Project Title: Expense Tracker Application

By

R. Sakthi Manikandan

To

Unified Mentors.

Duration: 3 Months.

Date: 15-05-2025 to 15-08-2025.

Contents

Abstract

The Expense Tracker App is a fully functional web application designed to help users track their personal finances. Built using HTML, Tailwind CSS, and JavaScript, this project allows users to add, edit, and delete expense entries along with the date and category of each expense. One of the key highlights of this app is its ability to filter expenses by category and time—daily, monthly, or yearly. It also calculates the total spent and visualizes the data using charts for better understanding.

This project allowed me to dive deeper into advanced JavaScript concepts like filtering arrays, working with dates, and integrating chart libraries. The user interface is modern and responsive, thanks to Tailwind CSS. The logic behind expense management not only adds real-world value but also demonstrates how useful tools can be created using front-end technologies. Overall, this project strengthened my coding and UI/UX skills and gave me confidence in building practical and interactive web apps.

Objective

- To design and develop a full-featured expense tracker for managing personal finances.
- To allow users to add, edit, and delete expense entries along with date and category fields.
- To implement filtering options by category and date (daily, monthly, yearly).
- To use JavaScript to calculate and display the total expense dynamically.
- To integrate chart visualization for better financial analysis and user experience.
- To improve skills in working with data arrays, filtering, date handling, and third-party chart libraries.

Technology Stack

3.1. HTM

- HTML is the standard language of the web, ensuring that your web pages are displayed consistently across all major browsers like Chrome, Firefox, Safari, and Opera.
- It works in conjunction with CSS (for styling) and JavaScript (for interactivity) to create visually appealing and dynamic web experiences.
- HTML uses various tags to structure and organize content logically, making it easier to read and understand for both human users and search engines.
- Semantic HTML tags
  like <header>, <nav>, <main>, <article>, <section>,
  and <footer> provide clear meaning and context to different parts of your webpage, improving readability, accessibility, and SEO.
- HTML integrates seamlessly with other web technologies like CSS for styling, JavaScript for interactivity, and backend languages like PHP, Python, and Ruby for dynamically generated content.

3.2. CSS

- CSS is the language we use to style an HTML document.
- CSS describes how HTML elements should be displayed.
- In this project, we have used the Css framework **Tailwind Css.**
- Faster UI building: Tailwind allows you to apply styles directly within HTML using predefined utility classes, eliminating the need to write custom CSS from scratch.
- Centralized configuration: The tailwind.config.js file allows you to define your design system (colors, spacing, typography, etc.) in one place, ensuring adherence to design guidelines.
- Reduced CSS file size: Tailwind's PurgeCSS integration automatically removes unused styles from your production build, resulting in a smaller CSS file and faster load times.

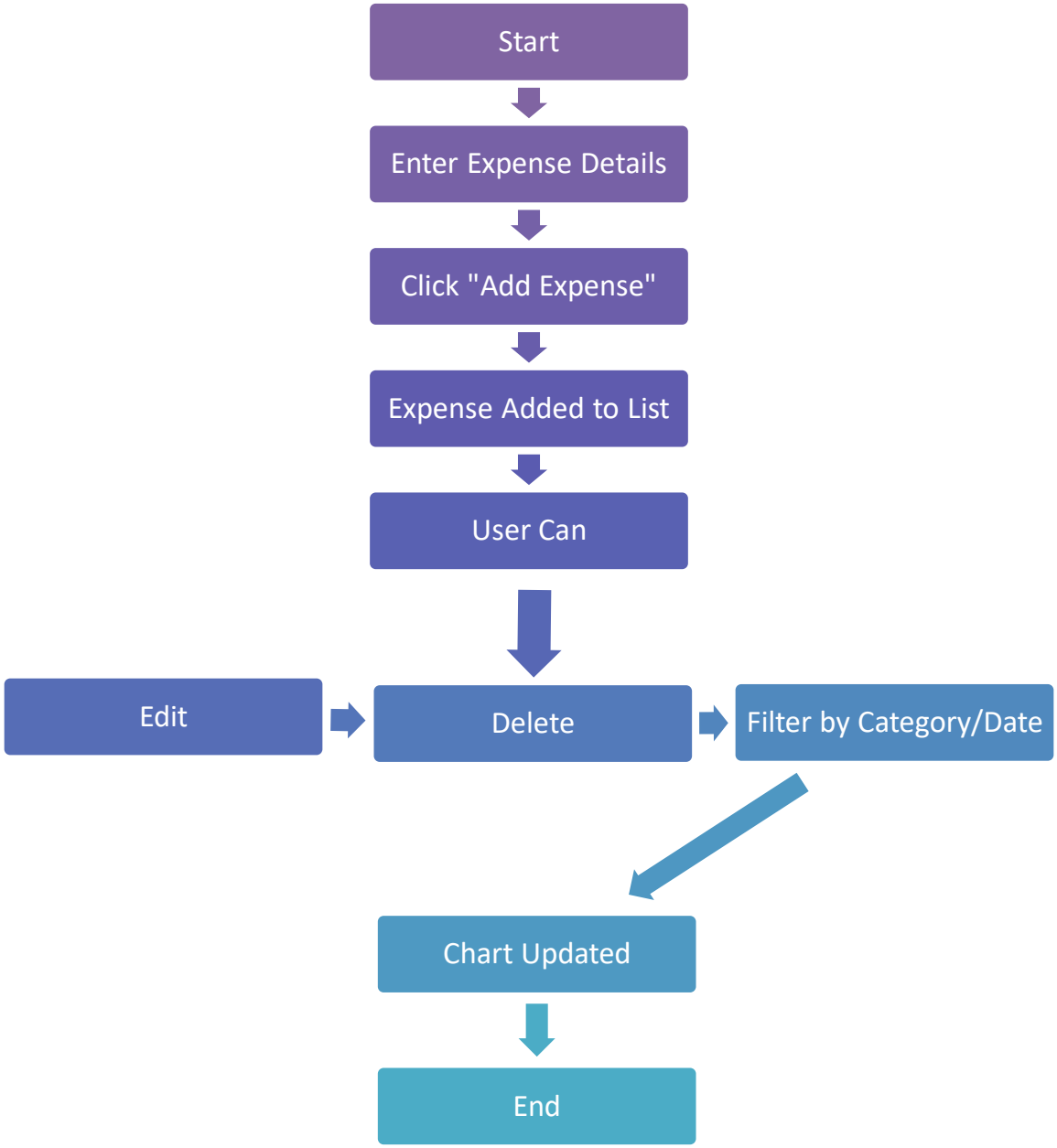
.

