Table of Contents

1.0 Introduction

1.1 Project Overview

"1 Minute" is a survival strategy game inspired by the challenges of the COVID-19 pandemic. Players are tasked with surviving a 30-day pandemic lockdown, where they must efficiently manage essential resources such as water, canned food, and medical supplies. Balancing hunger, thirst, health, and morale is crucial as each day brings new challenges, including random events and emergencies that demand quick decisions. The game challenges players' abilities in resource management and critical thinking, providing a realistic and immersive experience in navigating a pandemic through careful planning and strategic decision-making.

1.2 Technology Stack

To deliver the highest level of aesthetics, engagement, and functionality in "1 Minute," the following key technologies were used in its development:

1. **JavaFX** - JavaFX is the primary framework used for developing the user interface (UI) of "1 Minute". It offers a comprehensive set of UI controls, layouts, and visual effects for creating feature-rich applications through its extensive library of APIs. Additionally, JavaFX facilitates event handling for managing events generated by keyboard actions, mouse actions, and other source nodes. JavaFX also supports cross-platform development to ensure compatibility across different operating systems.

2. **Scene Builder** - Scene Builder is a necessary tool for JavaFX. It is a visual design tool used for quickly creating user interfaces (UI) for "1 Minute" without coding. It allows us to drag and drop UI components into a workspace, customize their properties, and apply style sheets. As we design, the corresponding FXML code for the layout is automatically generated in the background. This FXML file can then be integrated with a JavaFX project by linking the UI to the application's logic. developers.

3. **Figma -** Figma is a collaborative interface design tool used for prototyping the user interface (UI) and user experience (UX) of "1 Minute" before the development phase. With Figma's cloud-based platform, design teams can collaborate and share design ideas across the development team. Our team used Figma to prototype screens and navigation flows which involved creating visual representations of how different screens would look and function, and pathways through the game interface.

4. **Visual Studio Code -** Visual Studio Code was our chosen Integrated Development Environment (IDE) for coding. It provides a flexible and lightweight development environment. Visual Studio Code also provides extensions like the Scene Builder extension that facilitates integration with Scene Builder. Additionally, Visual Studio supports keyboard shortcuts for efficient task execution, thereby enhancing our coding experience and significantly improving our productivity.

5. **Backend Framework -** Java programming language has been chosen as the primary backend programming language for "1 Minute". Java is known for its wide variety of libraries and frameworks it provides, which is essential for various development needs. Besides that, Java's popularity and strong community support ensure continuous updates and improvements, making it a reliable choice for building scalable and maintainable backend systems in the game development context.

6. **Procreate -** Procreate is an advanced digital illustration app that our team used for creating concept art, sketches, and initial design ideas for "1 Minute". This tool was essential in the early stages of the game's development which allowed us to experiment with different visual styles and layouts. With its extensive set of tools and features, including layers, customizable brushes, and a wide range of color options, it allowed us to craft the initial visual concepts that helped define the distinct atmospheres and color palettes used throughout the game.

7. **Pixel Studio** - Pixel Studio is a pixel art editor used for crafting pixel art, mock-ups, and final designs of "1 Minute". Given that pixelated graphics were chosen as the main graphical user interface (GUI) for their effectiveness in synchronizing visual styles, it was the perfect tool for this purpose. Its specialized features for pixel art, such as a grid system, layer support, and an easy-to-use color palette made it ideal for creating the detailed and vibrant pixel-based graphics needed for the game.

1.3 Justification of UI Selection

Each section of our UI consists of an AnchorPane and an ImageView, with additional specialized UI elements tailored to fit our game.

- **AnchorPane:** Lays out the components with specific sizes and positions to maintain an organized and user-friendly layout.
- **ImageView(s):**
    - Place images on the login button and anchor pane to enhance the visual appeal and provide intuitive icons for actions.

1.3.1 Login Screen

- **TextField:** Enables the player to input their userId.
- **PasswordField**: Enables the player to input their password securely.
- **Label(s)**:
    - **Forget Password:** Enables the player to switch to the reset password screen when clicking the forgot password label.
    - **Register:** Enable the player to switch to the register screen when clicking the register label.
- **Button**: Enables the player to switch to the start screen upon clicking the login button, provided the login credentials are correct and registered.

1.3.2 Register Screen

- **TextField**: Enables the player to input their userId.
- **PasswordField**: Enables the player to input their password securely.
- **Label**: Enable the player to switch to the login screen when clicking the login label.
- **Button**: Enables the player to register upon clicking the register button, provided the userId does not already exist.

1.3.3 Reset Password Screen

- **TextField**: Enables the player to input their userId.
- **PasswordField(s)**:
    - **New Password**: Enables the player to input their new password securely.
    - **Confirm Password:** Enables the player to input their confirmed password securely that matches with the new password.
- **Label**: Enable the player to switch to the login screen when clicking the login label.
- **Button**: Enables the player to reset their password, provided they fill in all the required fields.

1.3.4 Start Screen

- **Button**: Enables the player to switch to the storyline when clicking the play button.

1.3.5 Storyline Screen

- **Label:** Displays the content for storyline screen.
- **Button**: Enables the player to switch to the shopping screen when clicking the continue.

1.3.6 Shopping Screen

- **TilePane:** Serves as a container for displaying shopping items in a shelf-like interface.
- **Text:** Used to display a description of the selected item in the user interface.
- **Label:** Act as placeholders where selected items from the item tilepane can be displayed.
- **ImageView(s):**
    - Contained within the labels, visually represents the items the user has added to their inventory.
- **Button**:
    - Each button within the TilePane represents an item available for selection on the shopping screen.
    - Enables the player to switch to the cashier screen when clicking the button.

1.3.7 Cashier Screen

- **TilePane:** Serves as a container for displaying shopping items in a shelf-like interface.
- **Label:** Act as placeholders where selected items from the shopping screen can be displayed.
- **ImageView(s):**
    - Contained within the labels, visually represents the items the user has added to their inventory.
- **Button**: Enables the player to switch to the daycounter screen when clicking the button.

1.3.8 Day Counter Screen

- **Text:** Displays the days which count up to day 30.
- **Button**: Enables the player to transition to the next day when clicking the arrow button, and allows the player to exit the day counter to the home screen.

1.3.9 Pause Screen

- **Button**: Enables the player to save, continue and exit the game when clicking on the respective buttons.

1.3.10 Home View

- **Button:** Enables the player to view the journal and allows them to pause the game when clicking on the respective buttons.

1.3.11 Journal Status

- **Button:** Enables the player to check character's status and navigate to other parts of the screen, like to the journal story or back to the home screen, when clicking on the respective buttons.

1.3.12 Journal Story

- **Button:** Enables the player to interact with the storyline, such as making choices or continuing to the next part of the story.
- **Text Area:** The TextArea is used to display the storyline content. This allows for multiline text display, which is essential for presenting journal entries and story details to the user. The TextArea is scrollable, ensuring that long texts can be read easily without overwhelming the user interface.

1.3.13 Defeat Screen

- **Button:** Enables the player to retry or quit the game when clicking on the respective buttons.

1.3.14 Winning Screen

- **Button:** Enables the player to return to the main menu when clicking on the respective button.

2.0 Gameplay Mechanics

2.1 Settings, start-up, and initialization

a) Start-up Process

Initial Screen: Upon launching the game, players are presented with an initial screen where they can perform the following actions:

- **Log In:** Players can log in using their registered userId and password.
- **Register:** New players can create an account for game access.
- **Reset Password:** Existing players can recover their password if forgotten.
- **User Profile:** Players can create new accounts or select existing ones to track progress and settings.

b) Initialization Process

Initial Setup: Upon starting a new game, the following processes occur:

- **Player's Resources Initialization:** The player's inventories such as water, canned food, medical supplies are initialized to default values.
- **Game World and Environment Setup:** The game world and player's environment such as house setup and initial day are set up.
- **Game State Variables Initialization :**Game state variables such as health, hunger, thirst, morale are initialized to appropriate starting.
- **Loading Textures and Sounds:**Textures and sounds are loaded at the start to ensure smooth gameplay.
- **Checking for Saved Games:** The game checks for saved games and sets initial conditions based on saved progress.

2.2 Game level and difficulty

a) **Levels:**

- The game consists of multiple levels representing days the player must survive.
- Each day introduces new challenges and decisions.

b) **Difficulty:**

- Difficulty increases progressively as the player survives more days.
- Early days are relatively easier, while later days pose tougher scenarios and resource management challenges.

Example:

- **Initial Days (1-10): Easy**
    - Difficulty: Low
    - Daily Events: Less challenging
    - Resource Availability: Higher
- **Mid Days (11-20): Medium**
    - Difficulty: Moderate
    - Daily Events: More challenging
    - Resource Availability: Moderate
- **Later Days (21-30): Hard**
    - Difficulty: High
    - Daily Events: Very challenging
    - Resource Availability: Scarce

2.3 Start play, pause, and saving

a) **Start Play:**

- After clicking ‘Play’ on the Start Screen, players will proceed to the Storyline Screen.
- To navigate to the next screen, players must click the ‘Continue’ button.

b) **Pause:**

- The player can pause the game at any time by clicking a designated pause button on the screen.
- While paused, the game halts all actions and displays a pause menu.

c) **Continue, Save, or Quit:**

- From the pause menu, the player can choose to continue playing, save the game progress, or quit the game entirely.
- Saving allows the player to resume from the last saved point in a future session, with game data stored in a binary data file.

2.4 Win, draw, or lose logic

a) Win Conditions:

- Survive all 30 days until the vaccine arrives.

b) Lose Conditions:

- **Death from Hunger or Thirst:** If any family member's hunger or thirst level reaches 0/100, that family member will die.
- **Death from Health or Morale:** If any family member's health or morale level drops to 0/100, that family member will die.
- **Family Member Deaths:** If two family members die due to any combination of hunger, thirst, health, or morale reaching critical levels, the game will end, resulting in a loss.

2.5 Visual and Audio Design

a) Visual Design:

- **Art Style**: The game features a pixel art style with vibrant colors and detailed environments. For example, item and character designs are created using Pixel Studio, as shown in Figure 1.

Figure 1 Item and character designs created using Pixel Studio

- **Color Palettes**: Carefully chosen color schemes create distinct atmospheres for different settings. The Winning Screen celebrates success with vibrant and cheerful colors, while the Defeat Screen reflects melancholy with dull and somber hues as shown in Figure 2.

Figure 2 Winning Screen with vibrant colors and Defeat Screen with somber hues

- **UI Design**: The user interface is designed to be intuitive and easy to navigate, featuring clear icons and menus. For instance, players can easily return to the Journal Story page by clicking the journal/book button below the heart/love button, or exit the Journal by clicking on the X in the top right corner of the screen as shown in Figure 3.

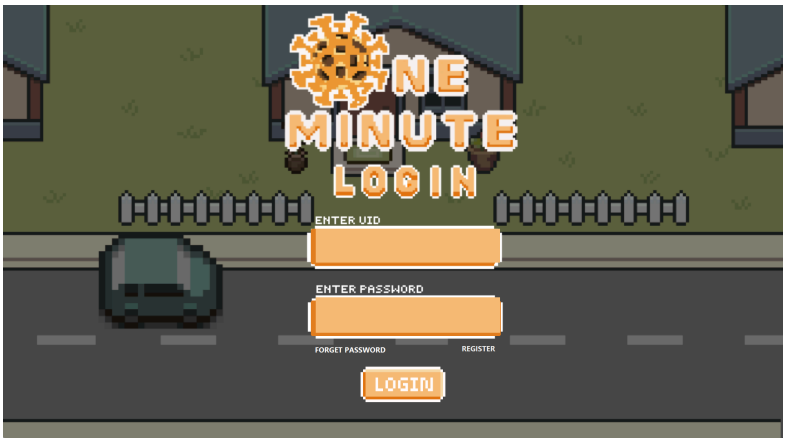
Figure 3 Clear icons for navigating the Journal Story page

b) Audio Design:

- **Music**: The game features a dynamic soundtrack that plays throughout, enhancing the overall atmosphere. For example, the background music (BGM) "littleSlimeAdventure.mp3" accompanies the player through different gameplay scenarios, making the experience more engaging and immersive.

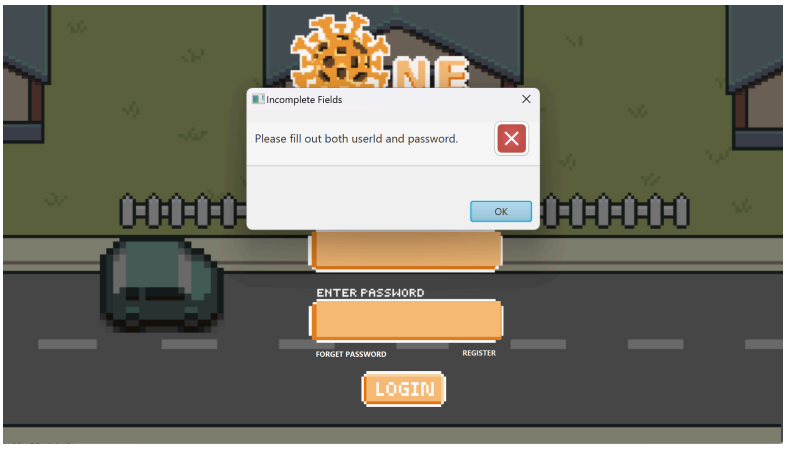
- **Sound Effects:** The game features sound effects that enhance the immersive experience of the player by providing auditory feedback for various actions and events. For example:
    - Journal: Includes page-flipping sound effects.
    - Cashier Screen: Features a 'kaching' sound effect after 60 seconds.

3.0 User Manual
3.1 Login

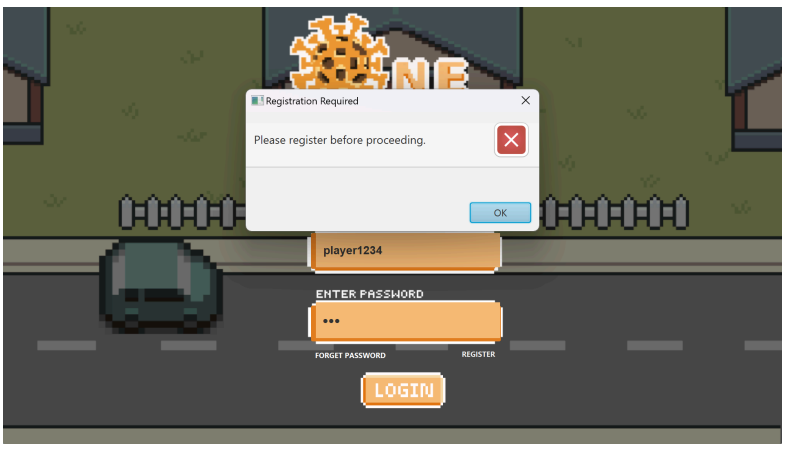


When starting up our game, players will be greeted by our login screen. Here, they can log in with their existing credentials. There are four possible outcomes:

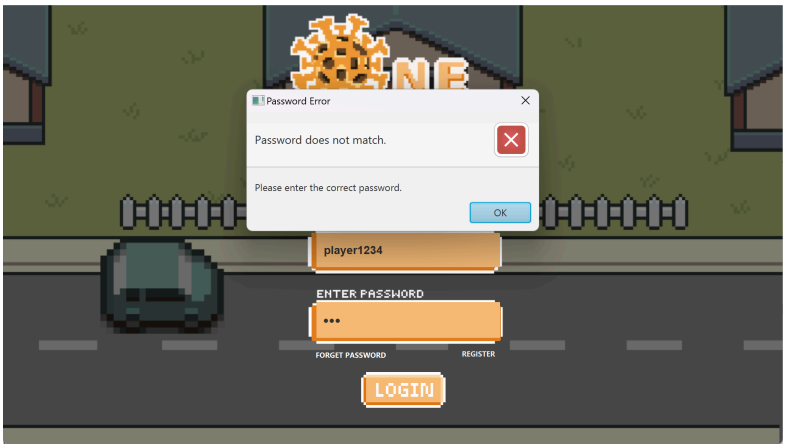
- **Successful Login**: Players will be transitioned to the start page, where they can begin playing the game.
- **Incomplete Credentials**: If players attempt to log in without filling in the textboxes, an alert will appear, prompting them to complete their credentials.



