium = ALTF.getText();
            calculate();
            displayResults();
    }
}

// Calculates carbon footprint using all user inputs
```

```
    public void calculate() {
        FlightFootprint calculator = new FlightFootprint(totalFlightDuration, numberOfFlights, 0);
        double totalFlightFootprint = calculator.calculatefootprint();

        HouseFootprint calculation = new HouseFootprint(electricBill, gasBill, oilBill);
        double totalHouseFootprint = calculation.calculateFootprint();

        CarbonFootprint calculateCarFootprint = new CarbonFootprint(mileage);
        double totalCarFootprint = calculateCarFootprint.calculateCarFootprint();

        Recycle recycle = new Recycle(choiceNewspaper, choiceAluminium);
        double totalNewspaperFootprint = recycle.calculateNewspaper();
        double totalAluminiumFootprint = recycle.calculateAluminium();

        double footprint = totalHouseFootprint + totalCarFootprint
                + totalFlightFootprint + totalNewspaperFootprint + totalAluminiumFootprint;
        this.totalFootprint = footprint;

        double credit = footprint * 0.069;
        this.creditScore = credit;
    }

    // Displays credit score and awards title
    public void displayResults() {
        // lower the credit score, better the titles
        JOptionPane.showMessageDialog(null, " Your total carbon footprint of the year is " +
totalFootprint + " metric tons of CO2");
        if (creditScore <= 2000) {
            JOptionPane.showMessageDialog(null, "Your total carbon credit score is " +
this.creditScore);
            title = "Climate Champion";
            JOptionPane.showMessageDialog(null, "You have earned the title: " + title);
        } else if (creditScore <= 4000) {
            JOptionPane.showMessageDialog(null, "Your total carbon credit score is " +
this.creditScore);
            title = "Earth Advocate";
            JOptionPane.showMessageDialog(null, "You have earned the title: " + title);
        } else if (creditScore <= 6000) {
            JOptionPane.showMessageDialog(null, "Your total carbon credit score is " +
this.creditScore);
            title = "Carbon Neutral Pro";
            JOptionPane.showMessageDialog(null, "You have earned the title: " + title);
        } else if (creditScore <= 8000) {
            JOptionPane.showMessageDialog(null, "Your total carbon credit score is " +
this.creditScore);
            title = "Green Guardian";
            JOptionPane.showMessageDialog(null, "You have earned the title: " + title);
        } else if (creditScore <= 10000) {
            JOptionPane.showMessageDialog(null, "Your total carbon credit score is " +
this.creditScore);
            title = "Footprint Fighter";
            JOptionPane.showMessageDialog(null, "You have earned the title: " + title);
        } else if (creditScore <= 11000) {
            JOptionPane.showMessageDialog(null, "Your total carbon credit score is " +
this.creditScore);
            title = "Eco Warrior";
            JOptionPane.showMessageDialog(null, "You have earned the title: " + title);
```

```
        } else {
            JOptionPane.showMessageDialog(null, "Your total carbon credit score is " +
this.creditScore);
            title = "Sustainable Hero";
            JOptionPane.showMessageDialog(null, "You have earned the title: " + title);
        }
    }

    // Variables declaration - do not modify
    private javax.swing.JTextField ALTF;
    private javax.swing.JTextField NewspaperTF;
    private javax.swing.JTextField carMileageTF;
    private javax.swing.JTextField electricBillTF;
    private javax.swing.JTextField flightDurationTF;
    private javax.swing.JTextField flighttakenTF;
    private javax.swing.JTextField gasBillTF;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JTextField oilBillTF;
    private javax.swing.JButton submitBtn;
    // End of variables declaration

}
```

7.6 CI_User Class

```
package com.mycompany.ClimateEducationApp;

public class CI_User extends javax.swing.JFrame {
    StringBuilder newsBuilder = new StringBuilder();
    StringBuilder infoBuilder = new StringBuilder();

    // Creates new form CI_User
    public CI_User() {
        initComponents();
        displayNews();
        Blog.setEditable(false);
        Blog.setContentType("text/html");
        Blog.setText(newsBuilder.toString());
    }


    // Display the news in the JTextArea
    private void displayNews() {
        newsBuilder.append("<html><b><font size='5'>--------- Temperature News
---------</font></b><br>");
```

```
        newsBuilder.append("Kuala Lumpur 27°C<br>");
        newsBuilder.append("Johor Baru: 25°C<br>");
        newsBuilder.append("Ipoh: 26°C<br>");
        newsBuilder.append("Kuching: 26°C<br><br>");

        newsBuilder.append("<b><font size='5'>--------- Energy Breakthrough News
---------</font></b><br>");
        newsBuilder.append("SCIENTISTS DISCOVER NUCLEAR ENERGY
BREAKTHROUGH<br>");
        newsBuilder.append("Nuclear scientists say this is a key step toward producing clean and
potentially cheap power,<br>");
        newsBuilder.append("though they warn the fledgling form of energy still has a long way to
go before<br>");
        newsBuilder.append("it becomes a viable option. <br><br>");

        newsBuilder.append("<b><font size='5'>--------- Plastic Waste News
---------</font></b><br>");
        newsBuilder.append("Since 2018, more than 1,000 organisations have given their backing
to the Global Commitment,<br>");
        newsBuilder.append("led by the Foundation in partnership with the UN Environment
Programme, to stop plastic<br>");
        newsBuilder.append("packaging from becoming waste.<br>");
        newsBuilder.append("And also increasing their use of recycled plastics by 1.5 million tonnes
per annum<br></html>");

        // Set the text of the JTextArea
        Blog.setText(newsBuilder.toString());
    }

    // Display the information in the JTextField
    private void displayInfo() {
        infoBuilder.append("<html><b><font size='7'>-------- Interactive Info System
--------</font></b><br>");
        infoBuilder.append("<b><font size='5'>--------- Understanding Climate Change
---------</font></b><br>");
        infoBuilder.append("Climate change is a long-term change in the average weather
patterns.<br>");
        infoBuilder.append("It is primarily caused by human activities, including the burning of fossil
fuels.<br>");

        infoBuilder.append("<b><font size='5'>--------- Energy Breakthrough News
---------</font></b><br>");
        infoBuilder.append("To reduce carbon emissions, individuals can use energy-efficient
appliances<br>");
        infoBuilder.append("drive less, and support renewable energy sources.<br>");

        infoBuilder.append("<b><font size='5'>--------- Sustainable Living ---------</font></b><br>");
        infoBuilder.append("Sustainable living involves making choices that have a positive impact
on the environment.<br>");
        infoBuilder.append("This includes reducing waste, conserving energy, and supporting
eco-friendly products.<br></html>");

        // Set the text of the JTextField
        Blog.setText(infoBuilder.toString());
    }

    // NewsFeed button method
```

```
    private void NewsFeedBtnActionPerformed(java.awt.event.ActionEvent evt) {
        Blog.setText("");
        displayNews();
    }

    // Info System button method
    private void InfoBtnActionPerformed(java.awt.event.ActionEvent evt) {
        Blog.setText("");
        displayInfo();
    }

    // Exit button method
    private void exitButtonActionPerformed(java.awt.event.ActionEvent evt) {
        UserInterface UI = new UserInterface();
        UI.setVisible(true);
        UI.pack();
        UI.setLocationRelativeTo(null);

        this.dispose();
    }

    // Share button method
    private void shareButtonActionPerformed(java.awt.event.ActionEvent evt) {
        getBlogText();
        shareContent();
    }

    // Opens infographics & videos window
    private void infografActionPerformed(java.awt.event.ActionEvent evt) {
        CI_UserInfographicAndVid UI = new CI_UserInfographicAndVid();
        UI.setVisible(true);
        UI.pack();
        UI.setLocationRelativeTo(null);

        this.dispose();
    }

    // Getter method for Blog content
    public String getBlogText() {
        return Blog.getText();
    }

    // Displays shared content in dashboard of Home page
    private void shareContent() {
        // Assuming you have an instance of UserInterface, replace UIInstance with your actual
instance
        UserInterface UIInstance = new UserInterface();
        // Get the text from the CI_User Blog editor pane
        String blogText = getBlogText();
        // Set the HTML content in the UserInterface CItext editor pane
        UIInstance.setCItextHTML(blogText);
        UIInstance.setVisible(true);
        UIInstance.setLocationRelativeTo(null);

        this.dispose();
    }
```

```
    // Variables declaration - do not modify
    private javax.swing.JEditorPane Blog;
    private javax.swing.JButton InfoBtn;
    private javax.swing.JButton NewsFeedBtn;
    private javax.swing.JButton exitButton;
    private javax.swing.JButton infograf;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JButton shareButton;
    // End of variables declaration
}
```

7.7 CI_UserInfographicAndVid Class

```
package com.mycompany.ClimateEducationApp;

import java.awt.Desktop;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;

public class CI_UserInfographicAndVid extends javax.swing.JFrame {
    // Creates new form CI_UserInfographicAndVid
    public CI_UserInfographicAndVid() {
        initComponents();
    }

    // Exit button method
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        CI_User UI = new CI_User();
        UI.setVisible(true);
        UI.pack();
        UI.setLocationRelativeTo(null);

        this.dispose();
    }

    // Opens video link in browser when clicked
    private void jTextField1MouseClicked(java.awt.event.MouseEvent evt) {
        try {
            URI uri = new URI(jTextField1.getText());
            Desktop.getDesktop().browse(uri);
        } catch (IOException | URISyntaxException e) {
            e.printStackTrace();
        }

    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
```

```
    private javax.swing.JPanel jPanel3;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration
}
```

7.8 CarbonFootprint Class

```
package com.mycompany.ClimateEducationApp;

public class CarbonFootprint {
    public double mileage;

    // Initializes mileage value
    public CarbonFootprint (double mileage){
        this.mileage = mileage;
    }

    // Calculates Car footprint
    public double calculateCarFootprint() {
        return (mileage * 0.79);
    }
}
```

7.9 EA_AdminMenu Class

