**Timer App Assignment**

Welcome to the Timer App Assignment! This project is designed to evaluate your skills in React development, focusing on **UI implementation**, **code quality**, **state management**, and **best practices**. The project uses **React**, **Vite**, **Tailwind CSS**, and **Vitest** for testing.

**Objective**

Your task is to improve and enhance an existing Timer App based on the following requirements. The app currently has a partially implemented timer system, and your goal is to address the listed issues and extend its functionality.

**Tech Stack**

* **Frontend Framework**: React (with Vite for fast development)
* **Styling**: Tailwind CSS
* **Testing Framework**: Vitest (for unit and component testing)

**Commit Messages and Tracking**

* **Commit Often**: After each significant Feature or Change , commit your work.
* **Conventional Commit Standards**: Use commit messages that follow these guidelines:
  + **feat**: for new features (e.g., feat: add stopwatch lap functionality)
  + **fix**: for bug fixes (e.g., fix: resolve snack bar dismiss error)
  + **refactor**: for code refactoring (e.g., refactor: extract modal buttons into reusable component)
  + **Tracking Changes**: Clear, descriptive commit messages are essential for tracking progress and understanding the history of your changes.

**Stopwatch Feature**

| **Task** | **Marks** |
| --- | --- |
| UI Matching | 15 |
| Functionality | 28 |
| Responsive Navigation & Layout (Mobile) | 7 |
| Test Cases | 10 |
| **Total** | **60** |

**Timer Feature**

| **Task** | **Marks** |
| --- | --- |
| Match the UI | 3 |
| Simultaneous Timers | 7 |
| Snack Bar Behavior | 1 |
| Fix Snack Bar Console Error | 1 |
| Extract Common Components | 5 |
| Consolidate Modal Code | 5 |
| Validation Snack Bars | 1 |
| Responsive Snack Bar Placement | 1 |
| Write Tests | 5 |
| Timer Persistence | 10 |
| Use Google Font | 0.5 |
| Add Favicon Icon | 0.5 |
| **Total** | **40** |