- **Unregistered Users:** If players try to log in without previously registering, an error will notify them to register before logging in. The register button is located bottom right of the 'Enter Password' textbox.



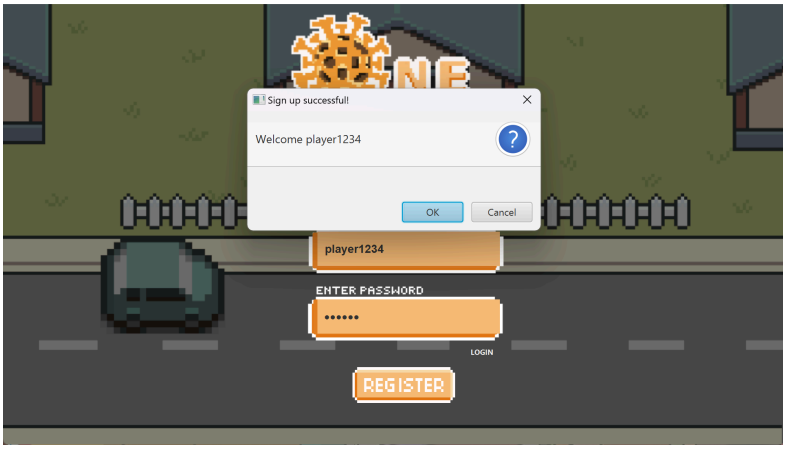
- **Incorrect Password**: If the entered password does not match the registered one, an error will inform players of the error. If players wish to change their password, they can click the 'Forgot Password' link at the bottom left of the 'Enter Password' textbox



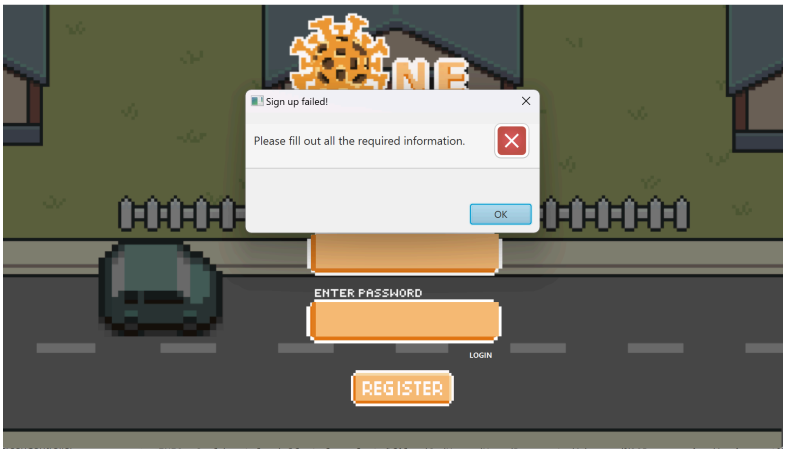
3.2 Register

Upon clicking the register button on the login page, players will be taken to the registration page, where they can enter their credentials and click on the ‘Register’ button when finished. There are three possible outcomes:

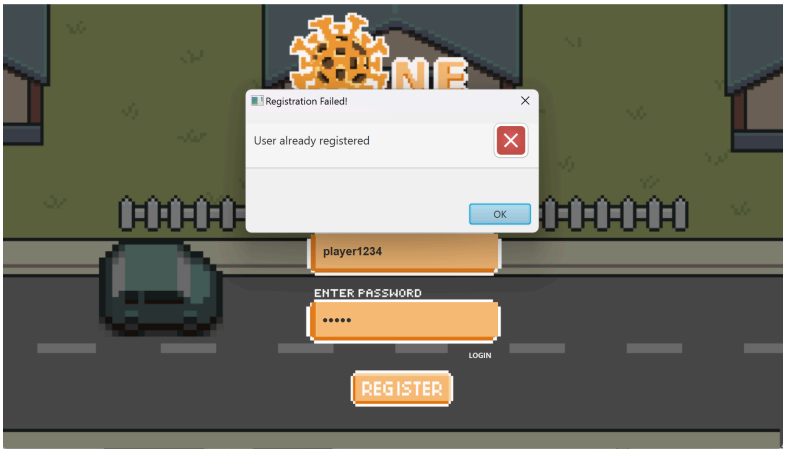
- **Successful Register:** Players can complete the registration process without any issues.



- **Incomplete Credentials:** If players attempt to register without filling in the required fields, an error will appear, prompting them to complete their credentials.



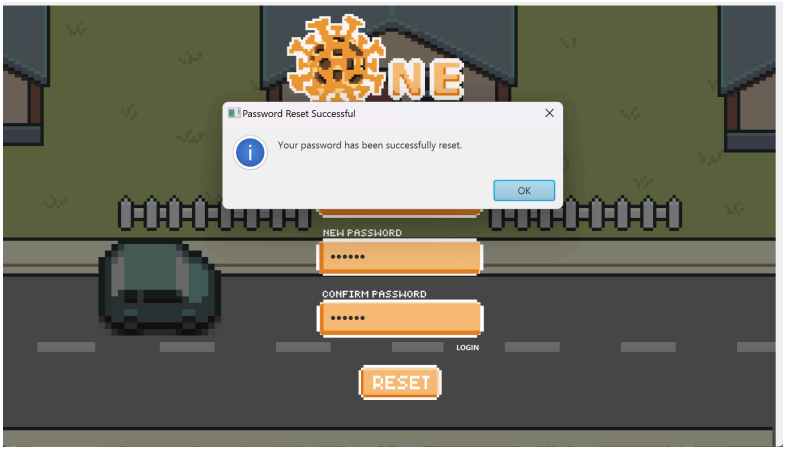
- **Registered Users:** If the provided credentials already exist, an error will notify players that they have already registered. They can return to the login page by clicking the login button located below the 'Enter Password' textbox.



3.3 Forget Password

Upon clicking the 'Forgot Password' button on the login page, players will be taken to the password reset page, where they can enter their credentials and click on the ‘Reset’ button when finished. There are four possible outcomes:

- **Successful Reset:** Players can complete the reset process without any issues.



- **Incomplete Credentials**: If players attempt to reset their password without filling in the required fields, an error will appear, prompting them to complete their credentials.

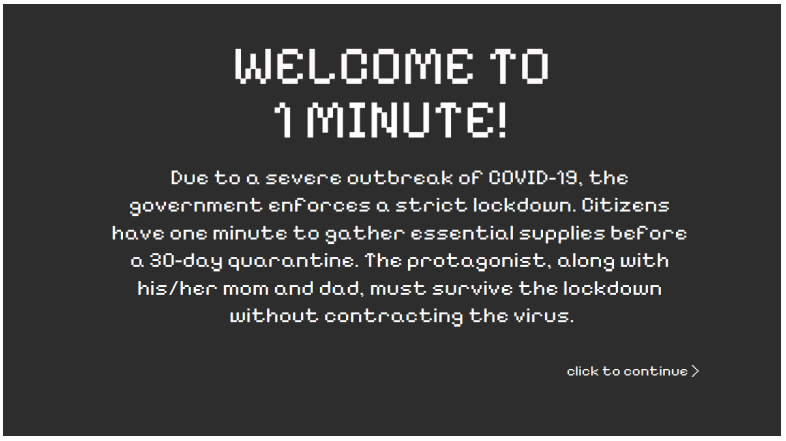
- **Incorrect UID**: If the UID is incorrect or unregistered, an error message will notify players of the issue.

- **Mismatched New Password and Confirm Password**: If the new password and confirm password fields do not match, an error will alert players to re-enter their passwords correctly.

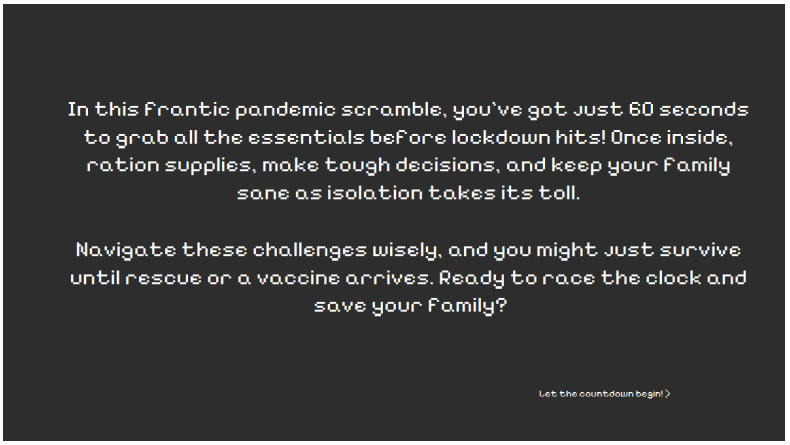
3.4 Start

After logging into the game, players will be taken to the start screen, where they can begin playing. Click on the 'Play' button to start!

3.5 Storyline



Players will arrive at this screen upon clicking ‘Play.’ This screen reveals the overarching plot of the game. To advance to the next screen, players must click the ‘Continue’ button.



The next screen explains the game rules. Clicking the ‘Let the Countdown Begin’ button will transition players to the grocer screen, where they can start choosing items to take with them during the lockdown.

3.6 Grocer

Upon clicking 'Let the Countdown Begin,' players can start selecting items to take with them into lockdown. Players will have 60 seconds to click on their desired items on the rack, which will then appear in their inventory bar at the bottom of the screen. When clicking an item, a description of the specific item will appear in the white text box, above the inventory.

If players are done before the 60-second countdown, they can click the arrow to leave before the countdown by clicking on the arrow button. Once the 60 seconds are up, players will be automatically directed to the Cashier Screen.

3.7 Cashier

In the cashier screen, players will have the opportunity to review the items they have selected. After reviewing, they can proceed to lockdown by clicking the 'Start' button.

3.8 Day Counter

After clicking ‘Start’ on the cashier screen, players will be brought to the Day Counter screen. Days will increment after clicking the arrow button in the bottom right corner.

To exit the Day Counter screen, players must click the close button at the top right corner of the screen. Once clicked, players will be redirected to the Home Screen.

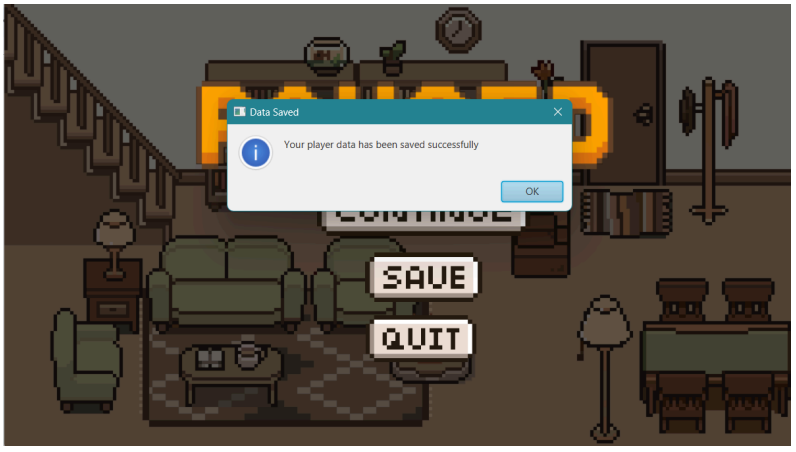
3.9 Home

The Home Screen consists of two buttons: the Journal Button and the Pause Button. The Journal Button is located at the bottom left corner, while the Pause Button is at the top right corner.

3.10 Pause

Upon clicking the Pause Button, players will be taken to the Pause Screen. Here, players will have the options to continue, save, or quit the game.

- **To Continue:** Players will be brought back to the Main Home Screen.
- **To Save:** An alert box will pop up confirming that your data has been saved.



- **To Quit:** A yes-or-no alert box will pop up to confirm if players want to quit the game. If "No" is selected, players will be brought back to the Pause Screen.

3.11 Journal - Story

Upon clicking the Journal Button, players will be taken to the Journal Story screen, where they can make decisions based on “Yes” or “No” events. Every day, there will be 1 event that prompts players to make their decision.

Example of Yes or No Event Outcomes:

‘Yes’ Decision ***‘No’ Decision***

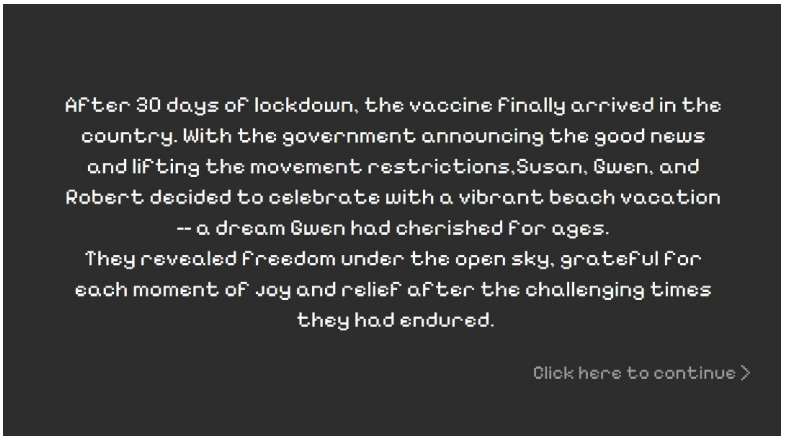
After clicking the "Yes" or "No" buttons, players will see the outcome displayed immediately in the middle of the journal. Each decision will lead to different outcomes. After that, players can click the right arrow button to proceed to the next day's counter.