```
package com.mycompany.ClimateEducationApp;

import javax.swing.*;
import java.io.*;
import java.util.ArrayList;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import javax.swing.table.DefaultTableModel;

public class EA_AdminMenu extends javax.swing.JFrame {
    private DefaultTableModel model;

    // Creates new form EA_AdminMenu
    public EA_AdminMenu() {
        initComponents();
        String[] columnNames = {"Location", "Danger", "Preventive Measure"};
        model = new DefaultTableModel(columnNames, 0);
        String filePath = "src/com/mycompany/ClimateEducationApp/WarningInfoStore.txt";
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] WarningData = line.split("/");
                if (WarningData.length == 3) {
                    model.addRow(WarningData);
                }
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
        warningInfoTable.setModel(model);
        warningInfoTable.getSelectionModel().addListSelectionListener(e -> {
            int selectedRow = warningInfoTable.getSelectedRow();
            if (selectedRow != -1) {
```

```
                displaySelectedRowData(selectedRow);
            }
        });
    }

    // Updates selected table row
    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedRow = warningInfoTable.getSelectedRow();
        if (selectedRow != -1) {
            String location = locationText.getText();
            String danger = dangerText.getText();
            String warning = warningText.getText();

            model.setValueAt(location, selectedRow, 0);
            model.setValueAt(danger, selectedRow, 1);
            model.setValueAt(warning, selectedRow, 2);
            updateFile(selectedRow, location, danger, warning);
            loadFromFile();
        }
    }

    // Delete selected table row
    private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedRow = warningInfoTable.getSelectedRow();
        if (selectedRow != -1) {
            deleteRow(selectedRow);
        } else {
            JOptionPane.showMessageDialog(null, "Please select a row to delete.");
        }
    }

    // Adds new warning data to table
    private void addbuttonActionPerformed(java.awt.event.ActionEvent evt) {
        String location = locationText.getText();
        String danger = dangerText.getText();
        String warning = warningText.getText();

        if (location.trim().isEmpty() || danger.trim().isEmpty() || warning.trim().isEmpty()) {
            JOptionPane.showMessageDialog(this, "There is no data to be added!");
        } else {
            model.addRow(new Object[]{location, danger, warning});
            appendToFile(location, danger, warning);
            loadFromFile();
        }
    }

    // Exit button method
    private void exitButtonActionPerformed(java.awt.event.ActionEvent evt) {
        AdminInterface adminUI = new AdminInterface();
        adminUI.setVisible(true);
        adminUI.pack();
        adminUI.setLocationRelativeTo(null);

        this.dispose();
    }

    // Clears input fields
```

```
    private void clearButtonActionPerformed(java.awt.event.ActionEvent evt) {
        locationText.setText("");
        dangerText.setText("");
        warningText.setText("");
    }

    // Delete row method
    private void deleteRow(int rowIndex) {
        model.removeRow(rowIndex);
        deleteRowFromFile(); // Update file after removing the row
        loadFromFile();
    }

    // Writes data to WarningInfoStore.txt file
    private void appendToFile(String location, String danger, String warning) {
        String filePath = "src/com/mycompany/ClimateEducationApp/WarningInfoStore.txt";

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath, true))) {
            writer.write(location + "/" + danger + "/" + warning + "\n");
        } catch (IOException e) {
            System.err.println("Error writing to file: " + e.getMessage());
        }
    }

    // Loads data from WarningInfoStore.txt file
    private void loadFromFile() {
        model.setRowCount(0);
        String filePath = "src/com/mycompany/ClimateEducationApp/WarningInfoStore.txt";

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] userData = line.split("/");
                if (userData.length == 3) {
                    model.addRow(userData);
                }
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
    }

    // Deletes data from WarningInfoStore.txt file
    private void deleteRowFromFile() {
        String filePath = "src/com/mycompany/ClimateEducationApp/WarningInfoStore.txt";
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {
            for (int row = 0; row < model.getRowCount(); row++) {
                StringBuilder rowData = new StringBuilder();
                for (int col = 0; col < model.getColumnCount(); col++) {
                    if (col > 0) {
                        rowData.append("/");
                    }
                    rowData.append(model.getValueAt(row, col));
                }
                writer.write(rowData.toString() + "\n");
            }
        } catch (IOException e) {
```

```
            System.err.println("Error writing to file: " + e.getMessage());
        }
    }

    // Method to initialize text fields with selected row data
    private void displaySelectedRowData(int rowIndex) {
        locationText.setText(model.getValueAt(rowIndex, 0).toString());
        dangerText.setText(model.getValueAt(rowIndex, 1).toString());
        warningText.setText(model.getValueAt(rowIndex, 2).toString());
    }

    // Updates data in WarningInfoStore.txt file
    private void updateFile(int rowIndex, String location, String danger, String warning) {
        try {
            ArrayList<String> lines = new ArrayList<>();
            String filePath = "src/com/mycompany/ClimateEducationApp/WarningInfoStore.txt";
            try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
                String line;
                while ((line = reader.readLine()) != null) {
                    lines.add(line);
                }
            }

            String selectedLine = lines.get(rowIndex);
            String[] WarningData = selectedLine.split("/");
            WarningData[0] = location;
            WarningData[1] = danger;
            WarningData[2] = warning;
            lines.set(rowIndex, String.join("/", WarningData));

            try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {
                for (String line : lines) {
                    writer.write(line + "\n");
                }
            }
        } catch (IOException e) {
            System.err.println("Error updating file: " + e.getMessage());
        }
    }

    // Variables declaration - do not modify
    private javax.swing.JButton addbutton;
    private javax.swing.JButton clearButton;
    private javax.swing.JTextField dangerText;
    private javax.swing.JButton deleteButton;
    private javax.swing.JButton exitButton;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel31;
    private javax.swing.JLabel jLabel32;
    private javax.swing.JLabel jLabel33;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField locationText;
    private javax.swing.JButton updateButton;
```

```
    private javax.swing.JTable warningInfoTable;
    private javax.swing.JTextField warningText;
    // End of variables declaration
}
```

7.10 EA_User Class

```
package com.mycompany.ClimateEducationApp;

import javax.swing.*;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import javax.swing.table.DefaultTableModel;

public class EA_User extends javax.swing.JFrame {
    private String location;
    private String danger;
    private String warning;
    private DefaultTableModel model;

    // Creates new form EA_User
    public EA_User() {
        initComponents();
        String[] columnNames = {"Location", "Danger", "Preventive Measure"};
        model = new DefaultTableModel(columnNames, 0);
        String filePath = "src/com/mycompany/ClimateEducationApp/WarningInfoStore.txt";
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] WarningData = line.split("/");
                if (WarningData.length == 3) {
                    model.addRow(WarningData);
                }
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
        warningInfoTable.setModel(model);
        warningInfoTable.getSelectionModel().addListSelectionListener(e -> {
            int selectedRow = warningInfoTable.getSelectedRow();
            if (selectedRow != -1) {
                displaySelectedRowData(selectedRow);
            }
        });
    }

    // Exit button method
    private void exitButtonActionPerformed(java.awt.event.ActionEvent evt) {
        UserInterface UI = new UserInterface();
        UI.setVisible(true);
        UI.pack();
        UI.setLocationRelativeTo(null);

        this.dispose();
    }

    // Share button method
```

```
    private void shareButtonActionPerformed(java.awt.event.ActionEvent evt) {
        int selectedRow = warningInfoTable.getSelectedRow();
        if (selectedRow != -1) {
            openWarningDashboard();
        } else {
            JOptionPane.showMessageDialog(null, "Please select a row to share.");
        }
    }

    // Search button method
    private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String targetLocation = locationText.getText();
        searchDataByLocation(targetLocation);
    }

    // Clears search input field
    private void refreshBtnActionPerformed(java.awt.event.ActionEvent evt) {
        locationText.setText("");
        loadFromFile();
    }

    // Shares content to dashboard in Home page
    private void openWarningDashboard() {
        UserInterface warningDashboard = new UserInterface();
        warningDashboard.setThreeLinesOfText(location, danger, warning);
        warningDashboard.setVisible(true);
        warningDashboard.setLocationRelativeTo(null);
        this.dispose();
    }

    // Searches data by String in search input field
    public void searchDataByLocation(String targetLocation) {
        String filePath = "src/com/mycompany/ClimateEducationApp/WarningInfoStore.txt";
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] warningData = line.split("/");
                if (warningData.length == 3 && warningData[0].equals(targetLocation)) {
                    model.setRowCount(0);
                    // Location found, assign data to variables
                    location = warningData[0];
                    danger = warningData[1];
                    warning = warningData[2];
                    model.addRow(new Object[]{location, danger, warning});

                    return; // Optional: If you want to stop searching after the first match
                }
            }
            // Location not found
            JOptionPane.showMessageDialog(null, "Location " + targetLocation + " not found.");
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
        warningInfoTable.setModel(model);
        warningInfoTable.getSelectionModel().addListSelectionListener(e -> {
            int selectedRow = warningInfoTable.getSelectedRow();
            if (selectedRow != -1) {
```

```
                displaySelectedRowData(selectedRow);
            }
        });
    }

    // Loads data from WarningInfoStore.txt file
    private void loadFromFile() {
        model.setRowCount(0);
        String filePath = "src/com/mycompany/ClimateEducationApp/WarningInfoStore.txt";
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] userData = line.split("/");
                if (userData.length == 3) {
                    model.addRow(userData);
                }
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
    }

    // Method to initialize text fields with selected row data
    private void displaySelectedRowData(int rowIndex) {
        location = model.getValueAt(rowIndex, 0).toString();
        danger = model.getValueAt(rowIndex, 1).toString();
        warning = model.getValueAt(rowIndex, 2).toString();
    }

