△ Homework 01: 👣 The Carbon Footprint Tracker

How To Guides

3/21/2025

Attempt 1

In Progress
NEXT UP: Submit Assignment

Add Comment

3/3/2025 to 3/23/2025

Details

Overview

In a world increasingly aware of its environmental impact, The Architect, is a forward-thinking designer passionate about creating sustainable spaces. Inspired by their commitment to eco-friendly living, you are tasked with designing an application that empowers individuals to measure and reduce their carbon footprint.

The Carbon Footprint Tracker is designed to be a personal environmental assistant. It allows users to log daily activities, each assigned a carbon value, and offers insights into lifestyle changes. By incorporating interactive elements, you can help users stay engaged and motivated.

The challenge is not just technical but philosophical: *how can design influence behavior for the better?* With The Architect's inspiration, you will explore how technology can foster a greener future, turning everyday actions into meaningful environmental contributions.

How To Guides

Objective: The Carbon Footprint Tracker assignment aims to deepen students' understa
by building a practical, eco-conscious application. This assignment will help you apply key React concepts, such as state management and context, while fostering an appreciation for sustainable living.

Learning Objectives

By completing this assignment, students will:

1. Gain hands-on experience with React hooks for state management.
2. Understand the use of React context for managing global state.
3. Learn to effectively use props for passing data between components.
4. Develop user interfaces that enhance user experience through dynamic updates.

Features

For this assignment, you have the flexibility to choose any 5 of the following 10 features to implement:

1. **Activity Logging**: Create a feature that allows users to log daily activities, categorizing them by type. Each activity will have an associated carbon value, helping users understand their environmental impact.

2. **Summary Views**: Design a dashboard that provides a visual summary of the user's carbon footprint over time, using charts or graphs to illustrate trends.

3. **Goal Setting**: Implement a goal-setting feature where users can set personal targets for reducing their carbon footprint, with progress tracking and motivational feedback.

4. **Notifications**: Add optional notifications to remind users to log activities, helping them stay consistent in their tracking.

5. **Resource Library**: Develop a section with practical tips and resources for sustainable living, encouraging users to adopt eco-friendly habits.

6. **Achievements**: Introduce gamification by creating achievements for reaching milestones, making the experience engaging and rewarding.

7. **Local Storage**: Ensure that all data is stored locally in the browser, maintaining user privacy and enabling offline access.

8. **Customizable UI**: Allow users to personalize the app's interface, making it visually appealing and user-friendly.

9. **Data Visualization**: Design interactive charts and graphs to help users visualize their carbon footprint and identify areas for improvement.

10. **User Profiles**: Implement a profile feature that allows users to track their progress and personalize their experience.

Choose any combination of these features to complete the assignment while demonstra How To Guides understanding of React!

Note: you do not need to have the application fully functional in all aspects. The goal is to complete just 5 features. You are welcome to complete additional features to further your grasp of a scalable front-end framework such as React.

Requirements

To complete this assignment, your implementation must meet the following technical criteria:

1. **React Framework** – The application must be built using React and structured with functional components.
2. **State Management** – Use React's `useState` and `useReducer` hooks for managing component state.
3. **Context API** – If your application has shared state across components, use React Context instead of prop drilling.
4. **Effect Handling** – Use the `useEffect` hook to handle side effects such as logging data changes, updating the UI in response to state changes, or setting up timers.
5. **Routing (Optional)** – If your design includes multiple views, use React Router (https://reactrouter.com) to manage navigation. Note, we did not cover React Router in class, however, it is a way to allow for multiple views in your UI. We provide this as optional, but highly recommend that you research this library and attempt to apply it.
6. **Event Handling** – Implement event listeners and handlers to create an interactive experience.
7. **In-Memory Data Management** – All application data should be stored in memory. No external APIs, databases, or local storage (`localStorage` or `sessionStorage`) should be used unless you choose to implement local storage as part of your feature choice.
8. **Responsive Design** – Ensure that the UI is accessible and adapts to different screen sizes using CSS or a utility framework like Tailwind CSS.
9. **Component Reusability** – Break down the UI into reusable components to promote modularity and maintainability.
10. **Error Handling** – Implement basic error handling to ensure a smooth user experience.
11. **Code Readability** – Write clean, well-commented code following React best practices.