**Bonus Points**

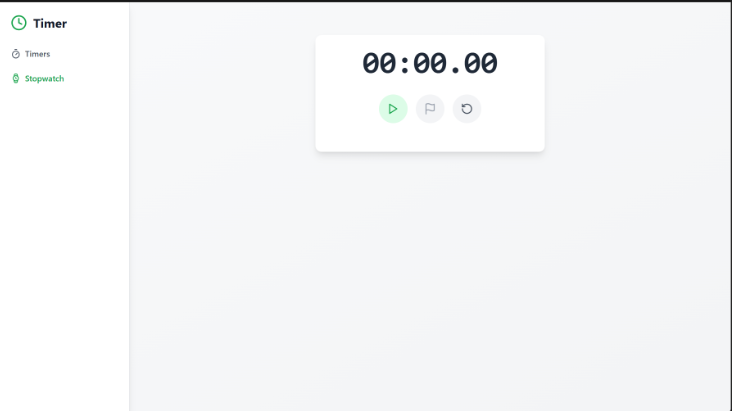
* For Conventional commits

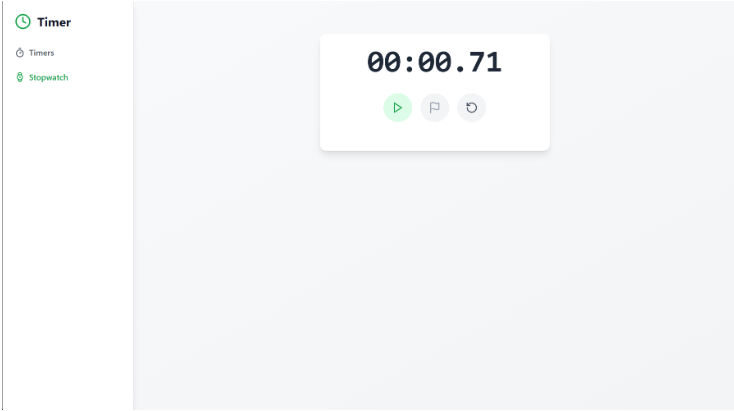
**Steps to Complete**

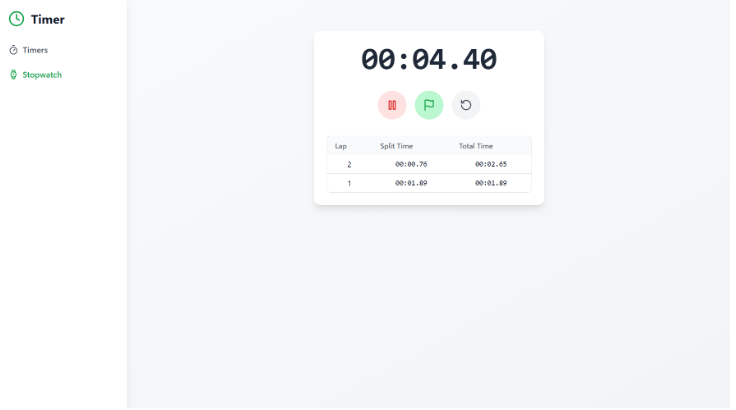
1. **Fork or Clone the Repository**
   * Fork or clone the repository to your local machine.
   * Set up the project using the provided instructions.
2. **Complete the Following Tasks**

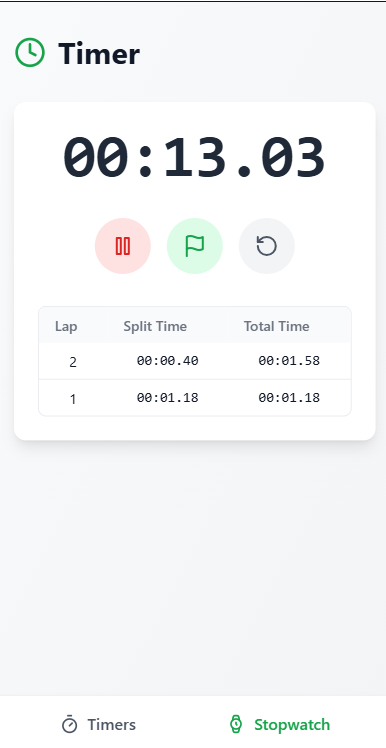
**Stopwatch Feature (Total Marks: 60)**

* + **UI Matching: Marks 15**
    - **Design Consistency:** The Stopwatch UI should match the provided design screenshots.
    - **Component Styling:** Ensure that the layout, colors, fonts, and spacing are consistent with the Timer section.
    - **Visual Feedback:** The stopwatch display and lap list should offer clear visual cues for start, stop, lap, and restart states.