    // Variables declaration - do not modify
    private javax.swing.JButton exitButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel32;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField locationText;
    private javax.swing.JButton refreshBtn;
    private javax.swing.JButton searchButton;
    private javax.swing.JButton shareButton;
    private javax.swing.JTable warningInfoTable;
    // End of variables declaration
}
```

7.11 FlightFootprint Class

```
package com.mycompany.ClimateEducationApp;

public class FlightFootprint {
    // Variables for flights
    double flightduration;
    int numberOfFlights;
    int flightfootprint;

    // Constructor method
    public FlightFootprint (double flightduration, int numberOfFlights, int flightfootprint){
```

```
        this.flightduration = flightduration;
        this.numberOfFlights = numberOfFlights;
        this.flightfootprint = flightfootprint;
    }

    // Calculates flight footprint
    public double calculatefootprint() {
        if (flightduration <= 4) {
            flightfootprint = numberOfFlights * 1100;
        } else if (flightduration > 4) {
            flightfootprint = numberOfFlights * 4400;
        } else {
            System.out.println("Error.");
        }
        return flightfootprint;
    }
}
```

7.12 Game Class

```
package com.mycompany.ClimateEducationApp;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JTextArea;
import java.awt.Container;
import java.awt.Color;
import java.awt.Font;
import java.awt.GridLayout;
import static com.mycompany.ClimateEducationApp.Game_Home.arriveHome;

public class Game {
    // Declaring Game window components
    public static JFrame window;
    public static Container con;
    public static JPanel
            titleNamePanel,
            startButtonPanel,
            exitButtonPanel,
            mainTextPanel,
            choiceButtonPanel,
            playerPanel,
            gameOverTitlePanel,
            gameOverTextPanel,
            yesButtonPanel,
            noButtonPanel;
    public static JLabel
            titleNameLabel,
            timeLabel,
            timeLabelNumber,
            scoreLabel,
            scoreLabelNumber,
            gameOverTitleLabel;
    public static JButton
            startButton,
            exitButton,
```

```
        choice1,
        choice2,
        choice3,
        choice4,
        yesButton,
        noButton;
public static JTextArea mainTextArea, gameOverTextArea;

// Initializing fonts for text
public static Font titleFont = new Font("Times New Roman", Font.PLAIN, 90);
public static Font normalFont = new Font("Times New Roman", Font.PLAIN, 30);
public static Font exitFont = new Font("Times New Roman", Font.PLAIN, 10);

// Declaring Game variables
public static int
        environmentalScore,
        timeCounter,
        mallScreenCounter,
        mallChoiceCounter,
        supermarketScreenCounter,
        supermarketChoiceCounter;
public static String
        position,
        location,
        mealTime;
public static boolean
        goneOut,
        tookBike,
        tookCar,
        tookPublic,
        fullBelly,
        ateAtHome,
        ateAtMall,
        ateAtSupermarket,
        shoppedClothes,
        shoppedPhone,
        boughtGrocery;
public static Game_TitleScreenHandler tsHandler = new Game_TitleScreenHandler();
public static Game_ChoiceHandler choiceHandler = new Game_ChoiceHandler();

// Game constructor
public Game() {
    window = new JFrame();
    window.setSize(800, 600);
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    window.getContentPane().setBackground(Color.black);
    window.setLayout(null);
    window.setVisible(true);
    window.setLocationRelativeTo(null);
    con = window.getContentPane();

    titleNamePanel = new JPanel();
    titleNamePanel.setBounds(100, 100, 600, 150);
    titleNamePanel.setBackground(Color.black);
    titleNameLabel = new JLabel("ONE  DAY");
    titleNameLabel.setForeground(Color.white);
    titleNameLabel.setFont(titleFont);
```

```
    startButtonPanel = new JPanel();
    startButtonPanel.setBounds(300, 375, 200, 50);
    startButtonPanel.setBackground(Color.black);

    startButton = new JButton("START");
    startButton.setBackground(Color.black);
    startButton.setForeground(Color.white);
    startButton.setFont(normalFont);
    startButton.addActionListener(tsHandler);
    startButton.setActionCommand("start");
    startButton.setFocusPainted(false);

    exitButtonPanel = new JPanel();
    exitButtonPanel.setBounds(725, 15, 40, 50);
    exitButtonPanel.setBackground(Color.black);

    exitButton = new JButton("EXIT");
    exitButton.setBackground(Color.black);
    exitButton.setForeground(Color.white);
    exitButton.setFont(exitFont);
    exitButton.addActionListener(tsHandler);
    exitButton.setActionCommand("exit");
    exitButton.setFocusPainted(false);

    titleNamePanel.add(titleNameLabel);
    startButtonPanel.add(startButton);
    exitButtonPanel.add(exitButton);

    con.add(titleNamePanel);
    con.add(startButtonPanel);
    con.add(exitButtonPanel);
}

// Create starting game screen
public static void createGameScreen() {
    titleNamePanel.setVisible(false);
    startButtonPanel.setVisible(false);

    mainTextPanel = new JPanel();
    mainTextPanel.setBounds(100, 100, 600, 250);
    mainTextPanel.setBackground(Color.black);
    con.add(mainTextPanel);

    mainTextArea = new JTextArea();
    mainTextArea.setBounds(100, 100, 600, 250);
    mainTextArea.setBackground(Color.black);
    mainTextArea.setForeground(Color.white);
    mainTextArea.setFont(normalFont);
    mainTextArea.setLineWrap(true);
    mainTextArea.setEditable(false);
    mainTextPanel.add(mainTextArea);

    choiceButtonPanel = new JPanel();
    choiceButtonPanel.setBounds(250, 350, 300, 150);
    choiceButtonPanel.setBackground(Color.black);
    choiceButtonPanel.setLayout(new GridLayout(4, 1));
```

```
con.add(choiceButtonPanel);

choice1 = new JButton();
choice1.setBackground(Color.black);
choice1.setForeground(Color.white);
choice1.setFont(normalFont);
choice1.setFocusPainted(false);
choice1.addActionListener(choiceHandler);
choice1.setActionCommand("c1");
choiceButtonPanel.add(choice1);

choice2 = new JButton();
choice2.setBackground(Color.black);
choice2.setForeground(Color.white);
choice2.setFont(normalFont);
choice2.setFocusPainted(false);
choice2.addActionListener(choiceHandler);
choice2.setActionCommand("c2");
choiceButtonPanel.add(choice2);

choice3 = new JButton();
choice3.setBackground(Color.black);
choice3.setForeground(Color.white);
choice3.setFont(normalFont);
choice3.setFocusPainted(false);
choice3.addActionListener(choiceHandler);
choice3.setActionCommand("c3");
choiceButtonPanel.add(choice3);

choice4 = new JButton();
choice4.setBackground(Color.black);
choice4.setForeground(Color.white);
choice4.setFont(normalFont);
choice4.setFocusPainted(false);
choice4.addActionListener(choiceHandler);
choice4.setActionCommand("c4");
choiceButtonPanel.add(choice4);

playerPanel = new JPanel();
playerPanel.setBounds(100, 15, 600, 50);
playerPanel.setBackground(Color.black);
playerPanel.setLayout(new GridLayout(1, 4));
con.add(playerPanel);

timeLabel = new JLabel("TIME:");
timeLabel.setForeground(Color.white);
timeLabel.setFont(normalFont);
playerPanel.add(timeLabel);

timeLabelNumber = new JLabel();
timeLabelNumber.setForeground(Color.white);
timeLabelNumber.setFont(normalFont);
playerPanel.add(timeLabelNumber);

scoreLabel = new JLabel("SCORE:");
scoreLabel.setForeground(Color.white);
scoreLabel.setFont(normalFont);
```

```
        playerPanel.add(scoreLabel);

        scoreLabelNumber = new JLabel();
        scoreLabelNumber.setForeground(Color.white);
        scoreLabelNumber.setFont(normalFont);
        playerPanel.add(scoreLabelNumber);

        gameSetup();
    }

    // Initializes game's variables
    public static void gameSetup() {
        timeCounter = 8;
        timeLabelNumber.setText(timeCounter + " AM");
        environmentalScore = 0;
        scoreLabelNumber.setText("" + environmentalScore);

        goneOut = false;
        tookBike = true;
        tookCar = true;
        tookPublic = true;
        fullBelly = false;
        ateAtHome = false;
        ateAtMall = false;
        ateAtSupermarket = false;
        shoppedClothes = false;
        shoppedPhone = false;
        boughtGrocery = false;

        mealTime = "breakfast";
        mallScreenCounter = 0;
        mallChoiceCounter = 0;
        supermarketScreenCounter = 0;
        supermarketChoiceCounter = 0;

        arriveHome();
    }

    // Toggles game over screen if user played long enough
    public static void showGameOverScreen() {
        if(timeCounter > 7 && timeLabelNumber.getText().contains("PM")) {
            playerPanel.setVisible(false);
            mainTextPanel.setVisible(false);
            choiceButtonPanel.setVisible(false);
            exitButtonPanel.setVisible(false);

            gameOverTitlePanel = new JPanel();
            gameOverTitlePanel.setBounds(100, 100, 600, 150);
            gameOverTitlePanel.setBackground(Color.black);
            gameOverTitleLabel = new JLabel("GAME OVER");
            gameOverTitleLabel.setForeground(Color.white);
            gameOverTitleLabel.setFont(titleFont);

            gameOverTitlePanel.add(gameOverTitleLabel);
            con.add(gameOverTitlePanel);

            gameOverTextPanel = new JPanel();
```

```
            gameOverTextPanel.setBounds(100, 250, 600, 150);
            gameOverTextPanel.setBackground(Color.black);

            gameOverTextArea = new JTextArea();
            gameOverTextArea.setBounds(100, 100, 600, 150);
            gameOverTextArea.setBackground(Color.black);
            gameOverTextArea.setForeground(Color.white);
            gameOverTextArea.setFont(normalFont);
            gameOverTextArea.setText("It's already too late in the day... time to rest.\n"
                    + "You've achieved a highscore of " + scoreLabelNumber.getText() + "!"
                    + "\nWould you like to play again?");
            gameOverTextArea.setLineWrap(true);
            gameOverTextArea.setEditable(false);
            mainTextPanel.add(mainTextArea);

            gameOverTextPanel.add(gameOverTextArea);
            con.add(gameOverTextPanel);

            yesButtonPanel = new JPanel();
            yesButtonPanel.setBounds(200, 400, 200, 50);
            yesButtonPanel.setBackground(Color.black);
            yesButton = new JButton("YES");
            yesButton.setBackground(Color.black);
            yesButton.setForeground(Color.white);
            yesButton.setFont(normalFont);
            yesButton.addActionListener(tsHandler);
            yesButton.setActionCommand("restart");
            yesButton.setFocusPainted(false);

            yesButtonPanel.add(yesButton);
            con.add(yesButtonPanel);

            noButtonPanel = new JPanel();
            noButtonPanel.setBounds(400, 400, 200, 50);
            noButtonPanel.setBackground(Color.black);
            noButton = new JButton("NO");
            noButton.setBackground(Color.black);
            noButton.setForeground(Color.white);
            noButton.setFont(normalFont);
            noButton.addActionListener(tsHandler);
            noButton.setActionCommand("exit");
            noButton.setFocusPainted(false);

            noButtonPanel.add(noButton);
            con.add(noButtonPanel);
        }
    }
}
```

7.13 Game_ChoiceHandler Class