3.12 Journal - Status

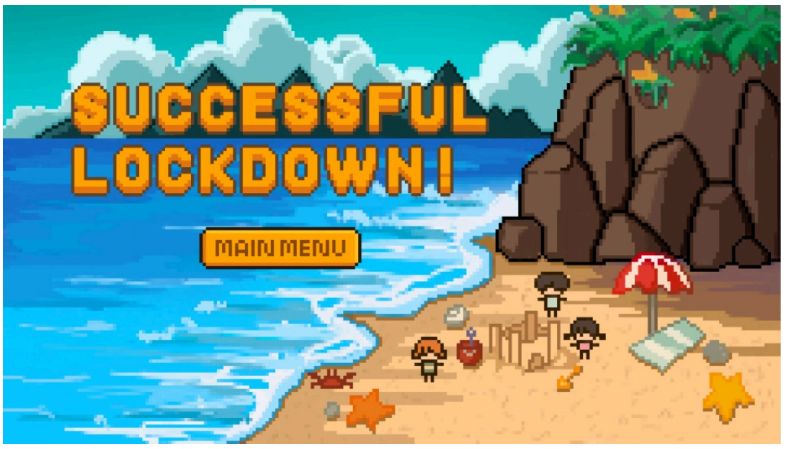
The Journal Status page displays the status of each family member. To reach this page, players can click the heart/love button. The Journal Status page contains values for health, hunger, thirst, and morale for every family member.

- **To Return to the Journal Story page:** Players can click the journal story (book) button below the status tab (heart shaped) button.
- **To Exit the Journal:** Players can click on the close button at the top right corner of the screen.

3.13 Win



If the 30 days have been successfully conducted and none of the family members caught covid nor died, players will be brought to this storyline screen. Clicking the 'Continue Button' will bring users to the final winning screen.



After players are transitioned to the final winning screen, they will have the option to return to the main menu. They will be redirected back to the Start Screen.

3.14 Defeat

However, if the 30-day lockdown is unsuccessful, players will be taken to the defeat screen, where they can choose to either quit the game or retry their attempt.

4.0 Testing Screenshots

Ensuring the quality and reliability of our game, "1 Minute", requires rigorous and comprehensive testing. Testing is a crucial phase in software development that helps identify and fix bugs, enhance performance, and ensure that the final product meets the desired standards. In our project, we employ several types of testing to validate different aspects of the game. These include:

i) Functional Testing

Test the create, retrieve, and update operations for each player by creating new passwords, updating the passwords, and retrieving specific entries. Verify that the system functions correctly at each stage. To accomplish this, use "System.out.println" to confirm that methods execute without errors when invoked. Figure 4 demonstrates the use of "System.out.println" when updating the player password.

```
public static void updatePassword(String userId, String newPassword) {
    System.out.println("Updating password for user ID: " + userId);
    List<Player> players = loadPlayerDataFromDatabase();
    for (Player player : players) {
        if (player.getUserId().equals(userId)) {
            System.out.println("Updating password from '" + player.getPassword() + "' to '" + newPassword + "'");
            player.setPassword(newPassword);
            break;
        }
    }
    savePlayerDataToDatabase(players);
    System.out.println(x:"Password update completed.");
}
```

Figure 4 Verify password update functionality

Outcome: Figure 5 shows the outcome of updating the password successfully: a confirmation message indicating success.

Figure 5 Successful Outcome of the Password Update Functionality

ii) Security Testing

Verify the authentication mechanisms by attempting to log in with both correct and incorrect credentials as shown in Figure 6. Ensure that the system correctly applies access controls to prevent unauthorized personnel from accessing protected information.

```
if (userId.getText().equals(Username) && password.getText().equals(Password)) {
    found = true;
    PlayerData.initPlayerData.setUserId(userId.getText());
    PlayerData.initPlayerData.setPassword(password.getText());
    System.out.println(x:"Login successful! Switching to StartScreen.");
    App.setRoot(fxml:"StartScreen");
    break;
}
```

Figure 6 Testing login with valid credentials

Outcome: Figure 7 shows the outcome of log in with valid credentials: a switch to the Start Screen.

Figure 7 Outcome of login with valid credentials

```
if (username.equals(storedUsername)) {
    // Check if input password matches the registered password
    if (!password.equals(storedDecryptedPassword)) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Password Error");
        alert.setHeaderText("Password does not match.");
        alert.setContentText("Please enter the correct password.");
        alert.showAndWait();
        return false;
    }
    return true; // Password matches, user is registered
}
```

Figure 8 Testing login with invalid credentials

Outcome: Figure 9 shows the outcome of log in with invalid credentials: an error message indicating failure.

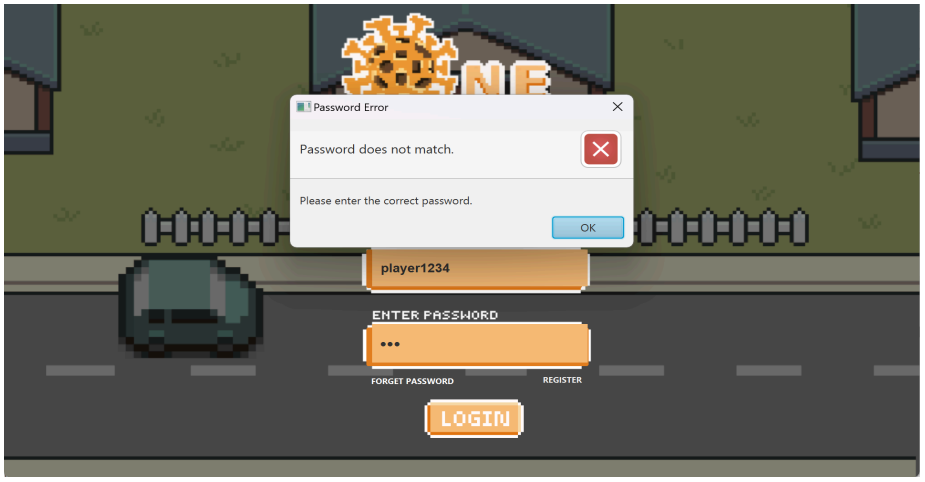


Figure 9 Outcome of login with invalid credentials

iii) Data Validation Testing

Validate the input fields by submitting empty values, such as leaving both the userId and password fields blank when prompted for strings as shown in Figure 10. Ensure that the system accurately detects and manages any errors or invalid inputs that occur.

```
@FXML
void switchToStartScreen(ActionEvent event) throws IOException {
    // Check if userId or password fields are empty
    if (userId.getText().isEmpty() || password.getText().isEmpty()) {
        showAlert(title:"Incomplete Fields", message:"Please fill out both userId and password.");
        return;
    }
```

Figure 10 Testing empty userId and password fields for error detection

Outcome: Figure 11 shows the outcome of submitting empty values while logging in: an error message indicating failure.

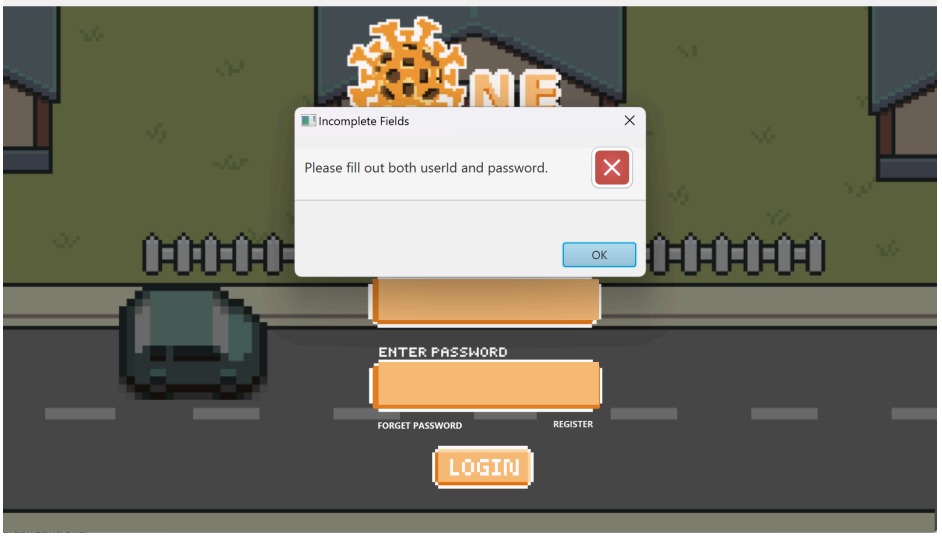


Figure 11 Outcome of submitting empty values

iv) User Interface Testing

Verify that the user interface elements, including buttons, and labels are functional and responsive. Test the navigation flow by performing tasks like registering a player, logging in successfully, and navigating through different screens.

v) Regression Testing

After adding new features or making changes to existing code in “1 Minute”, we conduct retesting on previously confirmed functionalities to verify their continued proper operation. This includes ensuring that any addressed bugs are resolved and that these fixes do not adversely affect other aspects of the game.

5.0 Conclusion

In conclusion, we have conducted self-learning through different types of technology tools. Our group chose JavaFx as the primary framework to develop the user interface (UI) and complemented by Scene Builder as the visual design tool for creating our layouts and integrating them with the game's logic. Moreover, we have used Figma for UI/UX prototyping and Pixel Studio as the platform to design the pixel-based graphics featured in our game. We have engaged in self-learning to master these platforms for the development of our game, "1 Minute".

Besides JavaFX programming and pixel designing, we learned how to compose ourselves to work effectively in a team. Five of us were in charge of different parts of the assignment, and we conducted multiple meetings to coordinate our efforts. By working together, we learned the essence of why teamwork and communication are key to getting the job done.

Through "1 Minute" , we aimed to provide players with not just entertainment, but also a deeper understanding of strategic planning and resource allocation under pressure. The game's simulation of pandemic survival underscores the importance of thoughtful decision-making in crisis situations.

In the future, we plan to implement the game settings feature which includes audio settings, graphics settings, control settings and language settings. By implementing these settings, our game will be more user friendly and customizable to meet diverse player preferences.

Due to time constraints, the current version of "1 Minute" simulates a 14-day lockdown period. However, our original vision was for a 30-day survival challenge, and we plan to implement this extended duration in future updates. This will allow for even more intricate gameplay and strategic depth, providing a richer experience for players.

6.0 Source Code and OOP Concepts implemented

6.1 Main Class

App.java

```
package oneminute;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import oneminute.classes.Audio;
import oneminute.classes.DayCounterApp;

import java.io.IOException;

public class App extends Application {

    private static Scene scene;
    private static Stage primaryStage;
    private static Audio audioManager;
    private static DayCounterApp dayCounterApp;

    @Override
    public void start(Stage stage) throws IOException {
        primaryStage = stage;
        audioManager = new Audio();
        audioManager.playBackgroundMusic("game");
        audioManager.setBackgroundMusicVolume("game", 0.5);
        dayCounterApp = new DayCounterApp(primaryStage);

        // Load initial scene (RegisterScreen)
        Parent root = loadFXML("RegisterScreen");
        scene = new Scene(root, 960, 540);
        stage.setTitle("OneMinute");
        stage.setScene(scene);
        stage.show();
    }

    public static void setRoot(String fxml) throws IOException {
        Parent root = loadFXML(fxml);
        scene.setRoot(root);
    }

    private static FXMLLoader loadFXMLLoader(String fxml) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(App.class.getResource("/oneminute/fxml/" +
fxml + ".fxml"));
        if (fxmlLoader.getLocation() == null) {
            throw new IllegalStateException("FXML file " + fxml + " not found");
        }
        return fxmlLoader;
    }

    private static Parent loadFXML(String fxml) throws IOException {
        return loadFXMLLoader(fxml).load();
    }

    public static Audio getAudioManager() {
        return audioManager; // Provide access to the audio manager
    }

    public static void nextDay() throws IOException {
        dayCounterApp.nextDay();
    }

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

6.2 Models

Player.java

```
package oneminute.classes;

public class Player {
    private String userId;
    private String password;

    public Player(String userId, String password) {
        this.userId = userId;
        this.password = password;
    }

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Item.java

```
package oneminute.classes;

public class Item {
    private final String itemName;
    private final String itemDescription;
    private final String itemUrl;

    public Item(String itemName, String itemDescription, String itemUrl) {
        this.itemName = itemName;
        this.itemDescription = itemDescription;
        this.itemUrl = itemUrl;
    }

    public String getItemname() {
        return itemName;
    }

    public String getDescription() {
        return itemDescription;
    }

    public String getItemUrl() {
        return itemUrl;
    }
}
```

Inventory.java

```
package oneminute.classes;

import java.util.ArrayList;
import java.util.List;


public class Inventory {

    private List<Item> items;

    public Inventory() {
        this.items = new ArrayList<>();
    }
```

```
    public void addItem(Item item) {
        items.add(item);
    }

    public void removeItem(Item item) {
        items.remove(item);
    }

    public List<Item> getItems() {
        return items;
    }
}
```

FamilyMemberStatus.java

```
package oneminute.classes;

public class FamilyMemberStatus {
    private String name;
    private int health;
    private int hunger;
    private int thirst;
    private int morale;

    public FamilyMemberStatus(String name, int health, int hunger, int thirst, int morale) {
        this.name = name;
        this.health = health;
        this.hunger = hunger;
        this.thirst = thirst;
        this.morale = morale;
    }

    // Getters and setters for each attribute
    public String getName() {
        return name;
    }

    public int getHealth() {
        return health;
    }

    public void setHealth(int health) {
        this.health = health;
    }

    public int getHunger() {
        return hunger;
    }

    public void setHunger(int hunger) {
        this.hunger = hunger;
    }

    public int getThirst() {
        return thirst;
    }

    public void setThirst(int thirst) {
        this.thirst = thirst;
    }

    public int getMorale() {
        return morale;
    }

    public void setMorale(int morale) {
        this.morale = morale;
    }
}
```

DayCounter.java

```
package oneminute.classes;

import java.util.ArrayList;
import oneminute.controllers.*;

public class DayCounter {
    private int currentDay;
    private final ArrayList<FamilyMemberStatus> familyMembers;
    private JournalStatusController statusController; // Controller reference if needed

    public DayCounter() {
        currentDay = 1; // Start at day 1
        this.familyMembers = new ArrayList<>();
        initializeFamilyMembers();
    }

    public int getCurrentDay() {
        return currentDay;
    }

    public void incrementDay() {
        if (currentDay < 30) {
            currentDay++;
        }
    }

    public boolean isMaxDayReached() {
        return currentDay >= 30;
    }

    private void initializeFamilyMembers() {
        familyMembers.add(new FamilyMemberStatus("Susan", 80, 70, 70, 70));
        familyMembers.add(new FamilyMemberStatus("Robert", 80, 70, 70, 70));
        familyMembers.add(new FamilyMemberStatus("Gwen", 80, 70, 70, 70));
    }

    public void nextDay() {
        try {
            if (!isMaxDayReached()) {
                incrementDay();
                updateFamilyMembersStatus();
            } else {
                System.out.println("Day limit reached. Cannot increment further.");
                // Optionally, display an alert to the user
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void updateFamilyMembersStatus() {
        for (FamilyMemberStatus member : familyMembers) {
            member.setHunger(member.getHunger() - 5);
            member.setThirst(member.getThirst() - 5);
            member.setMorale(member.getMorale() - 1);
            member.setHealth(member.getHealth() - 2);
        }
        if (statusController != null) {
            statusController.updateStatusBars();
        }
    }

    public ArrayList<FamilyMemberStatus> getFamilyMembers() {
        return familyMembers;
    }

    public void setStatusController(JournalStatusController statusController) {
        this.statusController = statusController;
    }

    public String getFormattedFamilyMembersStatus() {
        StringBuilder statusBuilder = new StringBuilder();
        for (FamilyMemberStatus member : familyMembers) {
```

```
            statusBuilder.append(member.getName()).append(" Health Value:
").append(member.getHealth()).append("/100\n");
            statusBuilder.append(member.getName()).append(" Hunger Value:
").append(member.getHunger()).append("/100\n");
            statusBuilder.append(member.getName()).append(" Thirst Value:
").append(member.getThirst()).append("/100\n");
            statusBuilder.append(member.getName()).append(" Morale Value:
").append(member.getMorale()).append("/100\n");
        }
        return statusBuilder.toString();
    }
}
```