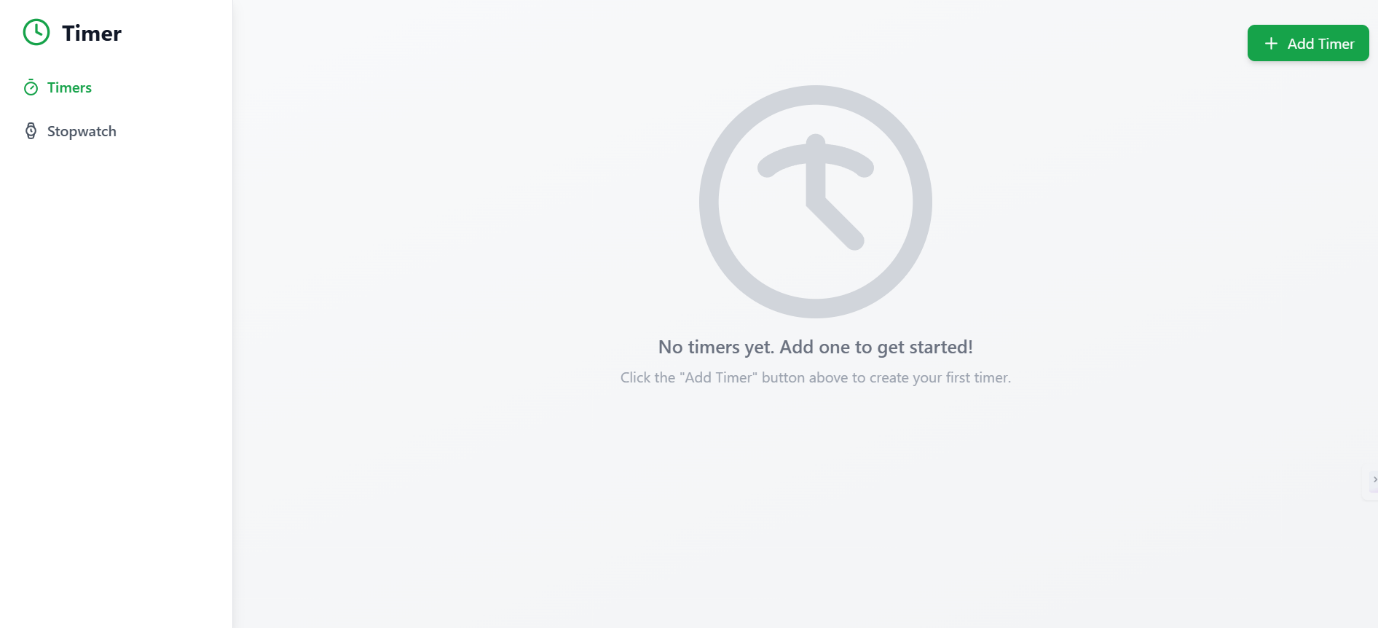


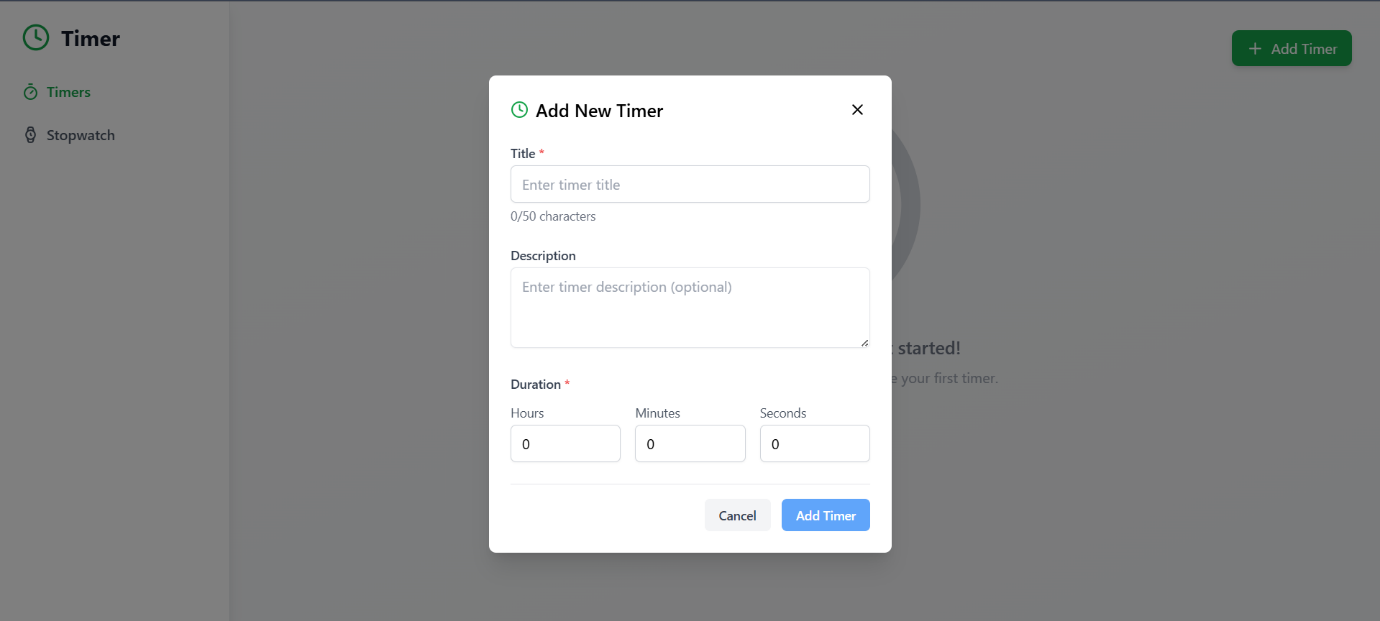


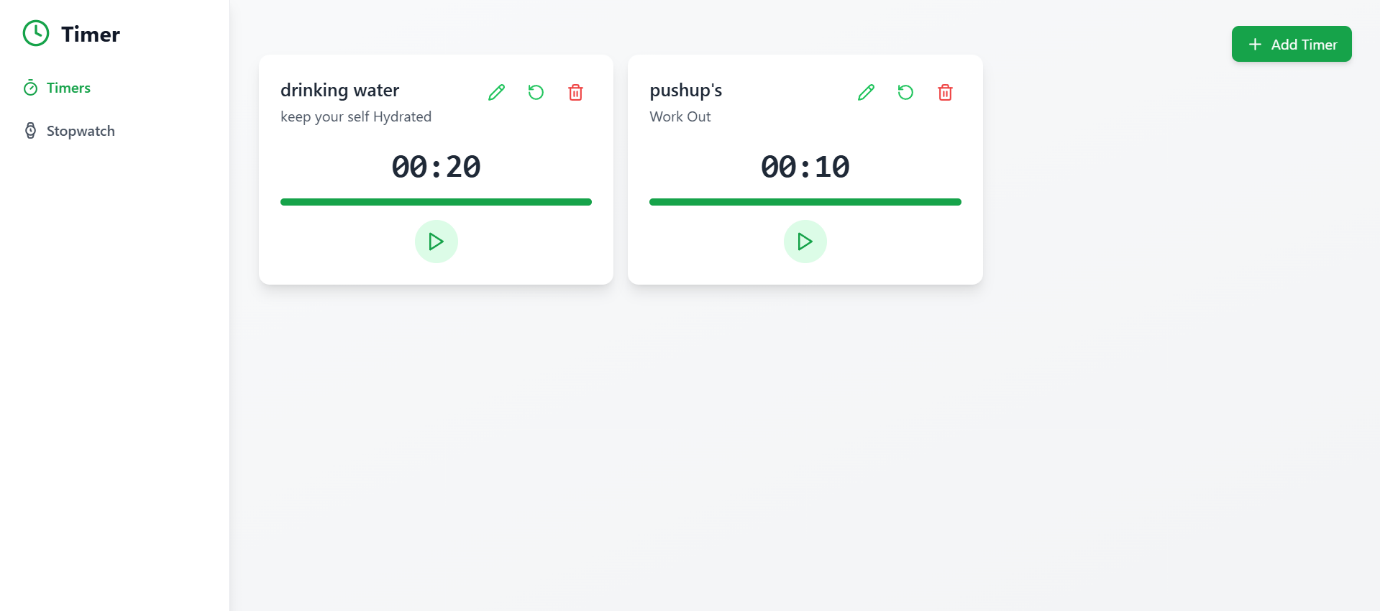
* + **Functionality (28 Marks)**
  + **Core Features : Marks 18**
    - **Start & Stop:** Implement a seamless transition between starting and stopping the stopwatch.
    - **Lap:** Allow users to record lap times. Display each lap with clear timing information.
    - **Restart:** Provide a reset option that clears the stopwatch and lap times.
  + **Additional Interactions : Marks 10**
    - **Button Feedback:** Visual cues (e.g., button highlighting) should indicate active states.
    - **Error Handling:** Ensure robust error handling for edge cases, such as multiple rapid clicks.
  1. **Responsive Navigation & Layout (7 Marks)**
  + **Mobile Devices: Marks 7**
    - Implement the tab navigation as a bottom navigation bar.
    - Ensure that the stopwatch and timer views are fully accessible and user-friendly on smaller screens.
  1. **Test Cases (10 Marks)**
  + **Unit Tests:**
    - Write tests for individual stopwatch functions (e.g., start, stop, lap, restart) to ensure they work as expected.
  + **Component Tests:**
    - Write component tests that validate UI rendering, user interactions (clicks on buttons), and state updates.