```
package com.mycompany.ClimateEducationApp;

import javax.swing.JButton;
import static com.mycompany.ClimateEducationApp.Game.*;
import static com.mycompany.ClimateEducationApp.Game_Mall.*;
import static com.mycompany.ClimateEducationApp.Game_Supermarket.*;
import static com.mycompany.ClimateEducationApp.Game_Score.*;
```

```
import static com.mycompany.ClimateEducationApp.Game_Home.*;
import static com.mycompany.ClimateEducationApp.Game_Food.*;
import static com.mycompany.ClimateEducationApp.Game_Transport.*;
import static com.mycompany.ClimateEducationApp.Game_Time.timePassed;
import static com.mycompany.ClimateEducationApp.Game_Study.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Game_ChoiceHandler implements ActionListener {
    /**
     *
     * @param event
     */
    @Override
    public void actionPerformed(ActionEvent event) {
        // Initializes userChoice with button User chooses
        String userChoice = event.getActionCommand();

        // Performs action depending on current game page
        switch(position) {
            case "arriveHome":
                switch(userChoice) {
                    case "c1":
                        checkMallOptions(position);
                        break;
                    case "c2":
                        checkSupermarketOptions();
                        break;
                    case "c3":
                         if(fullBelly) {
                            mainTextArea.setText("You're still full... maybe some other time.");
                            choice3.setEnabled(false);
                         } else { foodHome(); }
                        break;
                    case "c4":
                        study();
                        break;
                }
                break;

            case "toMall":
                switch(userChoice) {
                    case "c1":
                        transportScore(position, choice1.getText());
                        break;
                    case "c2":
                        transportScore(position, choice2.getText());
                        break;
                    case "c3":
                        transportScore(position, choice3.getText());
                        break;
                    case "c4":
                        arriveHome();
                        break;
                }
                break;
```

```
        case "arriveMall":
            switch(userChoice) {
                case "c1":
                    eatMall();
                    break;
                case "c2":
                    clothesChoice();
                    break;
                case "c3":
                    phoneChoice();
                    break;
                case "c4":
                    toHome();
                    break;
            }
            break;

        case "foodMall":
            switch(userChoice) {
                case "c1":
                    foodScore(position, choice1.getText());
                    break;
                case "c2":
                    foodScore(position, choice2.getText());
                    break;
                case "c3":
                    foodScore(position, choice3.getText());
                    break;
                case "c4":
                    arriveMall();
                    break;
            }
            break;

        case "toSupermarket":
            switch(userChoice) {
                case "c1":
                    transportScore(position, choice1.getText());
                    break;
                case "c2":
                    transportScore(position, choice2.getText());
                    break;
                case "c3":
                    transportScore(position, choice3.getText());
                    break;
                case "c4":
                    arriveHome();
                    break;
            }
            break;

        case "arriveSupermarket":
            switch(userChoice) {
                case "c1":
                    if(boughtGrocery) {
                        mainTextArea.setText("You still have groceries at home!");
                        choice1.setEnabled(false);
```

```
                } else { buyTomato(); }
                break;
            case "c2":
                foodSupermarket();
                break;
            case "c3":
                toHome();
                break;
        }
        break;

    case "buyTomato":
        switch(userChoice) {
            case "c1":
                tomatoScore(choice1.getText());
                break;
            case "c2":
                tomatoScore(choice2.getText());
                break;
            case "c3":
                tomatoScore(choice3.getText());
                break;
            case "c4":
                arriveSupermarket();
                break;
        }
        break;

    case "buyCheese":
        switch(userChoice) {
            case "c1":
                cheeseScore(choice1.getText());
                break;
            case "c2":
                cheeseScore(choice2.getText());
                break;
            case "c3":
                cheeseScore(choice3.getText());
                break;
            case "c4":
                buyTomato();
                break;
        }
        break;

    case "buyMilk":
        switch(userChoice) {
            case "c1":
                milkScore(choice1.getText());
                break;
            case "c2":
                milkScore(choice2.getText());
                break;
            case "c3":
                milkScore(choice3.getText());
                break;
            case "c4":
```

```
                buyCheese();
                break;
            }
            break;

        case "buyMeat":
            switch(userChoice) {
                case "c1":
                    meatScore(choice1.getText());
                    break;
                case "c2":
                    meatScore(choice2.getText());
                    break;
                case "c3":
                    meatScore(choice3.getText());
                    break;
                case "c4":
                    buyMilk();
                    break;
            }
            break;

        case "buyBag":
            switch(userChoice) {
                case "c1":
                    bagScore(choice1.getText());
                    break;
                case "c2":
                    bagScore(choice2.getText());
                    break;
            }
            break;

        case "paidGrocery":
            switch(userChoice) {
                case "c1":
                    arriveSupermarket();
                    break;
            }
            break;

        case "foodSupermarket":
            switch(userChoice) {
                case "c1":
                    foodScore(position, choice1.getText());
                    break;
                case "c2":
                    foodScore(position, choice2.getText());
                    break;
                case "c3":
                    checkMallOptions(position);
                    break;
                case "c4":
                    arriveSupermarket();
                    break;
            }
            break;
```

```
            case "toHome":
                switch(userChoice) {
                    case "c1":
                        transportScore(position, choice1.getText());
                        break;
                    case "c2":
                        transportScore(position, choice2.getText());
                        break;
                    case "c3":
                        transportScore(position, choice3.getText());
                        break;
                    case "c4":
                        if(location.equals("mall")) {
                            arriveMall();
                        } else { arriveSupermarket(); }
                        break;
                }
                break;

            case "goodMealMall":
                switch(userChoice) {
                    case "c1":
                        arriveMall();
                }
                break;

            case "goodMealSupermarket":
                switch(userChoice) {
                    case "c1":
                        arriveSupermarket();
                        break;
                }
                break;

            case "clothesChoice":
                switch(userChoice) {
                    case "c1":
                        clothesScore(choice1.getText());
                        break;
                    case "c2":
                        clothesScore(choice2.getText());
                        break;
                    case "c3":
                        clothesScore(choice3.getText());
                        break;
                    case "c4":
                        arriveMall();
                        break;
                }
                break;

            case "choseClothes":
                switch(userChoice) {
                    case "c1":
                        arriveMall();
                        break;
```

```
            }
            break;

        case "phoneChoice":
            switch(userChoice) {
                case "c1":
                    phoneScore(choice1.getText());
                    break;
                case "c2":
                    phoneScore(choice2.getText());
                    break;
                case "c3":
                    phoneScore(choice3.getText());
                    break;
                case "c4":
                    arriveMall();
                    break;
            }
            break;

        case "chosePhone":
            switch(userChoice) {
                case "c1":
                    arriveMall();
                    break;
            }
            break;

        case "foodHome":
            switch(userChoice) {
                case "c1":
                    foodScore(position, choice1.getText());
                    break;
                case "c2":
                    foodScore(position, choice2.getText());
                    break;
                case "c3":
                    if(!boughtGrocery) {
                        mainTextArea.setText("Oh no, You've run out of milk! Maybe you\nshould go
buy some.");
                        choice3.setEnabled(false);
                    } else { foodScore(position, choice3.getText()); }
                    break;
                case "c4":
                    arriveHome();
                    break;
            }
            break;

        case "goodMealHome":
            switch(userChoice) {
                case "c1":
                    arriveHome();
                    break;
            }
            break;
```

```
        case "study":
            switch(userChoice) {
                case "c1":
                    foodPage1();
                    break;
                case "c2":
                    transportPage();
                    break;
                case "c3":
                    shoppingPage1();
                    break;
                case "c4":
                    arriveHome();
                    break;
            }
            break;

        case "foodPage1":
            switch(userChoice) {
                case "c1":
                    foodPage2();
                    break;
            }
            break;

        case "foodPage2":
            switch(userChoice) {
                case "c1":
                    foodPage3();
                    break;
                case "c2":
                    foodPage1();
                    break;
            }
            break;

        case "foodPage3":
            switch(userChoice) {
                case "c1":
                    foodPage4();
                    break;
                case "c2":
                    foodPage2();
                    break;
            }
            break;

        case "foodPage4":
            switch(userChoice) {
                case "c1":
                    study();
                    timePassed(3);
                    break;
                case "c2":
                    foodPage3();
            }
            break;
```