3.3. JAVASCRIPT

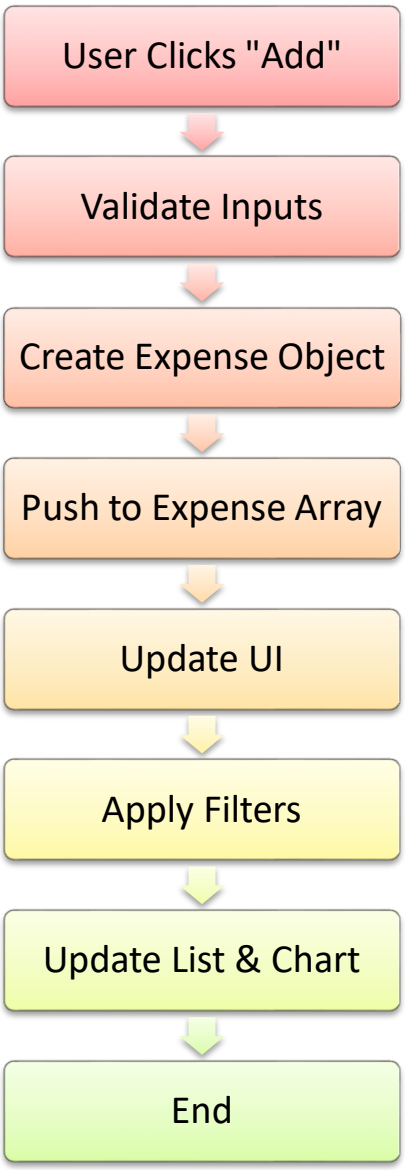
- JavaScript is a versatile and powerful programming language that forms a core part of web development alongside HTML and CSS. It's used to add interactivity, functionality, and dynamic behavior to web pages and applications.
- JavaScript breathes life into static HTML pages by allowing you to add dynamic content, animations, and interactive elements.
- JavaScript is primarily known for client-side (front-end) web development, where it runs directly in the user's browser, handling user interactions and manipulating the Document Object Model (DOM).
- JavaScript has a vast and vibrant ecosystem of libraries and frameworks like React, Angular, Vue.js, and Express.js.
- These tools simplify development, provide reusable components, and offer a structured approach to building complex web applications.
- JavaScript runs seamlessly across all major web browsers and operating systems, making it a reliable and versatile choice for web development
- Node.js – Server-side environment to handle real-time messaging.
- Express.js – Lightweight web framework used to serve HTML and manage routes.
- Socket.IO – Enables real-time, bi-directional communication between client and server.
- WebSockets – Underlying technology powering the chat feature.