DayCounterApp.java

```
package oneminute.classes;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import oneminute.controllers.DayCounterController;
import oneminute.controllers.HomeScreenController;
import oneminute.controllers.JournalStatusController;
import oneminute.controllers.JournalStoryController;

import java.io.IOException;
import java.util.ArrayList;

public class DayCounterApp {

    private final DayCounter dayCounter;
    private final ArrayList<FamilyMemberStatus> familyMembers;
    private Stage primaryStage;
    private JournalStatusController statusController;

    public DayCounterApp(Stage primaryStage) {
        this.primaryStage = primaryStage;
        this.dayCounter = new DayCounter();
        this.familyMembers = new ArrayList<>();
        initializeFamilyMembers();
    }

    private void initializeFamilyMembers() {
        familyMembers.add(new FamilyMemberStatus("Susan", 80, 70, 70, 70));
        familyMembers.add(new FamilyMemberStatus("Robert", 80, 70, 70, 70));
        familyMembers.add(new FamilyMemberStatus("Gwen", 80, 70, 70, 70));
    }

    public void switchToDayCounterScreen() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/oneminute/fxml/DayCounter.fxml"));
            Parent root = loader.load();

            DayCounterController controller = loader.getController();
            controller.setDayCounter(dayCounter);
            controller.setApp(this);
            controller.setPrimaryStage(primaryStage);

            Scene scene = new Scene(root);
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void switchToHomeScreen() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/oneminute/fxml/HomeView.fxml"));
            Parent root = loader.load();

            HomeScreenController controller = loader.getController();
```

```
            controller.setApp(this);
            controller.setPrimaryStage(primaryStage);

            Scene scene = new Scene(root);
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void switchToJournalStoryScreen() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/oneminute/fxml/JournalStory.fxml"));
            Parent root = loader.load();

            JournalStoryController controller = loader.getController();
            controller.setApp(this);
            controller.setPrimaryStage(primaryStage);

            Scene scene = new Scene(root);
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void switchToJournalStatusScreen() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/oneminute/fxml/JournalStatus.fxml"));
            Parent root = loader.load();

            JournalStatusController controller = loader.getController();
            controller.setApp(this);
            controller.setPrimaryStage(primaryStage);
            this.statusController = controller;

            Scene scene = new Scene(root);
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void nextDay() {
        try {
            if (!dayCounter.isMaxDayReached()) {
                dayCounter.incrementDay();
                updateFamilyMembersStatus();
                switchToDayCounterScreen(); // Update the scene for the next day
            } else {
                System.out.println("Day limit reached. Cannot increment further.");
                // Optionally, display an alert to the user
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void updateFamilyMembersStatus() {
        for (FamilyMemberStatus member : familyMembers) {
            member.setHunger(member.getHunger() - 5);
            member.setThirst(member.getThirst() - 5);
            member.setMorale(member.getMorale() - 1);
            member.setHealth(member.getHealth() - 2);
        }
        if (statusController != null) {
            statusController.updateStatusBars();
        }
    }
```

```
    public ArrayList<FamilyMemberStatus> getFamilyMembers() {
        return familyMembers;
    }

    public DayCounter getDayCounter() {
        return dayCounter;
    }

    public String getFormattedFamilyMembersStatus() {
        StringBuilder statusBuilder = new StringBuilder();
        for (FamilyMemberStatus member : familyMembers) {
            statusBuilder.append(member.getName()).append(" Health Value:
").append(member.getHealth()).append("/100\n");
            statusBuilder.append(member.getName()).append(" Hunger Value:
").append(member.getHunger()).append("/100\n");
            statusBuilder.append(member.getName()).append(" Thirst Value:
").append(member.getThirst()).append("/100\n");
            statusBuilder.append(member.getName()).append(" Morale Value:
").append(member.getMorale()).append("/100\n");
        }
        return statusBuilder.toString();
    }
}
```

Audio.java

```
package oneminute.classes;

import javafx.scene.media.AudioClip;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;

public class Audio {
    private MediaPlayer gameBgm;
    private MediaPlayer shoppingBgm;
    private AudioClip buttonStart;
    private AudioClip buttonClick;
    private AudioClip buttonTick;
    private AudioClip bookFlip;
    private AudioClip cashierRegister;

    public Audio() {
        // Initialize background music players
        gameBgm = new MediaPlayer(new
Media(getClass().getResource("/oneminute/audio/bgm-game.mp3").toExternalForm()));
        shoppingBgm = new MediaPlayer(new
Media(getClass().getResource("/oneminute/audio/bgm-shopping.mp3").toExternalForm()));

        // Initialize sound effect clips
        buttonStart = new
AudioClip(getClass().getResource("/oneminute/audio/button-start.mp3").toExternalForm());
        buttonClick = new
AudioClip(getClass().getResource("/oneminute/audio/button-click.mp3").toExternalForm());
        buttonTick = new
AudioClip(getClass().getResource("/oneminute/audio/button-tick.mp3").toExternalForm());
        bookFlip = new
AudioClip(getClass().getResource("/oneminute/audio/button-bookflip.mp3").toExternalForm());
        cashierRegister = new
AudioClip(getClass().getResource("/oneminute/audio/bgm-cashier.mp3").toExternalForm());
    }

    // Background Music Settings
    public void playBackgroundMusic(String type) {
        stopAllBackgroundMusic();
        switch (type) {
            case "game":
                playMusic(gameBgm);
                break;
            case "shopping":
                playMusic(shoppingBgm);
                break;
            default:
                break;
        }
    }
```

```
    private void playMusic(MediaPlayer mediaPlayer) {
        if (mediaPlayer != null) {
            mediaPlayer.setCycleCount(MediaPlayer.INDEFINITE);
            mediaPlayer.play();
        }
    }

    public void stopAllBackgroundMusic() {
        stopMusic(gameBgm);
        stopMusic(shoppingBgm);
    }

    private void stopMusic(MediaPlayer mediaPlayer) {
        if (mediaPlayer != null) {
            mediaPlayer.stop();
        }
    }

    public void setBackgroundMusicVolume(String type, double volume) {
        switch (type) {
            case "game":
                setVolume(gameBgm, volume);
                break;
            case "shopping":
                setVolume(shoppingBgm, volume);
                break;
            default:
                break;
        }
    }

    private void setVolume(MediaPlayer mediaPlayer, double volume) {
        if (mediaPlayer != null) {
            mediaPlayer.setVolume(volume);
        }
    }

    // Sound Effect Settings
    public void playSoundEffect(String type) {
        switch (type) {
            case "buttonStart":
                playSoundEffect(buttonStart);
                break;
            case "buttonClick":
                playSoundEffect(buttonClick);
                break;
            case "buttonTick":
                playSoundEffect(buttonTick);
                break;
            case "bookFlip":
                playSoundEffect(bookFlip);
                break;
            case "cashierRegister":
                playSoundEffect(cashierRegister);
                break;
            default:
                break;
        }
    }

    private void playSoundEffect(AudioClip clip) {
        clip.play();
    }

    public void setSoundEffectVolume(AudioClip clip, double volume) {
        if (clip != null) {
            clip.setVolume(volume);
        }
    }
}
```

6.3 Data

playerData.java

```
package oneminute.data;

import java.io.Serializable;
import java.util.List;

    public class playerData implements Serializable {
    // Define attributes to be saved
    private String playerUID;
    private int currentDaySaved; //save current day stopped
    private String gwenStatus; //gwen's status "healthy, sick or dead"
    private String susanStatus;
    private String robertStatus;
    private List<String> inventoryItems; //assuming items are stored as a string

    // Constructor
    public playerData(String playerUID, int currentDaySaved, String gwenStatus, String
susanStatus, String robertStatus, List<String> inventoryItems) {
        this.playerUID = playerUID; this.currentDaySaved = currentDaySaved; this.gwenStatus
= gwenStatus; this.susanStatus = susanStatus; this.robertStatus = robertStatus;
this.inventoryItems = inventoryItems;
}
    // Getters and setters
    public String getPlayerUID() {
        return playerUID;
    }

   public void setPlayerUid(String playerUid) {
        this.playerUID = playerUid;
    }

    public int getCurrentDaySaved() {
        return currentDaySaved;
    }

    public void setCurrentDaySaved(int currentDaySaved) {
        this.currentDaySaved = currentDaySaved;
    }

    public String getGwenStatus() {
        return gwenStatus;
    }

    public void setGwenStatus(String gwenStatus) {
        this.gwenStatus = gwenStatus;
    }

    public String getSusanStatus() {
        return susanStatus;
    }

    public void setSusanStatus(String susanStatus) {
        this.susanStatus = susanStatus;
    }

    public String getRobertStatus() {
        return robertStatus;
    }

    public void setRobertStatus(String robertStatus) {
        this.robertStatus = robertStatus;
    }

    public List<String> getInventoryItems() {
        return inventoryItems;
    }

    public void setInventoryItems(List<String> inventoryItems) {
        this.inventoryItems = inventoryItems;
    }
```

```
    @Override
    public String toString() {
        return "playerData {" + "UID = ' " + playerUID + '\'' +
                               ", current day = " + currentDaySaved +
                               ", Gwen Status = " + gwenStatus +
                               ", Susan Status = " + susanStatus +
                               ", Robert Status = " + robertStatus +
                               ", inventory = " + inventoryItems + '}';
    }
}
```

usernameData.java

```
package oneminute.data;

import java.io.*;
import java.security.Key;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

import oneminute.classes.Player;

public class usernameData {

    public static String fileName = "src/main/resources/oneminute/database/Player.txt";
    private static final String ALGORITHM = "AES";
    private static final byte[] keyValue = "MySuperSecretKey".getBytes(); // Must be 16
bytes for AES-128

    // Load player data from the database
    public static List<Player> loadPlayerDataFromDatabase() {
        List<Player> playerList = new ArrayList<>();

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] playerData = line.split(",");
                if (playerData.length >= 2) {
                    String playerUserId = playerData[0].trim();
                    String passwordOrEncrypted = playerData[1].trim();
                    String playerPassword;

                    if (isBase64Encoded(passwordOrEncrypted)) {
                        try {
                            playerPassword = decrypt(passwordOrEncrypted);
                            System.out.println("Decryption successful for user: " +
playerUserId);
                        } catch (Exception e) {
                            playerPassword = passwordOrEncrypted; // Treat as plain text if
decryption fails
                            System.out.println("Decryption failed for user: " + playerUserId
+ ". Treating password as plain text.");
                        }
                    } else {
                        playerPassword = passwordOrEncrypted;
                        System.out.println("Plain text password detected for user: " +
playerUserId);
                    }

                    Player newPlayer = new Player(playerUserId, playerPassword);
                    playerList.add(newPlayer);
                } else {
                    System.err.println("Invalid data format in Player.txt: " + line);
                }
            }
        } catch (IOException e) {
            System.err.println("Error reading player data from Player.txt: " +
e.getMessage());
            e.printStackTrace();
        }
```

```
        return playerList;
    }

    public static boolean isUserIdRegistered(String userId) {
        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] playerData = line.split(",");
                if (playerData.length >= 1) {
                    String playerUserId = playerData[0].trim();
                    if (playerUserId.equals(userId)) {
                        return true;
                    }
                } else {
                    System.err.println("Invalid data format in Player.txt: " + line);
                }
            }
        } catch (IOException e) {
            System.err.println("Error reading player data from Player.txt: " +
e.getMessage());
            e.printStackTrace();
        }
        return false;
    }

    public static void updatePassword(String userId, String newPassword) {
        System.out.println("Updating password for user ID: " + userId);
        List<Player> players = loadPlayerDataFromDatabase();
        for (Player player : players) {
            if (player.getUserId().equals(userId)) {
                System.out.println("Updating password from '" + player.getPassword() + "' to
'" + newPassword + "'");
                player.setPassword(newPassword); // Store plain text password temporarily
                savePlayerDataToDatabase(players); // Save updated passwords
                break;
            }
        }
        System.out.println("Password update completed.");
    }

    public static void savePlayerDataToDatabase(List<Player> playerList) {
        try (FileWriter account = new FileWriter(fileName, false)) {
            PrintWriter accountWriter = new PrintWriter(account);

            for (Player player : playerList) {
                accountWriter.println(player.getUserId() + "," +
encrypt(player.getPassword()));
            }
            accountWriter.close();
        } catch (IOException e) {
            System.err.println("Error saving player data to Player.txt: " + e.getMessage());
            e.printStackTrace();
        }
    }
    // Initialize a Player object that can be used for reference elsewhere
    public static Player initPlayerData = new Player(null, null);

    // Encrypt a string using AES algorithm
    public static String encrypt(String value) {
        try {
            Key key = generateKey();
            Cipher cipher = Cipher.getInstance(ALGORITHM);
            cipher.init(Cipher.ENCRYPT_MODE, key);
            byte[] encryptedValue = cipher.doFinal(value.getBytes());
            return Base64.getEncoder().encodeToString(encryptedValue);
        } catch (Exception e) {
            throw new RuntimeException("Error while encrypting", e);
        }
    }

    // Decrypt a string using AES algorithm
    public static String decrypt(String value) {
        try {
            Key key = generateKey();
            Cipher cipher = Cipher.getInstance(ALGORITHM);
```

```
            cipher.init(Cipher.DECRYPT_MODE, key);
            byte[] decodedValue = Base64.getDecoder().decode(value);
            byte[] decryptedValue = cipher.doFinal(decodedValue);
            return new String(decryptedValue);
        } catch (Exception e) {
            throw new RuntimeException("Error while decrypting", e);
        }
    }

    private static Key generateKey() {
        return new SecretKeySpec(keyValue, ALGORITHM);
    }

    private static boolean isBase64Encoded(String value) {
        try {
            Base64.getDecoder().decode(value);
            return true;
        } catch (IllegalArgumentException e) {
            return false;
        }
    }
}
```