```
            case "transportPage":
                switch(userChoice) {
                    case "c1":
                        study();
                        timePassed(2);
                        break;
                }
                break;

            case "shoppingPage1":
                switch(userChoice) {
                    case "c1":
                        shoppingPage2();
                        break;
                }
                break;

            case "shoppingPage2":
                switch(userChoice) {
                    case "c1":
                        shoppingPage3();
                        break;
                    case "c2":
                        shoppingPage1();
                        break;
                }
                break;

            case "shoppingPage3":
                switch(userChoice) {
                    case "c1":
                        study();
                        timePassed(3);
                        break;
                    case "c2":
                        shoppingPage2();
                        break;
                }
                break;
        }
        updateScore();
        showGameOverScreen();
    }

    // Enables/disables button depending on boolean variable
    public static void resetButton(boolean check, JButton buttonChoice) {
        if(check) {
            buttonChoice.setEnabled(false);
        } else { buttonChoice.setEnabled(true); }
    }

    // Enables all buttons
    public static void enableAllButtons() {
        choice1.setEnabled(true);
        choice2.setEnabled(true);
        choice3.setEnabled(true);
```

```
        choice4.setEnabled(true);
    }
}
```

7.14 Game_Food Class

```
package com.mycompany.ClimateEducationApp;

import static com.mycompany.ClimateEducationApp.Game.*;
import static com.mycompany.ClimateEducationApp.Game_ChoiceHandler.*;

public class Game_Food {
    // Displays screen after a meal
    public static void goodMeal(String place) {
        enableAllButtons();
        fullBelly = true;
        position = "goodMeal" + place;
        mainTextArea.setText("Ahhh... that was a delicious meal. Let's see\nwhat else we can do.");

        choice1.setText(">");
        choice2.setText("");
        choice3.setText("");
        choice4.setText("");

        // Modifies counter variables depending on location
        switch(place) {
            case "Mall":
                mallScreenCounter++;
                mallChoiceCounter++;
                ateAtMall = true;
                break;
            case "Supermarket":
                supermarketScreenCounter++;
                supermarketChoiceCounter++;
                ateAtSupermarket = true;
                break;
            case "Home":
                ateAtHome = true;
        }
    }

    // Displays screen for food choices at mall
    public static void foodMall() {
        enableAllButtons();
        position = "foodMall";
        mainTextArea.setText("You're starting to feel hungry... it's already time\nfor " + mealTime + ".
Where do you feel like eating?");

        choice1.setText("Domino's Pizza");
        choice2.setText("Chipotle");
        choice3.setText("McDonald's");
        choice4.setText("<");
    }

    // Displays screen for food choices at supermarket
    public static void foodSupermarket() {
        enableAllButtons();
```

```
        position = "foodSupermarket";
        mainTextArea.setText("You're starting to feel hungry... it's already time\nfor " + mealTime + ".
Where do you feel like eating?");

        choice1.setText("Local Cafe nearby");
        choice2.setText("Foreign restaurant");
        choice3.setText("Go to the mall");
        choice4.setText("<");
    }

    // Displays screen for food choices at home
    public static void foodHome() {
        enableAllButtons();
        position = "foodHome";
        mainTextArea.setText("What would you like to cook?");

        choice1.setText("Steak");
        choice2.setText("Vegetable Stir-Fry");
        choice3.setText("Cereal");
        choice4.setText("<");
    }
}
```

7.15 Game_Home Class

```
package com.mycompany.ClimateEducationApp;

import static com.mycompany.ClimateEducationApp.Game.*;
import static com.mycompany.ClimateEducationApp.Game_Transport.*;
import static com.mycompany.ClimateEducationApp.Game_ChoiceHandler.enableAllButtons;
import static com.mycompany.ClimateEducationApp.Game_ChoiceHandler.resetButton;

public class Game_Home {
    // Displays main Game_Home screen
    public static void arriveHome() {
        // Resets variable values
        enableAllButtons();
        location = "home";
        tookCar = true;
        tookPublic = true;
        tookBike = true;
        mallScreenCounter = 0;
        supermarketScreenCounter = 0;
        mallChoiceCounter = 0;
        supermarketChoiceCounter = 0;
        if(!fullBelly) {
            resetButton(ateAtHome, choice3);
        } else { choice3.setEnabled(false); }

        position = "arriveHome";
        if(!goneOut) {
            mainTextArea.setText("""
                                 It's a brand new day! You have the day off so
                                 you're feeling energetic. What do you feel like
                                 doing today?""");
        } else {
            mainTextArea.setText("Welcome back home. What else can we do?");
        }
```

```
        choice1.setText("Head to the mall");
        choice2.setText("Go to supermarket");
        choice3.setText("Make " + mealTime);
        choice4.setText("Study climate change");
    }

    // Checks if mall has available choices before displaying mall screen
    public static void checkMallOptions(String position) {
        if(shoppedClothes && shoppedPhone && ateAtMall && fullBelly) {
            if(position.equals("arriveHome")) {
                choice1.setEnabled(false);
            } else { choice3.setEnabled(false); }
            mainTextArea.setText("""
                                 There's nothing left to do at the mall. Check
                                 again later.""");
        } else { toMall(); }
    }

    // Checks if supermarket has available choices before displaying supermarket screen
    public static void checkSupermarketOptions() {
        if(supermarketChoiceCounter >= 2 && fullBelly) {
            choice2.setEnabled(false);
            mainTextArea.setText("""
                                 All done with the groceries. Not feeling very
                                 hungry either... maybe some other time.""");
        } else { toSupermarket(); }
    }
}
```

7.16 Game_Mall Class

```
package com.mycompany.ClimateEducationApp;

import static com.mycompany.ClimateEducationApp.Game.*;
import static com.mycompany.ClimateEducationApp.Game_Food.foodMall;
import static com.mycompany.ClimateEducationApp.Game_ChoiceHandler.*;

public class Game_Mall {
    // Displays main Game_Mall screen
    public static void arriveMall() {
        goneOut = true;
        location = "mall";
        position = "arriveMall";

        if(!fullBelly) {
            resetButton(ateAtMall, choice1);
        } else { choice1.setEnabled(false); }
        resetButton(shoppedClothes, choice2);
        resetButton(shoppedPhone, choice3);

        if(mallChoiceCounter > 0) {
            mainTextArea.setText("What else can we do?");
        } else {
            mainTextArea.setText("You've arrived at the mall.\nYour first action is to...");
        }

        choice1.setText("Eat " + mealTime);
```

```
        choice2.setText("Shop for clothes");
        choice3.setText("Shop for phone");
        choice4.setText("Go home");
    }

    // Checks if player is hungry before showing food choices
    public static void eatMall() {
        if(fullBelly) {
            mainTextArea.setText("You already ate " + mealTime + "! You're still full.");
            choice1.setEnabled(false);
        } else {
            if(!ateAtMall) { choice1.setEnabled(true); }
            foodMall();
        }
    }

    // Checks if player has shopped clothes before showing clothing choices
    public static void clothesChoice() {
        if(shoppedClothes) {
            mainTextArea.setText("Bought too much. Wouldn't want to spend all your allowance in a
day, right?");
            choice2.setEnabled(false);
        } else {
            enableAllButtons();
            position = "clothesChoice";
            mainTextArea.setText("You got the budget, just not the style. Which\nbrand will it be?");

            choice1.setText("H&M");
            choice2.setText("Nike");
            choice3.setText("Prada");
            choice4.setText("<");
        }
    }

    // Displays screen after player has shopped clothes
    public static void choseClothes() {
        enableAllButtons();
        mallScreenCounter++;
        shoppedClothes = true;

        position = "choseClothes";
        mainTextArea.setText("That'll cover the new few months of your\nfashion... let's see what
else we can do.");

        choice1.setText(">");
        choice2.setText("");
        choice3.setText("");
        choice4.setText("");
    }

    // Checks if player has shopped phone before showing phone choices
    public static void phoneChoice() {
        if(shoppedPhone) {
            mainTextArea.setText("You've already bought a phone! How many do you need?!");
            choice3.setEnabled(false);
        } else {
            enableAllButtons();
```

```
            position = "phoneChoice";
            mainTextArea.setText("Your current phone is already 10 years old...\ntime to buy a new
one. iPhone sounds good, but which model?");

            choice1.setText("iPhone X");
            choice2.setText("iPhone 13 PRO MAX");
            choice3.setText("iPhone 15");
            choice4.setText("<");
        }
    }

    // Displays screen after player has shopped phone
    public static void chosePhone() {
        enableAllButtons();
        mallScreenCounter++;
        shoppedPhone = true;

        position = "chosePhone";
        mainTextArea.setText("That was a bit of a hefty price... let's see what\nelse we can do.");

        choice1.setText(">");
        choice2.setText("");
        choice3.setText("");
        choice4.setText("");
    }
}
```

7.17 Game_Score Class

```
package com.mycompany.ClimateEducationApp;

import static com.mycompany.ClimateEducationApp.Game.*;
import static com.mycompany.ClimateEducationApp.Game_Mall.*;
import static com.mycompany.ClimateEducationApp.Game_Supermarket.*;
import static com.mycompany.ClimateEducationApp.Game_Home.arriveHome;
import static com.mycompany.ClimateEducationApp.Game_Food.goodMeal;
import static com.mycompany.ClimateEducationApp.Game_Time.timePassed;

public class Game_Score {
    // Declare variables for grocery scoring
    public static int tomato;
    public static int cheese;
    public static int milk;
    public static int meat;
    public static int bag;

    // Updates header data in game
    public static void updateScore() {
        scoreLabelNumber.setText("" + environmentalScore);
    }

    // Updates score from transport choice
    public static void transportScore(String position, String transport) {
        timePassed(1);
        switch(transport) {
            case "Self-drive":
                tookCar = true;
                tookPublic = false;
```

```
                tookBike = false;
                environmentalScore += 1;
                break;
            case "Public Transport":
                tookCar = false;
                tookPublic = true;
                tookBike = false;
                environmentalScore += 5;
                break;
            case "Bicycle":
                tookCar = false;
                tookPublic = false;
                tookBike = true;
                environmentalScore += 9;
                break;
        }