4. Flow chart

4.1. User flow chart.



4.2. System flow chart



5. Features Implemented

The Expense Tracker App is a comprehensive financial tracking tool that allows users to manage, categorize, and visualize their expenses in a user-friendly environment. The app was designed with real-life scenarios in mind and includes features commonly needed in personal finance applications.

Key features include:

- **Add Expense:** Users can input an expense with a description, amount, date, and category. The form ensures that all fields are required to avoid missing data.
- **Expense List:** All added expenses are listed clearly below the form. Each entry shows the expense title, date, amount, and category.
- **Edit Functionality:** If a user wants to change a previously added expense, the edit button will allow them to refill the form with existing values and re-add the corrected data.
- **Delete Functionality:** Users can delete any expense, which instantly removes it from the list and recalculates the total.
- **Category Filter:** A dropdown allows users to filter the expense list by category (e.g., Food, Transport, Utilities, etc.) to better understand their spending patterns.
- **Date Filter (Extended Feature):** A logical filter can be implemented to view expenses by day, month, or year, helping users analyze their financial habits over time.
- **Expense Total:** The application calculates and displays the total expense of the currently visible entries. This updates in real time based on filters or edits.
- **Responsive UI:** Tailwind CSS was used to design a mobile-first, clean, and functional layout that adapts to different devices and screen sizes.
- **Chart Visualization (Extended Feature):** A pie or bar chart can be used to visualize total spending by category, offering users a clear picture of where their money goes.

These features together create a powerful personal finance management app that is both functional and visually appealing.

6. Code Structure

The code is modular and broken into sections that separate data management, UI rendering, and user interaction. The technologies used include HTML, Tailwind CSS, and Vanilla JavaScript. The layout is clean and intuitive, allowing for easy understanding and future scalability.

HTML Structure:

The HTML includes:

- A form with input fields: description, amount, date, and category.
- Buttons for submitting data.
- A dropdown for filtering expenses.
- A dynamic list area to show expenses.
- A display area for the total expenses.

CSS Styling with Tailwind:

- Tailwind's utility-first classes are used for:
- Spacing (p-4, m-2)
- Layout (flex, grid, justify-between)
- Colors and typography (text-gray-700, font-semibold)
- Responsiveness with breakpoint utilities (md:, lg:).

JavaScript Functionality:

- **Data Structure:** Each expense is stored as an object with the following fields:

  ```
  {
  id: 123456789,
  description: "Lunch",
  amount: 120,
  date: "2025-08-05",
  category: "Food"
  }
  ```

- **Array of Expenses:** All expenses are stored in a global array.
- **Form Submission:** An event listener handles form validation, creates the expense object, and updates the array.
- **Display Function:** A reusable displayExpenses() function filters, renders, and updates the DOM elements.
- **Delete/Edit Logic:** Buttons are dynamically generated with each entry and tied to functions using their unique id.
- **Filters:** Category filter dropdown affects which expenses are displayed and recalculates the total.
- **Chart Integration:** A function renders a pie chart using Chart.js based on category totals.

This structure is scalable, clean, and allows further additions like login systems, export features, or analytics.

7. Challenges Faced

- Creating a full-featured expense tracker involved overcoming a variety of challenges that enhanced my development skills significantly.

- Data Validation and Error Handling: Initially, the form would allow submission with missing fields. JavaScript-based validations were added to prevent empty descriptions or zero amounts.

- Dynamic Rendering: Managing a list of expense entries that change with filters or after deletion required a smart update mechanism. I learned how to create a central function to render everything and refresh the view whenever the state changes.

- Maintaining Unique Identifiers: Editing and deleting specific expenses needed a way to track each entry. Using Date.now() to generate unique IDs solved this problem.

- Category Filter Logic: When applying category filters, the list was not correctly updating at first. The fix involved correctly chaining filter() and forEach() methods, and updating the total within the filter.

- Date Formatting and Filtering: Handling date filters like “today”, “this month”, or “this year” required understanding of how to work with Date() objects in JavaScript. It involved a lot of trial and error to compare ISO date strings and apply date-based logic.

- Chart Integration: Adding Chart.js and feeding it data dynamically introduced complexity, especially with filtering. I had to convert the array of expenses into a format that could be used by the chart.

- Responsiveness: Although Tailwind helps with responsive design, aligning elements like dropdowns and inputs consistently on different screen sizes was challenging.

- These hurdles were instrumental in learning how to debug, refactor code, and work with dynamic data in a real-world scenario.

4. Conclusion

The Expense Tracker App was an incredibly valuable project that bridged design, interactivity, and data handling. It helped me better understand how front-end applications can manage real-world problems like budgeting. From input validation to chart visualization and filter logic, each component added to my knowledge of web development. This app taught me not just to build interfaces but to build systems that users can actually depend on. It also laid the groundwork for future learning in backend integration, authentication, and financial reporting features.