6.4 Lists

itemList.java

```
package oneminute.lists;

import java.util.ArrayList;
import oneminute.classes.*;

public class ItemList {

    private ArrayList<Item> itemInitList;

    public ArrayList<Item> getItemList() {
        itemInitList = new ArrayList<>();
        itemInitList.add(new Item("Battery", "For keeping your devices alive longer than
your sanity.", "/oneminute/images/items/item_Battery.png"));
        itemInitList.add(new Item("Board Game", "Entertainment that doesn't need a Wi-Fi
signal.","/oneminute/images/items/item_BoardGame.png"));
        itemInitList.add(new Item("Books", "Escaping reality one page at a time, because
adventures are essential, even indoors.","/oneminute/images/items/item_Book.png"));
        itemInitList.add(new Item("Bug Spray", "Because even lockdown won't keep pests
away.","/oneminute/images/items/item_BugSpray.png"));

        itemInitList.add(new Item("Candles", "Ambiance and emergency lighting rolled into
one.","/oneminute/images/items/item_Candle.png"));
        itemInitList.add(new Item("Headphone", "Music: your escape from the four walls
closing in.","/oneminute/images/items/item_Headphone.png"));
        itemInitList.add(new Item("Canned Foods", "Comfort in a tin can, because cooking's
overrated.","/oneminute/images/items/item_CannedFood.png"));
        itemInitList.add(new Item("Cards", "Old-school fun for when the internet's
down.","/oneminute/images/items/item_Cards.png"));

        itemInitList.add(new Item("Perfume", "Smelling good, even when there's no one to
impress.","/oneminute/images/items/item_Perfume.png"));
        itemInitList.add(new Item("Deodorant", "Because social distancing doesn't mean
smelling bad.","/oneminute/images/items/item_Deodorant.png"));
        itemInitList.add(new Item("Ice Cream", "Cold, creamy solace in a world gone
crazy.","/oneminute/images/items/item_IceCream.png"));
        itemInitList.add(new Item("Maps", "Finding new escape routes or just
daydreaming.","/oneminute/images/items/item_Map.png"));

        itemInitList.add(new Item("First Aid Kit", "For those unexpected 'ouch' moments at
home.","/oneminute/images/items/item_FirstAidKit.png"));
        itemInitList.add(new Item("Soaps", "Warding off germs and maintaining lockdown
hygiene.","/oneminute/images/items/item_Soap.png"));
        itemInitList.add(new Item("Weapon", "For when social distancing fails as a
negotiation tactic.","/oneminute/images/items/item_Weapon.png"));
        itemInitList.add(new Item("Flashlight", "Shining a light on things that go bump in
the night.","/oneminute/images/items/item_Flashlight.png"));

        itemInitList.add(new Item("Radio", "Stay connected to the world outside your front
```

```
door.","/oneminute/images/items/item_Radio.png"));
        itemInitList.add(new Item("Pet Foods", "Because even furry friends need comfort
during lockdown.","/oneminute/images/items/item_PetFood.png"));
        itemInitList.add(new Item("Matches", "Fire-starting expertise, for warmth or cooking
adventures.","/oneminute/images/items/item_Matches.png"));
        itemInitList.add(new Item("Medicine", "Prescription for peace of mind in uncertain
times.","/oneminute/images/items/item_Medicine.png"));

        itemInitList.add(new Item("Pillow", "Comfort for those unexpected naptime
marathons.","/oneminute/images/items/item_Pillow.png"));
        itemInitList.add(new Item("Toilet Paper", "Defender of personal hygiene, even in
quarantine.","/oneminute/images/items/item_ToiletPaper.png"));
        itemInitList.add(new Item("Water", "Hydration: because survival never takes a sick
day.","/oneminute/images/items/item_Water.png"));
        itemInitList.add(new Item("Suitcase", "Packing for the great adventure of staying
indoors.","/oneminute/images/items/item_Suitcase.png"));

        itemInitList.add(new Item("Stationary", "From cutting paper to makeshift haircuts,
because survival requires versatile tools.","/oneminute/images/items/item_Stationary.png"));

        return itemInitList;
    }
}
```

6.5 Controllers

RegisterScreenController.java

```
package oneminute.controllers;

import java.io.FileWriter;
import java.io.PrintWriter;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.input.MouseEvent;
import oneminute.App;
import oneminute.classes.Player;
import oneminute.data.usernameData;

public class RegisterScreenController {

    @FXML
    private Label loginLabel;

    @FXML
    private PasswordField password;

    @FXML
    private TextField userId;

    @FXML
    private Button registerButton;

    @FXML
    void register(ActionEvent event) throws Exception {
        App.getAudioManager().playSoundEffect("buttonClick");
        try {
            if (!(userId.getText()).isEmpty() && !(password.getText().isEmpty())) {
                if (usernameData.isUserIdRegistered(userId.getText())) {
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setTitle("Registration Failed!");
                    alert.setHeaderText("User already registered");
                    alert.showAndWait();
                    return;
                }

                String encryptedPassword = usernameData.encrypt(password.getText());
```

```
                Player newPlayer = new Player(userId.getText(), encryptedPassword);

                System.out.println("New Player: " + newPlayer.getUserId() + ", " +
newPlayer.getPassword());

                try (FileWriter account = new
FileWriter("src/main/resources/oneminute/database/Player.txt", true);
                     PrintWriter accountWriter = new PrintWriter(account)) {

                    // Write new user into database
                    accountWriter.println(newPlayer.getUserId() + "," +
newPlayer.getPassword());
                    System.out.println("New player registered: " + newPlayer.getUserId());

                    Alert alert = new Alert(AlertType.CONFIRMATION);
                    alert.setTitle("Sign up successful!");
                    alert.setHeaderText("Welcome " + userId.getText());
                    alert.showAndWait();
                }

                // Set current user into init object for later reference
                usernameData.initPlayerData.setUserId(newPlayer.getUserId());
                usernameData.initPlayerData.setPassword(encryptedPassword); // Store
encrypted password
                App.setRoot("LoginScreen");

            } else {
                Alert alert = new Alert(AlertType.ERROR);
                alert.setTitle("Sign up failed!");
                alert.setHeaderText("Please fill out all the required information.");
                alert.showAndWait();
            }

        } catch (Exception e) {
            e.printStackTrace(); // Debug: Print stack trace to diagnose issues
        }
    }

    @FXML
    void switchToLoginScreen(MouseEvent event) throws Exception {
        System.out.println("(RegisterScreen) Login Button Clicked! Switching to Login
Screen.");
        App.setRoot("LoginScreen");
    }
}
```

LoginScreenController.java

```
package oneminute.controllers;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.StringTokenizer;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseEvent;
import oneminute.App;
import oneminute.data.usernameData;

public class LoginScreenController {

    @FXML private Label forgetPasswordLabel;
    @FXML private PasswordField password;
    @FXML private Button playButton;
    @FXML private ImageView playButtonImage;
    @FXML private Label registerLabel;
```

```
    @FXML private TextField userId;

    @FXML
    void logIn(ActionEvent event) {
        boolean found = false;

        try (BufferedReader infoR = new BufferedReader(
                new FileReader("src/main/resources/oneminute/database/Player.txt"))) {
            String line = "";

            while ((line = infoR.readLine()) != null) {
                StringTokenizer infoToken = new StringTokenizer(line, ",");
                String Username = infoToken.nextToken();
                String encryptedPassword = infoToken.nextToken();
                String decryptedPassword = usernameData.decrypt(encryptedPassword);

                // Checking if username & password match any records in database
                if (userId.getText().equals(Username) &&
password.getText().equals(decryptedPassword)) {
                    found = true;
                    usernameData.initPlayerData.setUserId(userId.getText());
                    usernameData.initPlayerData.setPassword(password.getText());
                    System.out.println("(Login Screen) Login successful! Switching to Start
Screen.");
                    App.setRoot("StartScreen");
                    break;
                }
            }

            if (!found) {
                showAlert("Credentials incorrect", "Please enter your credentials
correctly");
            }

        } catch (IOException e) {
            showAlert("Error", "An error occurred while accessing the user database.");
            e.printStackTrace();
        }
    }

    @FXML
    void switchToRegisterScreen(MouseEvent event) throws IOException {
        System.out.println("(Login Screen) Register label clicked! Switching to Register
Screen.");
        App.setRoot("RegisterScreen");
    }

    @FXML
    void switchToResetPasswordScreen(MouseEvent event) throws IOException {
        System.out.println("(Login Screen) Forget Password label clicked! Switching to Reset
Password Screen.");
        App.setRoot("ResetPasswordScreen");
    }

    @FXML
    void switchToStartScreen(ActionEvent event) throws IOException {
        App.getAudioManager().playSoundEffect("buttonClick");
        // Check if userId or password fields are empty
        if (userId.getText().isEmpty() || password.getText().isEmpty()) {
            showAlert("Incomplete Fields", "Please fill out both userId and password.");
            return;
        }else{
            System.out.println("(Login Screen) Play button clicked! Switching to Start
Screen.");
        }

        // Validate if the user is registered before allowing to proceed
        if (isRegistered(userId.getText(), password.getText())) {
            App.setRoot("StartScreen");
        } else {
            showAlert("Registration Required", "Please register before proceeding.");
        }
    }
```

```
    private boolean isRegistered(String username, String password) {
        boolean registered = false;
        try (BufferedReader infoR = new BufferedReader(
                new FileReader("src/main/resources/oneminute/database/Player.txt"))) {
            String line = "";
            while ((line = infoR.readLine()) != null) {
                StringTokenizer infoToken = new StringTokenizer(line, ",");
                String storedUsername = infoToken.nextToken();
                String storedEncryptedPassword = infoToken.nextToken();
                String storedDecryptedPassword =
usernameData.decrypt(storedEncryptedPassword);


                if (username.equals(storedUsername)) {
                    // Check if input password matches the registered password
                    if (!password.equals(storedDecryptedPassword)) {
                        Alert alert = new Alert(Alert.AlertType.ERROR);
                        alert.setTitle("Password Error");
                        alert.setHeaderText("Password does not match.");
                        alert.setContentText("Please enter the correct password.");
                        alert.showAndWait();
                        return false;
                    }
                    return true; // Password matches, user is registered
                }
            }
        } catch (IOException e) {
            showAlert("Error", "An error occurred while accessing the user database.");
            e.printStackTrace();
        }
        return registered;
    }

    private void showAlert(String title, String message) {
        Alert alert = new Alert(AlertType.ERROR);
        alert.setTitle(title);
        alert.setHeaderText(message);
        alert.showAndWait();
    }
}
```

ResetPasswordController.java

```
package oneminute.controllers;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.input.MouseEvent;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import oneminute.App;
import oneminute.classes.Player;
import oneminute.data.usernameData;

public class ResetPasswordScreenController {

    @FXML private PasswordField confirmPassword;
    @FXML private PasswordField newPassword;
    @FXML private TextField userId;
    @FXML private Button resetButton;
    private List<Player> players;

    public void initialize() {
        players = usernameData.loadPlayerDataFromDatabase(); // Load player data from
```

```
database
    }

    @FXML
    void handleIncomplete(ActionEvent event) {
        App.getAudioManager().playSoundEffect("buttonClick");
        showAlert(AlertType.WARNING, "Incomplete Fields", "Please fill in all fields.");
    }

    @FXML
    void handleResetButton(ActionEvent event) throws IOException {
        if (isAnyFieldIncomplete()) {
            App.getAudioManager().playSoundEffect("buttonClick");
            handleIncomplete(event);
        } else {
            validateAndShowDialog();
        }
    }

    private boolean isAnyFieldIncomplete() {
        return userId.getText().isEmpty() || newPassword.getText().isEmpty() ||
confirmPassword.getText().isEmpty();
    }

    private boolean isUserIdValid(String userId) {
        return players.stream().anyMatch(player -> player.getUserId().equals(userId));
    }

    private void validateAndShowDialog() throws IOException {
        String userIdInput = userId.getText();
        String newPwd = newPassword.getText();
        String confirmPwd = confirmPassword.getText();

        if (!isUserIdValid(userIdInput)) {
            showAlert(AlertType.ERROR, "Invalid User ID", "The user ID does not exist in the
database.");
        } else if (newPwd.equals(confirmPwd)) {
            showResetSuccess();
        } else {
            showAlert(AlertType.ERROR, "Password mismatch", "New password and confirm
password do not match.");
        }
    }

    private void showResetSuccess() throws IOException {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Password Reset Successful");
        alert.setHeaderText(null);
        alert.setContentText("Your password has been successfully reset.");
        alert.showAndWait();

        String userIdInput = userId.getText();
        String newPwd = newPassword.getText();
        updatePasswordInDatabase(userIdInput, newPwd);

        switchToLoginScreen(null);
    }

    private void showAlert(AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }

    private void updatePasswordInDatabase(String userId, String newPassword) {
        String encryptedPassword = usernameData.encrypt(newPassword);
        File originalFile = new File("src/main/resources/oneminute/database/Player.txt");
        File tempFile = new File("src/main/resources/oneminute/database/Player_temp.txt");

        try (BufferedReader reader = new BufferedReader(new FileReader(originalFile));
             FileWriter writer = new FileWriter(tempFile);
             PrintWriter printWriter = new PrintWriter(writer)) {
```

```
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split(",");
                String storedUsername = parts[0].trim(); // Trim to handle whitespace
                String storedEncryptedPassword = parts[1].trim(); // Trim to handle
whitespace

                if (storedUsername.equals(userId)) {
                    printWriter.println(userId + "," + encryptedPassword);
                } else {
                    printWriter.println(line);
                }
            }

        } catch (IOException e) {
            showAlert(null, "Error", "An error occurred while updating the password.");
            e.printStackTrace();
        }

        // Delete the original file
        if (!originalFile.delete()) {
            showAlert(null, "Error", "Failed to delete the original file.");
            return;
        }

        // Rename temp file to original file
        if (!tempFile.renameTo(originalFile)) {
            showAlert(null, "Error", "Failed to update the password.");
        } else {
            System.out.println("Password updated successfully.");
        }
    }

    @FXML
    void switchToLoginScreen(MouseEvent event) throws IOException {
        System.out.println("(ResetPasswordScreen) Reset Button Clicked! Switching to Login
Screen. ");
        App.setRoot("LoginScreen");
    }
}
```

StartScreenController.java

```
package oneminute.controllers;


import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import oneminute.App;

public class StartScreenController {
    @FXML
    public void switchToStorylineScreen(ActionEvent event) throws Exception {
        App.getAudioManager().playSoundEffect("buttonStart");
        System.out.println("(Start Screen) Play Button clicked! Switching to Storyline
Screen.");
        App.setRoot("StorylineScreen");
    }
}
```