        // Displays screen depending on location
        switch(position) {
            case "toMall":
                arriveMall();
                break;
            case "toSupermarket":
                arriveSupermarket();
                break;
            case "toHome":
                arriveHome();
                break;
        }
    }

    // Updates score from food choice
    public static void foodScore(String position, String food) {
        timePassed(1);
        // Different scores for different locations
        switch(position) {
            case "foodMall":
                switch(food) {
                    case "Domino's Pizza":
                        environmentalScore += 1;
                        break;
                    case "Chipotle":
                        environmentalScore += 9;
                        break;
                    case "McDonald's":
                        environmentalScore += 5;
                        break;
                }
                goodMeal("Mall");
                break;

            case "foodSupermarket":
                switch(food) {
                    case "Local Cafe nearby":
                        environmentalScore += 8;
                        break;
                    case "Foreign restaurant":
```

```
                environmentalScore += 2;
                break;
            }
            goodMeal("Supermarket");
            break;

        case "foodHome":
            switch(food) {
                case "Steak":
                environmentalScore += 1;
                break;
                case "Vegetable Stir-Fry":
                environmentalScore += 7;
                break;
                case "Cereal":
                environmentalScore += 7;
                break;
            }
            goodMeal("Home");
            break;
    }
}

// Updates score from clothes choice
public static void clothesScore(String brand) {
    timePassed(2);
    switch(brand) {
        case "H&M":
            environmentalScore += 5;
            break;
        case "Nike":
            environmentalScore += 9;
            break;
        case "Prada":
            environmentalScore += 1;
            break;
    }
    mallChoiceCounter++;
    choseClothes();
}

// Updates score from phone choice
public static void phoneScore(String model) {
    timePassed(2);
    switch(model) {
        case "iPhone 10":
            environmentalScore += 9;
            break;
        case "iPhone 13 PRO MAX":
            environmentalScore += 5;
            break;
        case "iPhone 15":
            environmentalScore += 1;
            break;
    }
    mallChoiceCounter++;
    chosePhone();
```

```
    }

    // Updates score from tomato choice
    public static void tomatoScore(String tomatoChoice) {
        switch(tomatoChoice) {
            case "No. Use leftovers":
                tomato = 9;
                break;
            case "Buy local produce":
                tomato = 5;
                break;
            case "Buy with packaging":
                tomato = 1;
                break;
        }
        buyCheese();
    }

    // Updates score from cheese choice
    public static void cheeseScore(String cheeseChoice) {
        switch(cheeseChoice) {
            case "Hard Cheese":
                cheese = 1;
                break;
            case "Soft Cheese":
                cheese = 5;
                break;
            case "Vegan Cheese":
                cheese = 9;
                break;
        }
        buyMilk();
    }

    // Updates score from milk choice
    public static void milkScore(String milkChoice) {
        switch(milkChoice) {
            case "Hazelnut Milk":
                milk = 5;
                break;
            case "Soy Milk":
                milk = 9;
                break;
            case "Dairy Milk":
                milk = 1;
                break;
        }
        buyMeat();
    }

    // Updates score from meat score
    public static void meatScore(String meatChoice) {
        switch(meatChoice) {
            case "Lamb":
                meat = 1;
                break;
            case "Chicken":
```

```
                meat = 9;
                break;
            case "Beef":
                meat = 5;
                break;
        }
        buyBag();
    }

    // Updates score from grocery bag choice
    public static void bagScore(String bagChoice) {
        switch(bagChoice) {
            case "Plastic":
                bag = 1;
                break;
            case "Paper":
                bag = 10;
                break;
        }
        paidGrocery();
    }

    // Calculates total score from grocery choices
    public static void totalGroceryScore() {
        timePassed(3);
        boughtGrocery = true;
        supermarketScreenCounter++;
        supermarketChoiceCounter++;
        environmentalScore += (tomato + cheese + milk + meat + bag);
    }
}
```

7.18 Game_Study Class

```
package com.mycompany.ClimateEducationApp;

import static com.mycompany.ClimateEducationApp.Game.choice1;
import static com.mycompany.ClimateEducationApp.Game.choice2;
import static com.mycompany.ClimateEducationApp.Game.choice3;
import static com.mycompany.ClimateEducationApp.Game.choice4;
import static com.mycompany.ClimateEducationApp.Game.mainTextArea;
import static com.mycompany.ClimateEducationApp.Game.position;
import static com.mycompany.ClimateEducationApp.Game_ChoiceHandler.enableAllButtons;

public class Game_Study {
    // Displays screen for studying
    public static void study() {
        enableAllButtons();
        position = "study";
        mainTextArea.setText("Which area of Climate Change would you like\nto study about?");

        choice1.setText("Food");
        choice2.setText("Transport");
        choice3.setText("Shopping");
        choice4.setText("Go back");
    }

    // Displays page 1 of food study topic
```

```
public static void foodPage1() {
    enableAllButtons();
    position = "foodPage1";
    mainTextArea.setText("""
                    Regarding meat, lamb produces an average of
                    20.44kg of C02 emissions/kg of product. Beef,
                    5kg fewer C02 emissions/kg than lamb.
                    Chicken however comes out on top, producing
                    only 2.33 kg of C02/kg.""");

    choice1.setText(">");
    choice2.setText("");
    choice3.setText("");
    choice4.setText("");
}

// Displays page 2 of food study topic
public static void foodPage2() {
    position = "foodPage2";
    mainTextArea.setText("""
                    Dairy requires 9x more land than any
                    plant-based milk. Hazelnut milk thrives on rain
                    and sequester carbon, creating a net benefit for
                    the climate. Oat milk and soy milk is tied for
                    most environment friendly milk!""");

    choice1.setText(">");
    choice2.setText("<");
    choice3.setText("");
    choice4.setText("");
}

// Displays page 3 of food study topic
public static void foodPage3() {
    position = "foodPage3";
    mainTextArea.setText("""
                    The carbon footprint of cheese is almost double
                    that of chicken. Soft cheese and plant-based
                    alternatives generally have a smaller footprint.
                    Hard cheese usually requires more milk than
                    soft, which means higher emissions associated
                    with livestock and farming.""");

    choice1.setText(">");
    choice2.setText("<");
    choice3.setText("");
    choice4.setText("");
}

// Displays page 4 of food study topic
public static void foodPage4() {
    position = "foodPage4";
    mainTextArea.setText("""
                    When shopping for tomatoes, consider buying
                    locally, as demand for local produce reduces
                    food miles and lessens greenhouse gas
                    emission. Packaging-free produce is also a
```

```
                    good alternative.""");

    choice1.setText("Read another topic");
    choice2.setText("<");
    choice3.setText("");
    choice4.setText("");
}

// Displays transport study topic page
public static void transportPage() {
    enableAllButtons();
    position = "transportPage";
    mainTextArea.setText("""
                    Transportation takes up to 1/5th of global
                    greenhouse gas emissions. Public transportation
                    is advised due to simple efficiency:
                    buses and trains can carry substantially more
                    than a single car.""");

    choice1.setText("Read another topic");
    choice2.setText("");
    choice3.setText("");
    choice4.setText("");
}

// Displays page 1 of shopping study topic
public static void shoppingPage1() {
    enableAllButtons();
    position = "shoppingPage1";
    mainTextArea.setText("""
                    Some businesses adopt sustainable practices.
                    For clothing brands, the ones leading the
                    fray are Nike, Adidas and Levi's, followed by
                    H&M as the middle tier. Puma and Uniqlo rank
                    the worst of all.""");

    choice1.setText(">");
    choice2.setText("");
    choice3.setText("");
    choice4.setText("");
}

// Displays page 2 of shopping study topic
public static void shoppingPage2() {
    position = "shoppingPage2";
    mainTextArea.setText("""
                    Even buying a new phone contributes to climate
                    change. Frequent changes and upgrades drives
                    the purchase of newer models and vice versa.
                    The issue lies not with the technology, but the
                    manufacturing process.""");

    choice1.setText(">");
    choice2.setText("<");
    choice3.setText("");
    choice4.setText("");
}
```

```
    // Displays page 3 of shopping study topic
    public static void shoppingPage3() {
        position = "shoppingPage3";
        mainTextArea.setText("""
                        Regarding fast food, most of Taco Bell's are
                        recycleable, goal of all by 2025. Chipotle
                        tracks the footprint for each digital order.
                        McDonald's have taken steps to find a
                        renewable solution for their waste.""");

        choice1.setText("Read another topic");
        choice2.setText("<");
        choice3.setText("");
        choice4.setText("");
    }
}
```

7.19 Game_Supermarket Class

```
package com.mycompany.ClimateEducationApp;

import static com.mycompany.ClimateEducationApp.Game_ChoiceHandler.*;
import static com.mycompany.ClimateEducationApp.Game.*;
import static com.mycompany.ClimateEducationApp.Game_Score.totalGroceryScore;

public class Game_Supermarket {
    // Displays main Game_Supermarket screen
    public static void arriveSupermarket() {
        goneOut = true;
        location = "supermarket";

        resetButton(boughtGrocery, choice1);
        if(!fullBelly) {
            resetButton(ateAtSupermarket, choice2);
        } else {
            choice2.setEnabled(false);
        }

        position = "arriveSupermarket";
        if(supermarketChoiceCounter > 0) {
            mainTextArea.setText("Let's take a look around somewhere else.");
        } else {
            mainTextArea.setText("Giant is always a reliable and cheap\nsupermarket. This time you
came here to...");
        }

        choice1.setText("Buy groceries");
        choice2.setText("Eat " + mealTime);
        choice3.setText("Go home");
        choice4.setText("");
    }

    // Displays Tomato choice screen
    public static void buyTomato() {
        enableAllButtons();
        position = "buyTomato";
        mainTextArea.setText("""
```

```
                Let's see... first thing's first are tomatoes. You
                recall still having some at home... should you
                still buy fresh ones?""");

    choice1.setText("No. Use leftovers");
    choice2.setText("Buy local produce");
    choice3.setText("Buy with packaging");
    choice4.setText("<");
}