Submission Instructions

Students must submit their completed project to **Gradescope** by the deadline. The submission should include:

1. **A ZIP File of the Project** – The entire React application should be compressed into a `.zip` file and uploaded to Gradescope. Ensure that all necessary files are included, but **omit** `node_modules`.
2. **A README File** – Include a `README.md` file that briefly describes the project, the features implemented, and any setup instructions needed to run it. The setup instructions must allow the

reviewer of your submission to easily install and run your app. In general, we expect `npm install` and an `npm run dev` will run your app.
3. **A Video Demo (Optional)** – You can record a short video of the application working and supply at link to the video in the README. This is optional, but recommended.

How To Guides

Late submissions will follow the standard course late policy. Please see the syllabus (https://umamherst.instructure.com/courses/25230/pages/compsci-426-scalable-web-systems-syllabus) for additional details.

Evaluation Criteria and Peer Review Rubric

This assignment will be **peer-reviewed**, meaning that each student will be responsible for reviewing and evaluating a set of submissions from other students using a provided structured rubric.

Why Peer Review?

Code review is an essential skill in software development, whether in industry, open-source projects, or academic settings. Being able to critically evaluate another developer's code helps improve your own coding style, deepens your understanding of best practices, and enhances your ability to communicate technical feedback.

How Will It Work?

1. You will complete this assignment and submit it to Gradescope by the assigned due date.
2. After the late day window, no additional submissions will be accepted and the peer review process will begin.
3. We will distribute to you a zip file containing three anonymized submissions from other students.
4. Detailed instructions on how to record your evaluation and feedback will be provided to you.
5. You will submit your reviews to Gradescope.
6. We will evaluate your peer review with three other peer reviews to determine:
    1. If the score you recorded for a submission you reviewed is "near" scores peer reviewed by other students for the same submission.
    2. We will evaluate your feedback to compute a quality score. High-quality feedback receives a higher score.
7. A grade will be assigned to you based on the peer-reviews from other students and the reviews you made to the three submissions.
8. The grade and feedback for your submission will be distributed back to you.

Note: this is the first time I have ever done this, so I am hoping that you will join me in this experiment. :-)

Peer Evaluation Rubric (Total: 90 Points)

Each submission is evaluated based on the following criteria.

Submission Received

Total Points: 10

1. You automatically get 10 points if you submit your homework to Gradescope.
2. The auto-grader checks your submission to ensure that you did not submit with a **node_modules** folder.
3. It doesn't perform any additional auto-grading functions.

Submission Review Criteria

Total Points: 60

1. React Concepts and Functionality (28 Points)

- **Component Structure (7 Points)** – Functional components are used correctly and modularly.
- **State & Context (7 Points)** – `useState` / `useReducer` and Context API are used correctly.
- **Effects & Events (7 Points)** – `useEffect` is used appropriately, and event handling is implemented correctly.
- **Data Management (7 Points)** – Data is stored in memory or local storage, with no external APIs or databases.

2. Code Quality & Organization (21 Points)

- **Readability & Maintainability (7 Points)** – Code is well-structured, readable, and follows best practices.
- **Error Handling (7 Points)** – Application handles errors gracefully and avoids crashes.
- **Reusability & Efficiency (7 Points)** – Components are reusable, and the code avoids redundancy.

3. User Experience & Design (11 Points)

- **Responsive & Accessible UI (6 Points)** – The UI adapts well to different screen sizes and is easy to use.
- **Intuitive Functionality (5 Points)** – The application is easy to navigate and behaves as expected.

Use the following generalize rubrics to grade each individual criteria item according to point value:

Generalized 7-Point Rubric

- **7 Points** – Exceeds expectations; goes beyond requirements with extra effort, optimizations, or outstanding implementation.
- **5 Points** – Meets expectations fully with a correct and complete implementation.
- **3 Points** – Partially meets expectations but is incomplete, incorrect, or missing key aspects.
- **0 Points** – Not attempted or completely non-functional.

Generalized 6-Point Rubric

- **6 Points** – Exceeds expectations; goes beyond requirements with extra effort, optimizations, or outstanding implementation.