StorylineController.java

```
package oneminute.controllers;

import java.io.IOException;
import javafx.animation.FadeTransition;
import javafx.animation.PauseTransition;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.util.Duration;
import oneminute.App;
import javafx.event.ActionEvent;
import javafx.scene.control.Label;
```

```
public class StorylineScreenController {
    @FXML private Button continueButton;
    @FXML private Label plotLabel;

    @FXML
    public void initialize() {
        initializeText();
        //set button hidden and disabled
        continueButton.setOpacity(0);
        continueButton.setDisable(true);

        // make transition pause for a few seconds
        PauseTransition pause = new PauseTransition(Duration.seconds(3.8));

        // Set the action after delay
        pause.setOnFinished(event -> {
            //create a fade transition for the button to appear
            FadeTransition fadeIn = new FadeTransition(Duration.seconds(0.8),
continueButton);
            fadeIn.setFromValue(0);
            fadeIn.setToValue(1);

            fadeIn.setOnFinished(e -> continueButton.setDisable(false));

            fadeIn.play();
        });

        pause.play();
    }

    public void initializeText() {
       plotLabel.setOpacity(0);
       plotLabel.setDisable(true);

       PauseTransition pause = new PauseTransition(Duration.seconds(0.8));

        pause.setOnFinished(event -> {
            //create a fade transition for the Label to appear
            FadeTransition fadeIn = new FadeTransition(Duration.seconds(0.8), plotLabel);
            fadeIn.setFromValue(0);
            fadeIn.setToValue(1);

            fadeIn.setOnFinished(e -> plotLabel.setDisable(false));

            fadeIn.play();
        });

        pause.play();
    }

    @FXML
    void switchToStoryline2Screen(ActionEvent event) throws IOException {
        App.getAudioManager().playSoundEffect("buttonTick");
        System.out.println("(StorylineScreen) Continue label clicked! Switching to Storyline
2 Screen.");
        App.setRoot("Storyline2Screen");
    }
}
```

Storyline2Controller.java

```
package oneminute.controllers;

import java.io.IOException;
import javafx.animation.FadeTransition;
import javafx.animation.PauseTransition;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.util.Duration;
import oneminute.App;
import javafx.event.ActionEvent;

public class Storyline2ScreenController {

    @FXML private Button beginLabel;
```

```
    @FXML
    public void initialize() {
        //set button hidden and disabled
        beginLabel.setOpacity(0);
        beginLabel.setDisable(true);

        // make transition pause for a few seconds
        PauseTransition pause = new PauseTransition(Duration.seconds(3.8));

        // Set the action after delay
        pause.setOnFinished(event -> {
            //create a fade transition for the button to appear
            FadeTransition fadeIn = new FadeTransition(Duration.seconds(0.8), beginLabel);
            fadeIn.setFromValue(0);
            fadeIn.setToValue(1);

            fadeIn.setOnFinished(e -> beginLabel.setDisable(false));

            fadeIn.play();
        });

        pause.play();
    }

    @FXML
    void switchToShoppingScreen(ActionEvent event) throws IOException {
        App.getAudioManager().playSoundEffect("buttonStart");
        System.out.println("(Storyline2Screen) Begin label clicked! Switching to Shopping
Screen.");
        App.setRoot("ShoppingScreen");
    }
}
```

ShoppingScreen.java

```
package oneminute.controllers;

import java.util.ArrayList;
import java.util.List;

import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.fxml.FXML;
import javafx.geometry.Point2D;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Tooltip;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.TilePane;
import javafx.util.Duration;
import javafx.scene.input.MouseEvent;
import oneminute.App;
import oneminute.classes.*;
import oneminute.lists.*;

public class ShoppingScreenController {
    private ArrayList<Item> itemList;
    private Inventory inventory = new Inventory();

    private List<Button> selectedItems = new ArrayList<>();
    private List<Point2D> itemPositions = new ArrayList<>();
    private int seconds = 62;

    private Timeline timeline;

    @FXML private TilePane tilePane1;
    @FXML private TilePane tilePane2;
    @FXML private TilePane inventoryTilePane;
    @FXML private Label itemDescription;
    @FXML private Label itemName;
    @FXML private Label timerLabel;
    @FXML private Button button_arrowRight;

    @FXML
```

```
    public void initialize() {
        App.getAudioManager().playBackgroundMusic("shopping");
        App.getAudioManager().setBackgroundMusicVolume("shopping", 0.5);
        try {
            itemList = new ItemList().getItemList(); // Initialize itemList
            loadImagesFromArrayList();
            startTimer();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // Load images into children of TilePane based on itemList
    public void loadImagesFromArrayList() {
        if (itemList == null) {
            System.err.println("Item list is null. Initialization failed.");
            return;
        }

        for (int i = 0; i < 12; i++) {
            if (i < tilePane1.getChildren().size()) {
                Item item = itemList.get(i);
                if (tilePane1.getChildren().get(i) instanceof Button) {
                    Button button = (Button) tilePane1.getChildren().get(i);
                    updateButtonWithImage(button, item);
                }
            }
        }

        for (int i = 12; i < 24; i++) {
            int indexInPane2 = i - 12;
            if (indexInPane2 < tilePane2.getChildren().size()) {
                Item item = itemList.get(i);
                if (tilePane2.getChildren().get(indexInPane2) instanceof Button) {
                    Button button = (Button) tilePane2.getChildren().get(indexInPane2);
                    updateButtonWithImage(button, item);
                }
            }
        }
    }
    private void updateButtonWithImage(Button button, Item item) {
        if (inventoryTilePane.getChildren().size() >= 8) {
            System.out.println("Inventory is full. Cannot add more items.");
            return;
        }
        try {
            Image image = new
Image(getClass().getResource(item.getItemUrl()).toExternalForm());
            ImageView imageView = new ImageView(image);
            imageView.setFitWidth(60);
            imageView.setFitHeight(60);
            button.setGraphic(imageView);
            button.setUserData(item);

            Tooltip tooltip = new Tooltip(item.getDescription());
            Tooltip.install(button, tooltip);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void moveItemToInventory(Button itemButton) {
        TilePane parentPane = (TilePane) itemButton.getParent();
        if (parentPane != null) {
            parentPane.getChildren().remove(itemButton);
        }
        inventoryTilePane.getChildren().add(itemButton);
        Object userData = itemButton.getUserData();
        if (userData instanceof Item) {
            Item item = (Item) userData;
            inventory.addItem(item);
        }
    }
```

```
    @FXML
    private void handleButtonClick(MouseEvent event) {
        Button clickedButton = (Button) event.getSource();
        moveItemToInventory(clickedButton);
    }

    @FXML
    private void handleMouseEnter(MouseEvent event) {
        Button hoveredButton = (Button) event.getSource();
        Item item = (Item) hoveredButton.getUserData();
        if (item != null) {
            itemDescription.setText(item.getDescription());
            itemName.setText(item.getItemname());
        }
    }

    @FXML
    private void handleMouseExit(MouseEvent event) {
        itemDescription.setText("");
        itemName.setText("");
    }

    @FXML
    private void handleItemClick(MouseEvent event) {
        Button clickedButton = (Button) event.getSource();
        if (!selectedItems.contains(clickedButton)) {
            selectedItems.add(clickedButton);
        } else {
            selectedItems.remove(clickedButton);
        }
    }

    // settings for timer
    private void startTimer() {
        timeline = new Timeline(new KeyFrame(Duration.seconds(1), event -> {
            if (seconds > 0) {
                seconds--;
                timerLabel.setText(formatTime(seconds));
            } else {
                switchToCashierScreen();
            }
        }));
        timeline.setCycleCount(Timeline.INDEFINITE);
        timeline.play();
    }

    private String formatTime(int seconds) {
        return String.valueOf(seconds);
    }

    //settings for switching scenes
    @FXML
    public void switchToCashierScreen() {
        App.getAudioManager().playBackgroundMusic("game");
        if (timeline != null) {
            timeline.stop();
        }
        try {
            System.out.println("(ShoppingScreen) Timer ended! Automatically switching to
Cashier Screen.");
            App.setRoot("CashierScreen");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void handleArrowClick(MouseEvent event) {
        App.getAudioManager().playSoundEffect("buttonTick");
        System.out.println("(Shopping Screen) Right Arrow Button Clicked! Switching to
Cashier Screen.");
        switchToCashierScreen();
    }
}
```

CashierScreen.java

```
package oneminute.controllers;

import java.io.IOException;
import java.util.List;
import javafx.geometry.Point2D;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.TilePane;
import oneminute.App;

public class CashierScreenController {

    @FXML private Button pauseButton;
    @FXML private Button startButton;
    @FXML private TilePane cashierinventoryTilePane;

    @FXML
    public void initialize() {
        App.getAudioManager().playSoundEffect("cashierRegister");
    }

    public void setSelectedItems(List<Button> items, List<Point2D> positions) {
        System.out.println("Setting selected items...");

        for (int i = 0; i < items.size(); i++) {
            Button item = items.get(i);
            Point2D position = positions.get(i);
            item.setLayoutX(position.getX());
            item.setLayoutY(position.getY());
            cashierinventoryTilePane.getChildren().add(item);

            System.out.println("Added item at position: " + position);
        }
    }

    @FXML
    void switchToDayCounterScreen(MouseEvent event) throws IOException {
        App.getAudioManager().playSoundEffect("buttonStart");
        System.out.println("(Cashier Screen) Start button clicked! Switching to Winning
Story Screen.");
        App.setRoot("WinningStoryScreen");
    }
}
```

DayCounterScreenController.java

```
package oneminute.controllers;

import javafx.fxml.FXML;
import javafx.scene.input.MouseEvent;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import oneminute.classes.DayCounter;
import oneminute.classes.DayCounterApp;

public class DayCounterController {

    private DayCounter dayCounter;
    private DayCounterApp app;
    private Stage primaryStage;

    @FXML private Text dayText;

    public void setDayCounter(DayCounter dayCounter) {
        this.dayCounter = dayCounter;
        updateDayText();
    }

    public void setApp(DayCounterApp app) {
        this.app = app;
    }

    public void setPrimaryStage(Stage primaryStage) {
```

```
        this.primaryStage = primaryStage;
    }

    public void updateDayText() {
        dayText.setText("DAY " + dayCounter.getCurrentDay());
    }

    @FXML
    void onEventComplete(MouseEvent event) {
        try {
            if (dayCounter != null) {
                dayCounter.nextDay();
            } else {
                System.out.println("DayCounter is null!");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @FXML
    void handleCloseDayCounterButton(MouseEvent event) {
        try {
            if (app != null) {
                app.switchToHomeScreen();
            } else {
                System.out.println("App instance is null!");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

HomeScreenController.java

```
package oneminute.controllers;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.stage.Stage;
import oneminute.classes.DayCounterApp;

public class HomeScreenController {
    private Stage primaryStage;
    private DayCounterApp app;

    public void setPrimaryStage(Stage stage) {
        this.primaryStage = stage;
    }

    public void setApp(DayCounterApp app) {
        this.app = app;
    }

    @FXML
    private void handleJournalButton(ActionEvent event) {
        try {
            if (app != null) {
                app.switchToJournalStoryScreen();
            } else {
                System.out.println("App instance is null!"); // Debug statement
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