// Displays Cheese choice screen
public static void buyCheese() {
    position = "buyCheese";
    mainTextArea.setText("""
                Jerry, your roommate, asked you to buy some
                cheese. He doesn't care what type, but you
                usually prefer hard cheese.""");

    choice1.setText("Hard Cheese");
    choice2.setText("Soft Cheese");
    choice3.setText("Vegan Cheese");
    choice4.setText("<");
}

// Displays Milk choice screen
public static void buyMilk() {
    position = "buyMilk";
    mainTextArea.setText("""
                Oh yeah... you've run out of fresh milk back
                at home. Better choose a good one if you
                want to eat cereal.""");

    choice1.setText("Hazelnut Milk");
    choice2.setText("Soy Milk");
    choice3.setText("Dairy Milk");
    choice4.setText("<");
}

// Displays Meat choice screen
public static void buyMeat() {
    position = "buyMeat";
    mainTextArea.setText("Last one on the list... meat!");

    choice1.setText("Lamb");
    choice2.setText("Chicken");
    choice3.setText("Beef");
    choice4.setText("<");
}

// Displays Bag choice screen
public static void buyBag() {
    position = "buyBag";
    mainTextArea.setText("""
                You head to the counter to pay for your
                groceries. The cashier asks if you prefer
                a paper bag or a plastic one. You reply...""");
```

```
        choice1.setText("Plastic");
        choice2.setText("Paper");
        choice3.setText("<");
        choice4.setText("");
    }

    // Displays screen after player bought grocery
    public static void paidGrocery() {
        totalGroceryScore();
        boughtGrocery = true;

        position = "paidGrocery";
        mainTextArea.setText("Total is around... RM79. You should head back\nquick before the
grocery gets bad.");

        choice1.setText(">");
        choice2.setText("");
        choice3.setText("");
        choice4.setText("");
    }
}
```

7.20 Game_Time Class

```
package com.mycompany.ClimateEducationApp;

import static com.mycompany.ClimateEducationApp.Game.ateAtHome;
import static com.mycompany.ClimateEducationApp.Game.ateAtMall;
import static com.mycompany.ClimateEducationApp.Game.ateAtSupermarket;
import static com.mycompany.ClimateEducationApp.Game.fullBelly;
import static com.mycompany.ClimateEducationApp.Game.mealTime;
import static com.mycompany.ClimateEducationApp.Game.timeCounter;
import static com.mycompany.ClimateEducationApp.Game.timeLabelNumber;

public class Game_Time {
    // Method for incrementing timeCounter depending on parameters
    public static void timePassed(int hour) {
        timeCounter += hour;

        // Updates time value in Game header
        if(timeCounter == 12) {
            timeLabelNumber.setText(timeCounter + " PM");
            timeCounter = 0;
        } else if(timeCounter > 12 && timeLabelNumber.getText().contains("AM")) {
            timeCounter -= 12;
            timeLabelNumber.setText(timeCounter + " PM");
        } else if(timeLabelNumber.getText().contains("AM")) {
            timeLabelNumber.setText(timeCounter + " AM");
        } else {
            timeLabelNumber.setText(timeCounter + " PM");
        }

        // Resets player hunger status if lunch or dinner time
        if(timeCounter > 5 && timeLabelNumber.getText().contains("PM")) {
            ateAtHome = false;
            ateAtMall = false;
            ateAtSupermarket = false;
            fullBelly = false;
```

```
            mealTime = "dinner";
        } else if((timeCounter < 6 && timeLabelNumber.getText().contains("PM")) ||
timeLabelNumber.getText().contains("12 PM")) {
            ateAtHome = false;
            ateAtMall = false;
            ateAtSupermarket = false;
            fullBelly = false;
            mealTime = "lunch";
        }
    }
}
```

7.21 Game_TitleScreenHandler Class

```
package com.mycompany.ClimateEducationApp;

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

public class Game_TitleScreenHandler implements ActionListener {
    //
    public void actionPerformed(ActionEvent event) {
        // Initializes userChoice with button User chooses
        String userChoice = event.getActionCommand();

        // Performs action depending on User's button choice
        switch(userChoice) {
            case "start":
                Game.createGameScreen();
                break;
            case "exit":
                exitButtonActionPerformed(event);
                break;
            case "restart":
                Game.window.dispose();
                new Game();
                break;
        }
    }

    // Closes game and opens Home page
    public static void exitButtonActionPerformed(java.awt.event.ActionEvent evt) {
        UserInterface UI = new UserInterface();
        UI.setVisible(true);            // Sets UI visibility to true
        UI.pack();                      // Packs UI frame
        UI.setLocationRelativeTo(null); // Centers UI on screen
        // Closes Game window
        Game.window.dispose();
    }
}
```

7.22 Game_Transport

```
package com.mycompany.ClimateEducationApp;

import static com.mycompany.ClimateEducationApp.Game.*;
import static com.mycompany.ClimateEducationApp.Game_ChoiceHandler.*;

public class Game_Transport {
```

```
    // Displays Game_Transport Choice screen
    public static void setTransportChoice() {
        enableAllButtons();
        resetButton(!tookCar, choice1);
        resetButton(!tookPublic, choice2);
        resetButton(!tookBike, choice3);
        mainTextArea.setText("How are you going there?");

        choice1.setText("Self-drive");
        choice2.setText("Public Transport");
        choice3.setText("Bicycle");
        choice4.setText("<");
    }

    // Sets position for Mall destination
    public static void toMall() {
        position = "toMall";
        setTransportChoice();
    }

    // Sets position for Supermarket destination
    public static void toSupermarket() {
        position = "toSupermarket";
        setTransportChoice();
    }

    // Sets position for Home destination
    public static void toHome() {
        // Checks if user has just arrived before going back home
        if(position.equals("arriveMall" ) && mallChoiceCounter == 0) {
                mainTextArea.setText("You just got here! It's way too early to go home\nright now. Go
do something else.");
                choice4.setEnabled(false);
        } else if(position.equals("arriveSupermarket") && supermarketChoiceCounter == 0) {
                mainTextArea.setText("You just got here! It's way too early to go home\nright now. Go
do something else.");
                choice3.setEnabled(false);
        } else {
            position = "toHome";
            setTransportChoice();
        }
    }
}
```

7.23 HouseFootprint Class

```
package com.mycompany.ClimateEducationApp;

public class HouseFootprint {
    // Declare variables for house footprint
    double electricBill;
    double gasBill;
    double oilBill;

    // Initializes variables
    public HouseFootprint(double electricBill, double gasBill, double oilBill) {
        this.electricBill = electricBill;
        this.gasBill = gasBill;
```

```
        this.oilBill = oilBill;
    }

    // Calculate carbon footprint produced by house
    public double calculateFootprint() {
        return (electricBill * 105) + (gasBill * 105) + (oilBill * 113);
    }
}
```

7.24 LoginPage Class

```
package com.mycompany.ClimateEducationApp;

import javax.swing.JOptionPane;
import java.io.IOException;
import java.util.ArrayList;
import java.io.BufferedReader;
import java.io.FileReader;

public class LoginPage extends javax.swing.JFrame {
    private ArrayList<User> users;

    // Creates new form LoginPage1
    public LoginPage() {
        initComponents();
        this.users = new ArrayList<>();
        // Load existing users from file (if any)
        loadUsersFromFile(users);
        // Adds default User account
        User defaultUser = new User("Yoo Joonghyuk", "yjh@gmail.com", "sunfish");
        users.add(defaultUser);
    }

    // Authenticates login credentials
    private boolean validateUser(String email, String password) {
        for(User user : users) {
            System.out.println("Checking user: " + user.getEmail());
            if (user.getEmail().equals(email) && user.getPassword().equals(password)) {
                return true; // User found with matching credentials
            }
        }
        return false; // User not found or invalid credentials
    }

    // Login button method
    private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String email = emailTextField.getText();
        String password = new String(passwordField.getPassword());

        if (validateUser(email, password)) {
            JOptionPane.showMessageDialog(this, "Login Successful!");
            UserInterface UI = new UserInterface();
            UI.setVisible(true);
            UI.pack();
            UI.setLocationRelativeTo(null);

            this.dispose(); // Close the login form after successful login
        } else {
```

```
            JOptionPane.showMessageDialog(this, "Invalid username or password!");
        }
    }

    // Loads user login data from userDetails.txt file
    private static void loadUsersFromFile(ArrayList<User> users) {
        String filePath = "src/com/mycompany/ClimateEducationApp/userDetails.txt";

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            String line = reader.readLine();
            do {
                String[] userData = line.split(",");
                if (userData.length == 3) {
                    User user = new User(userData[0], userData[1], userData[2]);
                    users.add(user);
                }
            } while ((line = reader.readLine()) != null);
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
    }

    // Sign Up button method
    private void SignUpButtonActionPerformed(java.awt.event.ActionEvent evt) {
        SignUp SignUpFrame = new SignUp();
        SignUpFrame.setVisible(true);            // Sets visibility of SignUpFrame
        SignUpFrame.pack();                      // Packs the frame
        SignUpFrame.setLocationRelativeTo(null);  // Centers frame on screen
        // Closes LoginFrame
        this.dispose();
    }

    // Opens Admin Login page
    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
        AdminLogin AdminLoginFrame = new AdminLogin();
        AdminLoginFrame.setVisible(true);
        AdminLoginFrame.pack();
        AdminLoginFrame.setLocationRelativeTo(null);
        //closes LoginFrame
        this.dispose();
    }

    // Toggles password text visibility
    private void showPwordActionPerformed(java.awt.event.ActionEvent evt) {
        if (showPword.isSelected()) {//show entered password
            passwordField.setEchoChar((char) 0);
        } else {//hide entered password
            passwordField.setEchoChar('*');
        }
    }