- **4 Points** – Meets expectations fully with a correct and complete implementation.
- **2 Points** – Partially meets expectations but is incomplete, incorrect, or missing key aspects.
- **0 Points** – Not attempted or completely non-functional.

How To Guides

Generalized 5-Point Rubric

- **5 Points** – Exceeds expectations; goes beyond requirements with extra effort, optimizations, or outstanding implementation.
- **3 Points** – Meets expectations fully with a correct and complete implementation.
- **1 Points** – Partially meets expectations but is incomplete, incorrect, or missing key aspects.
- **0 Points** – Not attempted or completely non-functional.

Peer Review Criteria

Total Points: 30

1. Peer Review Quality (30 Points)

- **You will peer review exactly 3 submissions from other students.**
- **You will review each submission according to the criteria above.**
- **Your peer review "quality" will be evaluated after you submit your peer reviews.**
- **Your peer review "quality" score will be included in your overall grade for this assignment.**
- **For each submission your review, you will be scored in terms of the accuracy of your grade as compared to other reviewers as well as the quality of the feedback you provide.**
  - **Grade Accuracy (3 points):** The grade you assign to the submission is within a near distance of other reviewer scores.
  - **Feedback (7 points)**: Your feedback is thoughtful, specific, and constructive.
- **TotalPoints:** 3 submissions x 10 = 30 total points
- **Peer reviews will be automatically evaluated after submission.**

How to Provide High-Quality Peer Review Feedback

1. **Be Specific and Objective**
   - Avoid vague comments like "*This needs work*" or "*Good job.*"
   - Instead, pinpoint what works well and what can be improved.
   - Example:
     - "*Your state updates correctly, but you could use* `useReducer` *instead of multiple* `useState` *calls to simplify the logic.*"
2. **Focus on Code, Not the Person**
   - Frame feedback around the code itself, not the person writing it.
   - Example:

        - "`useEffect` *is triggering too often because of an incorrect dependency array. You* *check that only necessary values are included.*"

How To Guides

3. **Balance Positive and Constructive Feedback**
    - A good review highlights both strengths and areas for improvement.
    - Example:
        - "*Your UI layout is clean and easy to navigate. One improvement could be adding error handling to form submissions to avoid blank entries.*"
4. **Suggest Improvements, Not Just Problems**
    - If something isn't working, provide a clear suggestion for how to fix it.
    - Example:
        - "*Consider defining event handlers outside the JSX to keep your component more readable.*"
5. **Keep It Concise and Actionable**
    - Avoid writing an essay—focus on key issues that will help the author improve.
    - If possible, group related feedback instead of listing every minor issue separately.
6. **Follow the Evaluation Criteria**
    - Stick to the peer review rubric and evaluate based on the assignment requirements.
    - Don't penalize missing features that weren't required.
7. **Be Respectful and Encouraging**
    - Remember that everyone is learning. Avoid dismissive or harsh language.
    - Example:
        - "*This part doesn't seem to be working as expected. You could try using a different state update pattern to fix it.*"

Resources

To help you complete this assignment, here are some useful references:

- **React Official Documentation** – https://react.dev (https://react.dev/) (Covers hooks, context, and component design)
- **Built-In Hooks** – https://react.dev/reference/react/hooks (https://react.dev/reference/react/hooks) (Detailed explanations of `useState`, `useEffect`, and `useReducer`)
- **Context API** – https://react.dev/reference/react/useContext (https://react.dev/reference/react/useContext) (How to use context to manage global state)
- **Responding to Events** – https://react.dev/learn/responding-to-events (https://react.dev/learn/responding-to-events) (How to handle user interactions in React)
- **Tailwind CSS Documentation** – https://tailwindcss.com/docs (https://tailwindcss.com/docs) (For responsive styling)

Naturally, refer back to the lectures that cover much of this material. However, there is How To Guides
the React framework than what we were able to cover. It is expected that you will be able to do your own research to apply the framework to complete this assignment successfully.

These resources should guide you through the implementation process. If you get stuck, refer to them before asking for help.

Submit Programming Assignment

Upload all files for your submission

Submission Method

Upload GitHub Bitbucket

Drag & Drop

Any file(s) including .zip. Click to browse.