JournalStoryScreenController.java

```
package oneminute.controllers;

import javafx.fxml.FXML;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.scene.control.TextArea;
import javafx.scene.control.Button;
import java.util.ArrayList;
import oneminute.classes.*;

public class JournalStoryController {
    private Stage primaryStage;
    private DayCounterApp app;

    public JournalStoryController() {
        System.out.println("JournalStoryController constructor called");
    }

    @FXML private TextArea journalTextArea;
    @FXML private Button journalStoryD1NOButton;
    @FXML private Button journalStoryD1YESButton;
    @FXML private Button goToDayCounterButton;
    @FXML private Button goToWinningSceneButton;

    private ArrayList<String> pagesDay1 = new ArrayList<>();
    private ArrayList<String> pagesDay2 = new ArrayList<>();
    private ArrayList<String> pagesDay3 = new ArrayList<>();
    private ArrayList<String> pagesDay4 = new ArrayList<>();
    private ArrayList<String> pagesDay5 = new ArrayList<>();
    private ArrayList<String> pagesDay6 = new ArrayList<>();
    private ArrayList<String> pagesDay7 = new ArrayList<>();
    private ArrayList<String> pagesDay8 = new ArrayList<>();
    private ArrayList<String> pagesDay9 = new ArrayList<>();
    private ArrayList<String> pagesDay10 = new ArrayList<>();
    private ArrayList<String> pagesDay11 = new ArrayList<>();
    private ArrayList<String> pagesDay12 = new ArrayList<>();
    private ArrayList<String> pagesDay13 = new ArrayList<>();
    private ArrayList<String> pagesDay14 = new ArrayList<>();

    private int currentPageIndex = 0;

    @FXML
    public void initialize() {
        // Example text pages for day 1
        pagesDay1.add("Day 1: Alternative Diet\n" +
                "\n" +
                "\n" +
                "Guess what! We've cultivated our very own indoor garden! We've discovered a
bunch of mushrooms growing on one of the "
                + "walls. They appear to be large enough to consume. Should we consider
having a mushroom meal today?");

        pagesDay1.add("We learned something today. Tomato soup is our best friend, and
mushrooms aren't. Yuck.\n" +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-All family member health (Decrease 10)");

        pagesDay1.add("We may be hungry, but there are just some things we will never eat.
Wall fungus is one of them.\n" +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-All family member hunger bar (Decrease 10)");

         pagesDay2.add("Day 2: Rationing Dilemma\n" +
                "\n" +
                "\n" +
                "With our supplies dwindling faster than expected, we're forced to implement
stricter rationing measures. "
                + "Portion sizes shrink, and each meal becomes a battle of wills against our
```

```
growling stomachs. "
                + "As our hunger grows, so does our desperation to make our remaining
supplies last. "
                + "But we still need to eat, survive, and move on, right?");

        pagesDay2.add("We can't stand the hunger any longer, and we end up eating a meal to
survive and move on.\n" +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-2 Canned Food\n" +
                "+All family member hunger bar (Increase 10)");

        pagesDay2.add("Sorry, We can't stand the hunger any longer, and we end up eating a
meal to survive and move on.\n" +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-2 Canned Food\n" +
                "+All family member hunger bar (Increase 10)");

        pagesDay3.add("Day 3: Coffee Run\n" +
                "\n" +
                "\n" +
                "Robert is tired of drinking plain, tasteless water and is really craving a
cup of coffee. "
                + "He misses the rich aroma and the jolt of energy from his favourite coffee
shop, "
                + "which used to be just a few blocks away. Maybe someone should go on a
coffee run to lift "
                + "everyone's spirits. But who will take the risk? Should we send someone?
");

        pagesDay3.add("Great, we got a whole pot of cold coffee! The taste is odd without
sugar or milk, "
                + "but it's a welcome change from the bland tap water we've been drinking.
The pot only "
                + "held a few cups, which we quickly consumed. Everyone feels more alert and
ready for anything... "
                + "at least for the next five or so minutes." +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "+Everyone drinks coffee\n" +
                "+Everyone feels rested (+ 10 Health)\n" +
                "+Increases everyone's morale (+10 morale)");

        pagesDay3.add("We're not going to risk our health for this. We have more important
things to "
                + "worry about. Those cracks in the walls aren't going to fix themselves!");

        pagesDay4.add("Day 4: Toilet Paper Rationing\n" +
                "\n" +
                "\n" +
                "Our toilet paper supply is dwindling fast, and we're faced with the tough
decision of how to ration "
                + "it effectively. With each passing day, the rolls seem to shrink before
our eyes, and tensions rise as "
                + "we grapple with the reality of our situation. Should we choose to manage
this precious commodity?");


        pagesDay4.add("We decide to implement strict rationing measures, limiting each
family member to "
                + "a specific number of squares per day. While this helps conserve our
supply, tensions "
                + "remain high as we navigate the discomfort of using less than we're
accustomed to." +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-Morale for each family member (-10 Morale)");

        pagesDay4.add("In a moment of desperation, we abandon all attempts at rationing and
use the "
```

```
                + "toilet paper as we please. While this provides temporary relief, we soon
face the consequences "
                + "of our recklessness as our supply dwindles even faster, leaving us in a
precarious situation.!"+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-2 Toilet Paper");

        pagesDay5.add("Day 5: Bread and Circuses\n" +
                "\n" +
                "\n" +
                "Both Robert and Gwen are eyeing one of the canned foods in a very, very
disturbing way. "
                + "Now it looks like they are going to duel over it using forks they found
in the shelter. "
                + "This may get bloody. Shall we stop them?");


        pagesDay5.add("Robert got his fair share of the can. Gwen was watching his every
move. "
                + "We had to step in and put everything in order. Some fair rationing is
what "
                + "this shelter needs if we're going to survive.");

        pagesDay5.add("We decided to watch and enjoy what sick entertainment it was. The can
soon found its "
                + "way to the ground and was spilled all over the floor. They didn't look
too happy about it."+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-1 Canned Food\n" +
                "+Everybody get tired (-5 Morale)");

        pagesDay6.add("Day 6: Medical Emergency\n" +
                "\n" +
                "\n" +
                "We've encountered a medical emergency in the shelter. Susan stumbled and
fell, injuring her leg badly. "
                + "It's swollen, and she's in excruciating pain. We need to act fast. Should
we use the First Aid Kit?");


        pagesDay6.add("Thankfully, we have a well-stocked first aid kit. With careful
attention and the "
                + "supplies from the kit, we manage to stabilize Susan's injury and
alleviate her pain. "
                + "She'll need some time to recover, but it's a relief to see her
comfortable for now." +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-1 First Aid Kit");

        pagesDay6.add("Without the proper medical supplies, we struggle to address Susan's "
                + "injury effectively. Her pain persists, and we're left feeling helpless. "
                + "We can only hope that her condition doesn't worsen without proper
treatment."+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-Overall family morale and health decreases (-5 morale and health)");

        pagesDay7.add("Day 7: Time to Ration Supplies\n" +
                "\n" +
                "\n" +
                "As the days drag on, the strain of rationing our dwindling supplies weighs
heavily on us. "
                + "Each passing day makes it more urgent to carefully portion out our
remaining canned foods "
                + "and water. The stark reality of our situation is undeniable - we must
make every bite count "
                + "if we are to survive until help arrives. It's time to eat.");
```

```
        pagesDay7.add("It's a delicious and satisfying meal, but we still feel hungry. Hope
we still have more food to eat." +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-3 Canned Foods\n" +
                "-3 Water\n" +
                "\n" +
                "For everybody:\n " +
                "Hunger: +30 \n" +
                "Thirst: +30\n" +
                "Morale: +5 \n" +
                "Health: +5 ");

        pagesDay7.add("Sorry, We can't stand the hunger any longer, and we end up eating a
meal to survive and move on."+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-3 Canned Foods\n" +
                "-3 Water\n" +
                "\n" +
                "For everybody:\n" +
                "Hunger: +30 \n" +
                "Thirst: +30\n" +
                "Morale: +5 \n" +
                "Health: +5 ");

        pagesDay8.add("Day 8: Call of the Tuba\n" +
                "\n" +
                "\n" +
                "Susan clearly misses her music lessons - she keeps humming her favourite
classical pieces, imitating the "
                + "sound of tuba. She said she would die for a chance to play it again...
which may very well happen if we "
                + "let her out of the shelter. If this would help to bring a smile to her
weary face, maybe we should consider "
                + "letting her out for a short walk around the house above?");


        pagesDay8.add("Susan got very excited when we agreed to let her look for her tuba.
She said if she can't "
                + "find the instrument, she'll settle for anything else she can find, like
some sheet music. We "
                + "sure hope the tuba didn't make it... we're pretty sure we prefer radio
static or even the sweet "
                + "sound of silence." +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "+Increases everyone’s morale (+10 morale)");

        pagesDay8.add("Risking your life for a tuba? Madness. If it was a guitar or at least
a trumpet…"
                + " but a tuba? Playing the tuba won’t put soup on the table! Susan isn’t
happy, but she "
                + "needs to realize how pointless this idea was. "+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "+Decreases everyone's morale (-10 morale)");

        pagesDay9.add("Day 9: Blackout Emergency\n" +
                "\n" +
                "\n" +
                "A sudden blackout plunges our shelter into darkness, leaving us disoriented
and vulnerable. "
                + "With no electricity to rely on, we're forced to navigate through the
darkness using only our "
                + "flashlight and limited battery supply. How will we manage this unforeseen
crisis? "
                + "Should we use our flashlight and battery?");
```

```
        pagesDay9.add("We quickly gather our flashlight and spare batteries, relying on
their faint glow to"
                + " illuminate our surroundings. While navigating through the darkness is
challenging, the "
                + "flashlight provides a sense of security, allowing us to move cautiously
and avoid potential "
                + "hazards." +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "+10 Morale for each family member\n" +
                "-1 Flashlight\n" +
                "-1 Battery");

        pagesDay9.add("Opting to conserve our battery supply, we choose to stumble through
the darkness without the aid "
                + "of a flashlight. However, our decision proves costly as we struggle to
navigate our surroundings,"
                + " tripping over objects and risking injury in the process."+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-10 Morale for each family member\n" +
                "-10 Health for each family member due to potential injuries");

        pagesDay10.add("Day 10: Danger of the Wasteland\n" +
                "\n" +
                "\n" +
                "We've been hearing a lot of commotion in the neighbourhood today. It seems
like there's "
                + "a group of people looting nearby stores. They're not getting away with
that! This might "
                + "be the perfect opportunity to gather some fresh supplies. Should we send
someone to go out and investigate?");


        pagesDay10.add("We found the group and managed to grab some supplies while they were
distracted. "
                + "We ended up with a few cans of food that will surely help us get through
these tough times." +
                "\n" +
                "\n" +
                "Outcome:\n" +
                "+3 Canned Foods");

        pagesDay10.add("A scavenging mission sounds way too risky. What if the situation
outside is more dangerous than we think? "
                + "It’s better to stay safe inside and avoid unnecessary risks.");


        pagesDay11.add("Day 11: Water Shortage\n" +
                "\n" +
                "\n" +
                "Today brings an unexpected challenge as our water supply unexpectedly runs
dry. With no water for basic hygiene, "
                + "we're left feeling uncomfortable and vulnerable. Should we resort to
using deodorant to mask the lack of bathing?");


        pagesDay11.add("Opting for a quick fix, we liberally apply the deodorant, hoping to
stave off the effects of the water "
                + "shortage on our personal hygiene. While the deodorant offers a temporary
solution, it's no substitute for proper "
                + "bathing. Nevertheless, it provides some relief amidst the discomfort."+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "+10 Morale for each family member\n" +
                "-1 Deodorant");

        pagesDay11.add("Prolonging the discomfort of the water shortage, we endure the day
without resorting to deodorant."
                + " However, as the day wears on, the lack of hygiene takes its toll,
resulting in a decline "
                + "in morale for each family member. The oppressive atmosphere serves as a
```

```
stark reminder of the "
                + "challenges we face in maintaining our well-being amidst scarcity."+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-10 Morale for each family member\n" +
                "-10 All character health");

        pagesDay12.add("Day 12:  Reunited (Infected Camp)\n" +
                "\n" +
                "\n" +
                "A young woman, dressed in protective gear, visited us today, requesting our
help. "
                + "Wasting no time, she explained that her twin brother is trapped in a
nearby camp overrun "
                + "with infected individuals. It seems the young man was foolish enough to
confront the leader "
                + "of a dangerous group hoarding supplies, and now his sister is very
worried. She insists we won't "
                + "regret it if we decide to help her.");


        pagesDay12.add("The woman quickly called a group of friends, and we set out on what
turned out to be a "
                + "successful mission. Finding the lost twin brother wasn't a problem - he
was perched on top of "
                + "a makeshift tower, trying to avoid the infected. In his confrontation
with the camp leader, he "
                + "suffered a serious injury, but the group quickly tended to his wounds,
and he'll recover. The "
                + "siblings thanked us profusely and provided us with some of their supplies
as a token of their "
                + "gratitude."+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "+3 Canned Foods\n" +
                "+1 First Aid Kit\n" +
                "+Boost to Morale ( +10 morale )");

        pagesDay12.add("Heading deep into an infected camp, looking for a confrontation?
These people seem "
                + "well-meaning, but we just can't take the risk right now. Perhaps another
time.");

        pagesDay13.add("Day 13: Pest Infestation \n" +
                "\n" +
                "\n" +
                "An unpleasant discovery awaits us as we find our shelter invaded by a horde
of pesky insects. "
                + "These critters threaten our hygiene and comfort, leaving us with no
choice but to address the "
                + "infestation. Should we use bug spray to tackle this nuisance head-on?");


        pagesDay13.add("With determination, we unleash the bug spray, filling the air with
its potent scent. "
                + "The insects scatter and retreat, granting us temporary relief from the
infestation. While "
                + "the smell lingers, it's a small price to pay for reclaiming our shelter
from these unwanted "
                + "guests."+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "+5 Morale for each family member\n" +
                "-1 Bug spray");

        pagesDay13.add("I'm sorry but we don't have bug spray at the moment. Without bug
spray, we attempt to endure the "
                + "insect invasion without intervention. "
                + "However, the pests continue to proliferate, posing a persistent threat to
our hygiene and well-being. "
                + "Our reluctance to use the bug spray results in prolonged discomfort and
frustration."+
```

```
                "\n" +
                "\n" +
                "Outcome:\n" +
                "-10 Morale for each family member\n" +
                "-10 All character health");

        pagesDay14.add("Day 14: The Quarantine Cookie Stand\n" +
                "\n" +
                "\n" +
                "As we cautiously approached the door upon hearing a knock, we were relieved
to find a pair "
                + "of girls' voices outside. It was the local Girl Scouts, adapting their
cookie-selling business to"
                + " these challenging times. Resourceful kids. Should we accept their
trade?");


        pagesDay14.add("Impressed by the Girl Scouts' innovative spirit during the pandemic,
we gladly accepted their offer. "
                + "Their fair prices and array of items provided a much-needed boost to our
supplies."+
                "\n" +
                "\n" +
                "Outcome:\n" +
                "+5 Canned Foods\n" +
                "+1 Medical Kit\n" +
                "-3 Water");

        pagesDay14.add("While we admired the Girl Scouts' courage in navigating the current
situation, "
                + "their offerings didn't align with our immediate needs. Politely, we
declined, albeit missing "
                + "out on potential supplies.");

        // Display the first page initially
        goToDayCounterButton.setVisible(false);
        goToWinningSceneButton.setVisible(false);
    }

    public void setPrimaryStage(Stage stage) {
        this.primaryStage = stage;
        System.out.println("PrimaryStage in JournalStoryController: " + primaryStage); //
Debug statement
    }

    public void setApp(DayCounterApp app) {
        this.app = app;
        System.out.println("App instance set in JournalStoryController: " + app); // Debug
statement
        updateTextArea();
    }

    @FXML
    private void handleCloseJournalStoryButton(ActionEvent event) {
        try {
            if (app != null) {
                app.switchToHomeScreen();
            } else {
                System.out.println("App instance is null!"); // Debug statement
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void handleJournalStatusButton(ActionEvent event) {
        try {
            if (app != null) {
                app.switchToJournalStatusScreen();
            } else {
                System.out.println("App instance is null!"); // Debug statement
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
    }
}

@FXML
private void sayNo(ActionEvent event) {
    if (app != null) {
        int currentDay = app.getDayCounter().getCurrentDay();
        switch (currentDay) {
            case 1 -> {
                // Display the text for the 'No' option for day 1
                journalTextArea.setText(pagesDay1.get(2));
                // Decrease all family member hunger
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHunger(member.getHunger() - 10);
                }
                updateFamilyMemberStatus();
            }
            case 2 -> {// Display the text for the 'No' option for day 2
                journalTextArea.setText(pagesDay2.get(2));
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHunger(member.getHunger() + 10);
                }
                updateFamilyMemberStatus();
            }
            case 3 ->{ // Display the text for the 'No' option for day 3
                journalTextArea.setText(pagesDay3.get(2));
                updateFamilyMemberStatus();
            }
            case 4 ->{ // Display the text for the 'No' option for day 4
                journalTextArea.setText(pagesDay4.get(2));
                updateFamilyMemberStatus();
            }
            case 5 ->{ // Display the text for the 'No' option for day 5
                journalTextArea.setText(pagesDay5.get(2));
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setMorale(member.getMorale() - 5);
                }
                updateFamilyMemberStatus();
            }
            case 6 ->{ // Display the text for the 'No' option for day 6
                journalTextArea.setText(pagesDay6.get(2));
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setMorale(member.getMorale() - 5);
                }
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHealth(member.getHealth() - 5);
                }
                updateFamilyMemberStatus();
            }
            case 7 ->{ // Display the text for the 'No' option for day 7
                journalTextArea.setText(pagesDay7.get(2));
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setMorale(member.getMorale() + 5);
                }
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHealth(member.getHealth() + 5);
                }
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setThirst(member.getThirst() + 30);
                }
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHunger(member.getHunger() + 30);
                }
                updateFamilyMemberStatus();
            }
            case 8 ->{ // Display the text for the 'No' option for day 8
                journalTextArea.setText(pagesDay8.get(2));
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setMorale(member.getMorale() - 10);
                }
                updateFamilyMemberStatus();
            }
            case 9 ->{ // Display the text for the 'No' option for day 9
                journalTextArea.setText(pagesDay9.get(2));
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
```

```
                    member.setMorale(member.getMorale() - 10);
                }
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHealth(member.getHealth() - 10);
                }
                updateFamilyMemberStatus();
            }
            case 10 ->{ // Display the text for the 'No' option for day 10
                journalTextArea.setText(pagesDay10.get(2));
                updateFamilyMemberStatus();
            }
            case 11 ->{ // Display the text for the 'No' option for day 11
                journalTextArea.setText(pagesDay11.get(2));
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setMorale(member.getMorale() - 10);
                }
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHealth(member.getHealth() - 10);
                }
                updateFamilyMemberStatus();
            }
            case 12 ->{ // Display the text for the 'No' option for day 12
                journalTextArea.setText(pagesDay12.get(2));
                updateFamilyMemberStatus();
            }
            case 13 ->{ // Display the text for the 'No' option for day 13
                journalTextArea.setText(pagesDay13.get(2));
                 for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setMorale(member.getMorale() - 10);
                }
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHealth(member.getHealth() - 10);
                }
                updateFamilyMemberStatus();
            }
            case 14 ->{ // Display the text for the 'No' option for day 14
                journalTextArea.setText(pagesDay14.get(2));
                updateFamilyMemberStatus();
                goToWinningSceneButton.setVisible(true);
                journalStoryD1NOButton.setDisable(true);
                journalStoryD1YESButton.setDisable(true);
                goToDayCounterButton.setDisable(true);

            }

        }
    }
    journalStoryD1NOButton.setVisible(false);
    journalStoryD1YESButton.setVisible(false);
    goToDayCounterButton.setVisible(true);
}

@FXML
private void sayYes(ActionEvent event) {
    if (app != null) {
        int currentDay = app.getDayCounter().getCurrentDay();
        switch (currentDay) {
            case 1 -> {
                // Display the text for the 'Yes' option for day 1
                journalTextArea.setText(pagesDay1.get(1));
                // Decrease all family member health
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHealth(member.getHealth() - 10);
                }
                updateFamilyMemberStatus();
            }
              case 2 ->{ // Display the text for the 'Yes' option for day 2
                journalTextArea.setText(pagesDay2.get(1));
                for (FamilyMemberStatus member : app.getFamilyMembers()) {
                    member.setHunger(member.getHunger() + 10);
                }
                updateFamilyMemberStatus();
            }
            case 3 ->{ // Display the text for the 'Yes' option for day 3
                journalTextArea.setText(pagesDay3.get(1));
```

```
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setHealth(member.getHealth() + 10);
            }
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setMorale(member.getMorale() + 10);
            }
            updateFamilyMemberStatus();
        }
        case 4 ->{ // Display the text for the 'Yes' option for day 4
            journalTextArea.setText(pagesDay4.get(1));
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setMorale(member.getMorale() - 10);
            }
            updateFamilyMemberStatus();
        }
        case 5 ->{ // Display the text for the 'Yes' option for day 5
            journalTextArea.setText(pagesDay5.get(1));
            updateFamilyMemberStatus();
        }
        case 6 ->{ // Display the text for the 'Yes' option for day 6
            journalTextArea.setText(pagesDay6.get(1));
            updateFamilyMemberStatus();
        }
        case 7 ->{ // Display the text for the 'Yes' option for day 7
            journalTextArea.setText(pagesDay7.get(1));
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setMorale(member.getMorale() + 5);
            }
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setHealth(member.getHealth() + 5);
            }
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setThirst(member.getThirst() + 30);
            }
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setHunger(member.getHunger() + 30);
            }
            updateFamilyMemberStatus();
        }
        case 8 ->{ // Display the text for the 'Yes' option for day 8
            journalTextArea.setText(pagesDay8.get(1));
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setMorale(member.getMorale() + 10);
            }
            updateFamilyMemberStatus();
        }
        case 9 ->{ // Display the text for the 'Yes' option for day 9
            journalTextArea.setText(pagesDay9.get(1));
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setMorale(member.getMorale() + 10);
            }
            updateFamilyMemberStatus();
        }
        case 10 ->{ // Display the text for the 'Yes' option for day 10
            journalTextArea.setText(pagesDay10.get(1));
            updateFamilyMemberStatus();
        }
        case 11 ->{ // Display the text for the 'Yes' option for day 11
            journalTextArea.setText(pagesDay11.get(1));
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setMorale(member.getMorale() + 10);
            }
            updateFamilyMemberStatus();
        }
        case 12 ->{ // Display the text for the 'Yes' option for day 12
            journalTextArea.setText(pagesDay12.get(1));
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setMorale(member.getMorale() + 10);
            }
            updateFamilyMemberStatus();
        }
        case 13 ->{ // Display the text for the 'Yes' option for day 13
            journalTextArea.setText(pagesDay13.get(1));
            for (FamilyMemberStatus member : app.getFamilyMembers()) {
                member.setMorale(member.getMorale() + 5);
```

```
                    }
                    updateFamilyMemberStatus();
                }
                case 14 ->{ // Display the text for the 'Yes' option for day 14
                    journalTextArea.setText(pagesDay14.get(1));
                    updateFamilyMemberStatus();
                    goToWinningSceneButton.setVisible(true);
                    journalStoryD1NOButton.setDisable(true);
                    journalStoryD1YESButton.setDisable(true);
                    goToDayCounterButton.setDisable(true);
                }
            }
        }
        journalStoryD1NOButton.setVisible(false);
        journalStoryD1YESButton.setVisible(false);
        goToDayCounterButton.setVisible(true);
    }
    @FXML
    private void goToDayCounter(ActionEvent event) {
        try {
            if (app != null) {
                app.switchToDayCounterScreen();
            } else {
                System.out.println("App instance is null!"); // Debug statement
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

        public void handleGoToWinningSceneButton(ActionEvent event) {
        try {
            if (app != null) {
                app.switchToDayCounterScreen();
            } else {
                System.out.println("App instance is null!"); // Debug statement
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    private void updateTextArea() {
        if (app != null && journalTextArea != null) {
            int currentDay = app.getDayCounter().getCurrentDay();
            switch (currentDay) {
                case 1 -> journalTextArea.setText(pagesDay1.get(currentPageIndex));
                case 2 -> journalTextArea.setText(pagesDay2.get(0));
                case 3 -> journalTextArea.setText(pagesDay3.get(0));
                case 4 -> journalTextArea.setText(pagesDay4.get(0));
                case 5 -> journalTextArea.setText(pagesDay5.get(0));
                case 6 -> journalTextArea.setText(pagesDay6.get(0));
                case 7 -> journalTextArea.setText(pagesDay7.get(0));
                case 8 -> journalTextArea.setText(pagesDay8.get(0));
                case 9 -> journalTextArea.setText(pagesDay9.get(0));
                case 10 -> journalTextArea.setText(pagesDay10.get(0));
                case 11 -> journalTextArea.setText(pagesDay11.get(0));
                case 12 -> journalTextArea.setText(pagesDay12.get(0));
                case 13 -> journalTextArea.setText(pagesDay13.get(0));
                case 14 -> journalTextArea.setText(pagesDay14.get(0));

            }
        } else {
            System.out.println("App instance or journalTextArea is null!"); // Debug
statement
        }
    }

    private void updateFamilyMemberStatus() {
        if (app != null && journalTextArea != null) {
            String status = app.getFormattedFamilyMembersStatus();
            journalTextArea.appendText("\n\n" + status); // Append the status at the bottom
        }
    }
```

```
}
```