    /**
     * @param args the command line arguments
     */

    // Runs start of ClimateEducationApp
```

```
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(LoginPage.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(LoginPage.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(LoginPage.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(LoginPage.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                // Open Login form
                LoginPage loginPage = new LoginPage();
                loginPage.setVisible(true);
                loginPage.setLocationRelativeTo(null);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton LoginButton;
    private javax.swing.JPanel Right;
    private javax.swing.JButton SignUpButton;
    private javax.swing.JTextField emailTextField;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton7;
    private javax.swing.JCheckBox jCheckBox1;
    private javax.swing.JLabel jLabel1;
```

```
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPasswordField jPasswordField1;
    private javax.swing.JPasswordField passwordField;
    private java.awt.PopupMenu popupMenu1;
    private javax.swing.JCheckBox showPword;
    // End of variables declaration
}
```

7.25 Recycle Class

```
package com.mycompany.ClimateEducationApp;

public class Recycle {
    // Declares variables for recycling footprint
    String recycleNewspaper;
    double totalNewspaperFootprint;
    String recycleAluminium;
    double totalAluminiumFootprint;

    // Constructor
    public Recycle(String recycleNewspaper, String recycleAluminium) {
        this.recycleNewspaper = recycleNewspaper;
        this.recycleAluminium = recycleAluminium;
    }

    // Calculates newspaper footprint
    public double calculateNewspaper() {
        if (recycleNewspaper.equalsIgnoreCase("Yes")) {
        } else if (recycleNewspaper.equalsIgnoreCase("No")) {
            totalNewspaperFootprint = 184;
        } else {
            System.out.println("Please enter a valid answer");
        }
        return totalNewspaperFootprint;
    }

    // Calculates aluminium footprint
    public double calculateAluminium() {
        if (recycleAluminium.equalsIgnoreCase("Yes")) {
        } else if (recycleAluminium.equalsIgnoreCase("No")) {
            totalAluminiumFootprint = 166;
        } else {
            System.out.println("Please enter a valid answer");
        }
        return totalAluminiumFootprint;
    }
}
```

7.26 SignUp Class

```
package com.mycompany.ClimateEducationApp;

import javax.swing.JOptionPane;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.regex.*;

public class SignUp extends javax.swing.JFrame {
    private ArrayList<User> users;

    // Creates new SignUp form
    public SignUp() {
        initComponents();
        this.users = new ArrayList<>();
    }

     // Toggles password text visibility
    private void showPwordActionPerformed(java.awt.event.ActionEvent evt) {
        if (showPword.isSelected()) {
            pword.setEchoChar((char) 0);
            confirmPword.setEchoChar((char) 0);
        } else {
            pword.setEchoChar('*');
            confirmPword.setEchoChar('*');
        }
    }

    // Sign up button method
    private void signUpActionPerformed(java.awt.event.ActionEvent evt) {
        String username = userFullName.getText();
        String password = new String(pword.getPassword());
        String conPassword = new String(confirmPword.getPassword());
        String email = userEmail.getText();
        String regex =
"^[\\w!#$%&'+/=?`{|}~^-]+(?:\\.[\\w!#$%&'+/=?`{|}~^-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}";
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(email);

        if(username.trim().isEmpty() || email.trim().isEmpty() || password.trim().isEmpty() ||
conPassword .trim().isEmpty()) {
            // Displays message if any fields are empty
            JOptionPane.showMessageDialog(this, "Please Ensure No Field Is Empty");
        } else if(!matcher.matches()) {
            // Displays message if email is invalid
            JOptionPane.showMessageDialog(this, "Please Enter A Valid Email");
        } else if(!conPassword.equals(password)) {
            // Displays message if password confirmation fails
            JOptionPane.showMessageDialog(this, "Your Password Does Not Match!");
        } else {
            // Signs up user
            User newUser = new User(username, password, email);
            users.add(newUser);
            saveUsersToFile();
            UserInterface UI = new UserInterface();
```

```
            UI.setVisible(true);
            UI.pack();
            UI.setLocationRelativeTo(null);

            this.dispose();
        }
    }

    // Writes new user signup data to userDetails.txt file
    private void saveUsersToFile() {
        String filePath = "src/com/mycompany/ClimateEducationApp/userDetails.txt"; // Adjust as
needed

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath, true))) {
            for (User user : users) {
                writer.write(user.getUsername() +"," + user.getPassword() + "," + user.getEmail() +
"\n");
             System.out.println(user.getUsername() + "," + user.getPassword() + "," + user.getEmail()
+ "\n");
            }
            JOptionPane.showMessageDialog(this, "Registration Successful!");
        } catch (IOException e) {
            System.err.println("Error writing to file: " + e.getMessage());
            JOptionPane.showMessageDialog(this, "Error occurred while saving user data!");
        }
    }

    // Shows user login page
    private void loginActionPerformed(java.awt.event.ActionEvent evt) {
        LoginPage LoginFrame = new LoginPage();
        LoginFrame.setVisible(true);
        LoginFrame.pack();
        LoginFrame.setLocationRelativeTo(null);
        //Closes SignUpFrame
        this.dispose();
    }

    // Variables declaration - do not modify
    private javax.swing.JPasswordField confirmPword;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JButton login;
    private javax.swing.JPasswordField pword;
    private javax.swing.JCheckBox showPword;
    private javax.swing.JButton signUp;
```

```
    private javax.swing.JTextField userEmail;
    private javax.swing.JTextField userFullName;
    // End of variables declaration
}
```

7.27 User Class

```
package com.mycompany.ClimateEducationApp;

public class User {
    // Declaring User class variables
    private String username;
    private String email;
    private String password;

    // Intializing variables
    public User(String username,  String email,String password) {
        this.username = username;
        this.password = password;
        this.email = email;
    }

    // username getter method
    public String getUsername() {
        return username;
    }

    // password getter method
    public String getPassword() {
        return password;
    }

    // email getter method
    public String getEmail() {
        return email;
    }
}
```

7.28 UserInterface Class

```
package com.mycompany.ClimateEducationApp;

public class UserInterface extends javax.swing.JFrame {
    // Creates new UserInterface form
    public UserInterface() {
        initComponents();
        CItext.setEditable(false);
    }

    public void setCItextHTML(String htmlContent) {
        CItext.setContentType("text/html");
        CItext.setText(htmlContent);
    }

    // Opens back Login page
    private void exitButtonActionPerformed(java.awt.event.ActionEvent evt) {
        LoginPage LoginFrame = new LoginPage();
        LoginFrame.setVisible(true);           // Sets visibility of LoginFrame to true
        LoginFrame.pack();                     // Packs the frame
```

```
        LoginFrame.setLocationRelativeTo(null); // Centers frame on the screen

        this.dispose();
    }

    // Opens Early Warning System page
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        EA_User EAUserUI = new EA_User();
        EAUserUI.setVisible(true);
        EAUserUI.pack();
        EAUserUI.setLocationRelativeTo(null);

        this.dispose();
    }

    // Opens Carbon Footprint Calculator page
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        CF_UserUI UI = new CF_UserUI();
        UI.setVisible(true);
        UI.pack();
        UI.setLocationRelativeTo(null);

        this.dispose();
    }

    // Opens Climate Information page
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        CI_User UI = new CI_User();
        UI.setVisible(true);
        UI.pack();
        UI.setLocationRelativeTo(null);

        this.dispose();
    }

    // Launches Educational Game
    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        Game game = new Game();
        game.window.setVisible(true);
        game.window.setLocationRelativeTo(null);

        this.dispose();
    }

    // Sets Warning dashboard data
    public void setThreeLinesOfText(String line1, String line2, String line3) {
        WarningDashboard.setText("");
        WarningDashboard.setText("Location: " + line1 + "\n" + "Danger: " + line2 + "\n" + "Warning:
" +line3);
    }

    // Sets Warning dashboard data
     public void setValueToDisplay(String value) {
        WarningDashboard.setText(value);
    }

    // Variables declaration - do not modify
```

```
    private javax.swing.JEditorPane CItext;
    private javax.swing.JTextArea WarningDashboard;
    private javax.swing.JButton exitButton;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem1;
    private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem2;
    private javax.swing.JDialog jDialog1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenu jMenu3;
    private javax.swing.JMenu jMenu4;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JMenuItem jMenuItem4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JRadioButtonMenuItem jRadioButtonMenuItem1;
    private javax.swing.JRadioButtonMenuItem jRadioButtonMenuItem2;
    private javax.swing.JRadioButtonMenuItem jRadioButtonMenuItem3;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JScrollPane jScrollPane5;
    private javax.swing.JScrollPane jScrollPane6;
    private javax.swing.JTextArea jTextArea2;
    // End of variables declaration
}
```

7.29 WarningInfo Class

```
package com.mycompany.ClimateEducationApp;

public class WarningInfo {
    // WarningInfo class constructor
    public class Location {
        private String locationName;

        // Location class constructor
        public Location(String locationName) {
            this.locationName = locationName;
        }

        // Location getters and setters
        public String getLocationName() {
            return locationName;
        }
```

```
        public void setLocationName(String locationName) {
            this.locationName = locationName;
        }

        // Declaring Danger class
        public class Danger {
            private String danger;

            // Danger class constructor
            public Danger(String danger) {
                this.danger = danger;
            }

            // Danger getters and setters
            public String getDanger() {
                return danger;
            }

            public void setDescription(String description) {
                this.danger = description;
            }

            // Declaring PreventiveMeasures class
            public class PreventiveMeasures {
                private String preventiveMeasures;

                // PreventiveMeasures class constructor
                public PreventiveMeasures(String preventiveMeasures) {
                    this.preventiveMeasures = preventiveMeasures;
                }

                // PreventiveMeasures getters and setters
                public String getMeasure() {
                    return preventiveMeasures;
                }

                public void setMeasure(String preventiveMeasures) {
                    this.preventiveMeasures = preventiveMeasures;
                }
            }
        }
    }
}
```