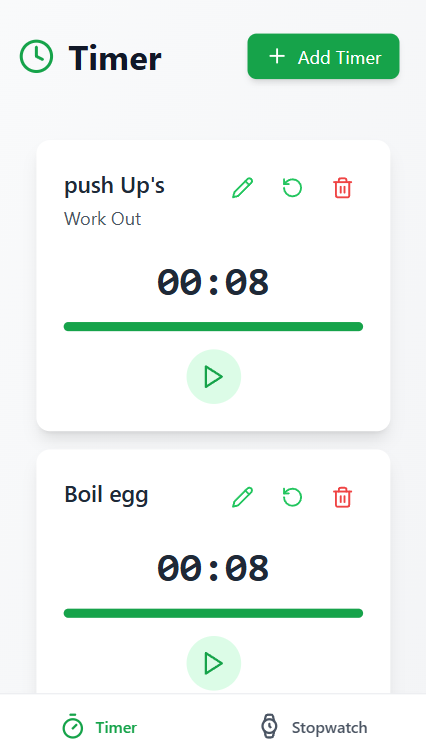
**Timer Feature (Total Marks: 30 )**

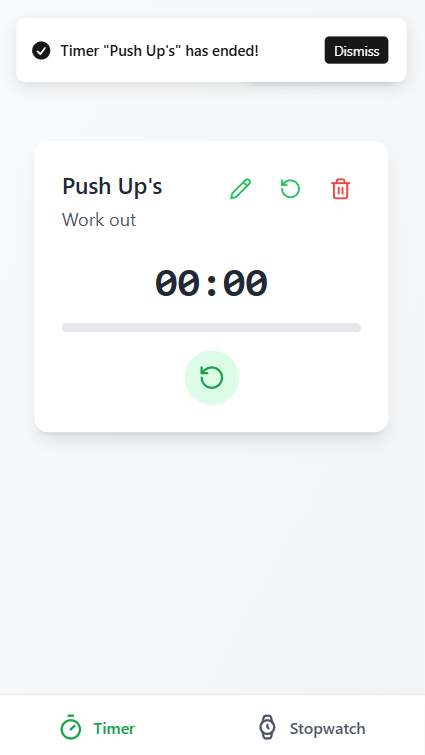
* 1. **Match the UI: Marks 3**
     1. Ensure the app's UI matches the given **screenshots**.











* 1. **Simultaneous Timers: Marks 7**
     1. Update the app to allow multiple timers to run simultaneously (currently, only one timer runs at a time).
  2. **Snack Bar Behavior: Marks 1**
     1. When a timer is completed:
        1. A snack bar notification should display.
        2. The notification sound should keep playing until the snack bar is dismissed.
  3. **Fix Snack Bar Console Error: Marks 1**
     1. Resolve the **console error** that occurs when the snack bar's **dismiss button** is clicked.
  4. **Extract Common Components: Marks 5**
     1. Extract the buttons in the **Add/Edit Timer Modal** as a **separate reusable component**.
     2. Replace all instances of similar buttons in the app with this component.
  5. **Consolidate Modal Code: Marks 5**
     1. Refactor the code to use a **single modal component** for both adding and editing timers, eliminating duplication.
  6. **Validation Snack Bars: Marks 1**
     1. Currently, the **Submit button** is disabled when the form is invalid.
     2. Show an **error snack bar** or notification when the form is submitted with invalid data.
  7. **Responsive Snack Bar Placement: Marks 1**
     1. For **desktop devices**: Display snack bars in the **top-right corner**.
     2. For **mobile devices**: Display snack bars at the **bottom of the screen**.
  8. **Write Tests: Marks 5**
  9. Add **unit tests** for the validation.ts file to ensure all validation rules work as expected.
  10. Write **component tests** for reusable components like TimerItem and ModalButtons.
  11. **Timer Persistence: Marks 10**
      1. Use **localStorage** to persist timers across page refreshes.
  12. **Use Google Font: Marks 0.5**
      1. Apply the **"Tinos"** font from [Google Fonts](https://fonts.google.com/specimen/Tinos) across the entire app for consistent typography.
  13. **Add Favicon Icon: Marks 0.5**
      1. Add a favicon to the app.
      2. The icon is already present in the public/icons folder.
      3. Ensure it displays correctly in browser tabs.

**Project Setup**

1. Clone the repository:

git clone https://github.com/CW-Codewalnut/timer.git

1. Install dependencies:

npm install

1. Start the development server:

npm run dev

1. Run tests:

npm vitest

**Evaluation Criteria**

You will be evaluated on the following points:

1. **UI Matching:**
   * The app's UI should match the provided screenshots.
2. **Code Quality:**
   * Clean, modular, and readable code.
   * Avoid code duplication and ensure reusable components are implemented.
3. **Functionality:**
   * Simultaneous timers, snack bar notifications, and localStorage persistence should work seamlessly.
4. **State Management:**
   * Effective use of React hooks or Context API for managing state.
5. **Testing:**
   * Comprehensive unit and component tests, especially for validation logic and reusable components.
6. **Error Handling:**
   * Resolve the existing snack bar console error and provide meaningful feedback to users for invalid forms.
7. **Responsiveness:**
   * Snack bar placement should adapt based on device type (desktop vs. mobile).
8. **Commit Messages:**
   * Follow **conventional commit standards** (e.g., feat:, fix:, refactor:).
   * Practice committing after each change rather than committing everything at once.

**Deliverables**

1. A **GitHub repository link** to your completed project (forked from the original repo).
2. Include a README.md describing:
   * Steps to run your project.
   * Any additional changes or enhancements you made.

**Time Constraint**

You are expected to complete this assignment in **4 hours** of focused effort.

**Contact**

If you have any questions or issues, feel free to reach out via the provided contact channels in the repository.

Good luck! 🚀