JournalStatusScreenController.java

```
package oneminute.controllers;

import javafx.fxml.FXML;
import javafx.stage.Stage;
import javafx.scene.image.ImageView;
import javafx.event.ActionEvent;
import java.util.HashMap;
import java.util.Map;
import oneminute.classes.*;

public class JournalStatusController {

    private DayCounterApp app;
    private Stage primaryStage;
    private Map<String, FamilyMemberStatus> familyMemberMap = new HashMap<>();

    @FXML private ImageView susanHealthBar;
    @FXML private ImageView susanHungerBar;
    @FXML private ImageView susanThirstBar;
    @FXML private ImageView susanMoraleBar;
    @FXML private ImageView robertHealthBar;
    @FXML private ImageView robertHungerBar;
    @FXML private ImageView robertThirstBar;
    @FXML private ImageView robertMoraleBar;
    @FXML private ImageView gwenHealthBar;
    @FXML private ImageView gwenHungerBar;
    @FXML private ImageView gwenThirstBar;
    @FXML private ImageView gwenMoraleBar;

    public void setPrimaryStage(Stage stage) {
        this.primaryStage = stage;
        System.out.println("PrimaryStage in JournalStatusController: " + primaryStage); //
Debug statement
    }

    public void setApp(DayCounterApp app) {
        this.app = app;
        for (FamilyMemberStatus member : app.getFamilyMembers()) {
            familyMemberMap.put(member.getName(), member);
        }
        updateStatusBars();
    }

    @FXML
    private void handleCloseJournalStatusButton(ActionEvent event) {
        try {
            if (app != null) {
                app.switchToHomeScreen();
            } else {
                System.out.println("App instance is null!"); // Debug statement
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void handleJournalStoryButton(ActionEvent event) {
        try {
            if (app != null) {
                app.switchToJournalStoryScreen();
            } else {
                System.out.println("App instance is null!"); // Debug statement
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void updateStatusBars() {
        updateStatusBar(susanHealthBar, familyMemberMap.get("Susan").getHealth());
```

```
        updateStatusBar(susanHungerBar, familyMemberMap.get("Susan").getHunger());
        updateStatusBar(susanThirstBar, familyMemberMap.get("Susan").getThirst());
        updateStatusBar(susanMoraleBar, familyMemberMap.get("Susan").getMorale());

        updateStatusBar(robertHealthBar, familyMemberMap.get("Robert").getHealth());
        updateStatusBar(robertHungerBar, familyMemberMap.get("Robert").getHunger());
        updateStatusBar(robertThirstBar, familyMemberMap.get("Robert").getThirst());
        updateStatusBar(robertMoraleBar, familyMemberMap.get("Robert").getMorale());

        updateStatusBar(gwenHealthBar, familyMemberMap.get("Gwen").getHealth());
        updateStatusBar(gwenHungerBar, familyMemberMap.get("Gwen").getHunger());
        updateStatusBar(gwenThirstBar, familyMemberMap.get("Gwen").getThirst());
        updateStatusBar(gwenMoraleBar, familyMemberMap.get("Gwen").getMorale());
    }

    private void updateStatusBar(ImageView bar, int value) {

    }
}
```

PauseScreenController.java

```
package oneminute.controllers;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.Optional;
import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.input.MouseEvent;
import oneminute.App;
import oneminute.data.*;

public class PauseScreenController {

    @FXML private Button quitButton;
    @FXML private Button continueButton;
    @FXML private Button saveButton;


    @FXML
    private void toQuit(MouseEvent event) {
        // Get the current stage and close it
        System.out.println("quit button clicked");
        boolean confirmQuit = showConfirmationAlert("Quit Game", "Are you sure you want to
quit the game?"); // for yes or no alert box to pop up

        if (confirmQuit) {
            System.out.println ("User confirmed quit. Exiting application");
            Platform.exit();
        } else {
            System.out.println ("User canceled quit. Application continues");
        }
    }

   @FXML
    private void toContinue(MouseEvent event) throws Exception{
        System.out.println("(PauseScreen) Continue Button clicked! Switching to Home
Screen.");
        App.setRoot("HomeScreen");
    }

    // Method to handle Save Button
    @FXML
    private void toSave (MouseEvent event) {

        ArrayList <String> inventoryItems = new ArrayList <>(); // making a new arraylist
```

```
with inventory
        inventoryItems.add("Water Bottle");
        inventoryItems.add("Canned Food");
        inventoryItems.add("First Aid Kit");

        playerData playerData = new playerData("player123", 2
,"Healthy","Healthy","Healthy", inventoryItems); // creating playerData object

        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("GameData.bin"))) { // to save the file
            oos.writeObject(playerData);
            showInformationAlert("Data Saved", "Your player data has been saved
successfully"); //window to pop up saying that your data has been saved
        } catch (IOException e) {
            e.printStackTrace();
        }

        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("GameData.bin"))) { // to read the file
            playerData readPlayerData = (playerData) ois.readObject();
            System.out.println(readPlayerData);
            System.out.println("playerUID = " + readPlayerData.getPlayerUID());

        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    //Yes or No Alert Box
     private boolean showConfirmationAlert(String title, String message) {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);

        ButtonType yesButton = new ButtonType ("Yes");
        ButtonType noButton = new ButtonType ("No");

        alert.getButtonTypes().setAll(yesButton, noButton);
        Optional<ButtonType> result = alert.showAndWait();

        return result.isPresent() && result.get() == yesButton;
    }

    // Information Alert Box
    private void showInformationAlert(String title, String message) {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```

WinningStoryScreenController.java

```
package oneminute.controllers;

import java.io.IOException;
import javafx.animation.FadeTransition;
import javafx.animation.PauseTransition;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.util.Duration;
import oneminute.App;

public class WinningStoryScreenController {

    @FXML private Button continueButton;
    @FXML private Label winningStoryText;

    @FXML
    public void initialize() {
```

```
        //set button hidden and disabled
        continueButton.setOpacity(0);
        continueButton.setDisable(true);

        // make transition pause for a few seconds
        PauseTransition pause = new PauseTransition(Duration.seconds(3.8));

        // Set the action after delay
        pause.setOnFinished(event -> {
            //create a fade transition for the button to appear
            FadeTransition fadeIn = new FadeTransition(Duration.seconds(0.8),
continueButton);
            fadeIn.setFromValue(0);
            fadeIn.setToValue(1);

            fadeIn.setOnFinished(e -> continueButton.setDisable(false));

            fadeIn.play();
        });

        pause.play();
    }

    public void initializeText() {
        winningStoryText.setOpacity(0);
        winningStoryText.setDisable(true);

        PauseTransition pause = new PauseTransition(Duration.seconds(0.5));

         pause.setOnFinished(event -> {
             //create a fade transition for the Label to appear
             FadeTransition fadeIn = new FadeTransition(Duration.seconds(0.8),
winningStoryText);
             fadeIn.setFromValue(0);
             fadeIn.setToValue(1);

             fadeIn.setOnFinished(e -> winningStoryText.setDisable(false));

             fadeIn.play();
         });
         pause.play();
     }

    @FXML
    void switchToWinningScreen(ActionEvent event) throws IOException {
        App.getAudioManager().playSoundEffect("buttonTick");
        System.out.println("Continue label clicked! Switching to Winning Screen.");
        App.setRoot("WinningScreen");
    }
}
```

WinningScreenController.java

```
package oneminute.controllers;

import java.io.IOException;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import oneminute.App;

public class WinningScreenController {

    @FXML
    private Button mainMenuButton;

    @FXML
    void switchToStartScreen(ActionEvent event) throws IOException {
        App.getAudioManager().playSoundEffect("buttonTick");
        System.out.println("Main Menu Button clicked! Switching to Start Screen.");
        App.setRoot("StartScreen");
    }
}
```

DefeatScreenController.java

```java
package oneminute.controllers;

import java.io.IOException;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import oneminute.App;
public class DefeatScreenController {

    @FXML private Button retryButton;
    @FXML private Button quitButton;

    @FXML
    void switchToStartScreen(ActionEvent event) throws IOException {
        App.getAudioManager().playSoundEffect("buttonTick");
        System.out.println("Retry Button clicked! Switching to Start Screen.");
        App.setRoot("StartScreen");
    }

    @FXML
    void exitGame(ActionEvent event) {
        App.getAudioManager().playSoundEffect("buttonTick");
        System.out.println("Quit button clicked! Closing the game.");
        Platform.exit();
    }
}
```