**1. Requirements Specification**

* **Overview of the Project**:
  + Create an eco-friendly habit tracker app that allows users to input the frequency of their eco-friendly actions, calculates the impact, and displays feedback with tips for improvement.
* **Functional Requirements**:
  + **User Input**: Ability for users to enter the number of times per week they perform eco-friendly habits.
  + **Habit Display**: A list of at least five eco-friendly habits, each with matching images and labels.
  + **Calculations**: Perform calculations to show the user's impact on the environment and potential cost savings.
  + **Feedback Generation**: Display personalized feedback and hints based on the user input.
  + **File Generation**: Save the feedback message to a text file for the user to print or share.
* **Non-Functional Requirements**:
  + **Usability**: The application should be easy to navigate and user-friendly.
  + **Performance**: The app should run smoothly without significant delays.
  + **Reliability**: The app should handle incorrect or unexpected input gracefully.
  + **Security**: The application should not have vulnerabilities that could compromise user data.
* **User Stories**:
  + *As a user*, I want to input my weekly eco-habit frequency so that I can see how my actions contribute to sustainability.
  + *As a user*, I want to see a summary of my weekly habits with estimated benefits so that I feel motivated to continue.
  + *As a user*, I want to save my feedback as a text file so I can review it later.

**2. Design Specification**

* **Pseudocode for the Program Algorithm**:

plaintext

Copy code

Start

Display the main interface with input fields for each eco-habit.

Get user input for each habit.

Calculate the environmental impact and cost savings based on user input.

Generate a personalized feedback message.

Display the feedback message on the screen.

Save the feedback message to a text file.

End

* **Wireframes**:
  + **Wireframe 1**: Main input page with text fields and images for each eco-habit.
  + **Wireframe 2**: Feedback page showing the results of calculations and a button to save the feedback.
  + **Wireframe 3**: Confirmation and instructions page (e.g., "Your feedback has been saved to a file").

**3. Completed Code Solution**

* Develop the app using Java with a graphical user interface (e.g., JavaFX or Swing).
* Include detailed comments in the code and make sure it adheres to best coding practices.
* Package the project as a JAR file.
* Ensure the code solution includes:
  + Java classes for managing user input and calculations.
  + A main GUI class for building the user interface.
  + A file-handling class for saving the feedback to a text file.

**4. Completed Test Log**

* **Test Plan**: Outline the testing strategy (e.g., functional testing, boundary testing, exception handling).
* **Test Cases**:
  + *Case 1*: Verify that entering valid data and clicking "Submit" displays correct feedback.
  + *Case 2*: Verify handling of zero input (boundary case).
  + *Case 3*: Test invalid input (e.g., negative numbers or non-integer values) and ensure proper handling.
* **Test Data**:
  + *Normal*: Regular input for weekly eco-habits (e.g., 3 times per week).
  + *Boundary*: Edge cases like 0 or 7 days per week.
  + *Exceptional*: Invalid inputs such as text or negative numbers.

**5. Deployment Instructions**

* **Platform Requirements**:
  + Java Runtime Environment (JRE) version 11 or above.
  + Ensure that JavaFX or Swing libraries are included in the project.
* **Instructions on Installation**:
  + Download the JAR file and place it in a folder.
  + Open a terminal or command prompt and navigate to the folder.
  + Run the command: java -jar EcoHabitTracker.jar.
  + Follow the on-screen instructions to start using the application.

**Checklist for Submission:**

* **Requirements Specification**: Completed with overview, requirements, and user stories.
* **Design Specification**: Pseudocode and wireframes provided.
* **Completed Code Solution**: Fully developed and commented code, packaged as a JAR file.
* **Test Log**: Includes a test plan, test cases, and test data.
* **Deployment Instructions**: Clear steps for platform setup and app installation.

**Final Notes:**

Ensure all documentation is professional and neatly organized with a cover page that includes:

* **Name**
* **Date**
* **Assessment Task**