SmartLife Tracker

A Project Report *Submitted by:*

N Venkata Praneeth – AP23110010960

P V Saketh Ram – AP23110010955

Y Lohith Sai – AP23110010951

K Janardhan – AP23110011326

Under the Supervision of
Mr. Yatharth Shahrawat

Assistant Professor

Department of Computer Science and Engineering

SRM University-AP

In partial fulfilment for the requirements of the

FullStack Development-I(SEC 160) Project

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Certificate

This is to certify that the project work entitled **"SmartLife Tracker"** is a Bonafide record of project work carried out by the following students:

- **Mr. N. Venkata Praneeth** (AP23110010960)
- **Mr. P. V Saketh Ram** (AP23110010955)
- **Mr. Y. Lohith Sai** (AP23110010951)
- **Mr. K. Janardhan** (AP23110011326)

from the **Department of Computer Science and Engineering, SRM University-AP.**

The students conducted this project work under my supervision during the period August 2025 to December 2025. It is further certified that, to the best of my knowledge, this project has not previously formed the basis for the award of any degree or any similar title to this or any other candidate.

This is also to certify that the project work represents the teamwork of the candidates.

Station: Mangalagiri

Date: 06/12/2025

Mr. Yatharth Shahrawat

Assistant Professor
Department of Computer Science & Engineering

SRM University-AP
Andhra Pradesh.

Acknowledgement

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success.

We are extremely grateful and express my profound gratitude and indebtedness to our project guide, **Mr. Yatharth Shahrawat,** Department of Computer Science & Engineering, SRM University, Andhra pradesh, for his kind help and for giving us the necessary guidance and valuable suggestions in completing this project work.

Table of Contents

Acknowledgements 2

Certificate 3

Table of Contents 4

1. Introduction 5
2. Scenario-Based Introduction 6
3. Target Audience 7
4. Project Goals and Objectives 8-9
5. Key Features 10-11
6. Pre-Requisites 12-13
7. Project Structure 14-15
8. Project Flow 16-25
9. Project Execution 26-34
10. Project Demo Links 35

1. Introduction

Managing multiple aspects of daily life—such as tasks, expenses, and health habits—can become overwhelming without an organized system. Individuals often depend on separate apps or manual notes to track different activities, resulting in fragmented information and reduced productivity. To address this challenge and provide a unified, structured, and user-friendly solution, our team developed **SmartLife Tracker**, a modern web application designed to centralize essential lifestyle management into one streamlined platform.

SmartLife Tracker enables users to manage daily tasks, monitor expenses, track personal health activities, and maintain a personalized profile. Built using **React.js** for the frontend and **JSON Server** for backend simulation, the application demonstrates a complete, functional ecosystem capable of storing, retrieving, and updating user data without requiring a full-scale backend infrastructure. Through dynamic UI updates, responsive layouts, and structured components, SmartLife Tracker ensures that users can interact with the application seamlessly and efficiently.

The system follows clean design principles, ensuring minimal complexity and maximum usability. Each module—Tasks, Expenses, Health, Profile, and Dashboard—is built to function independently while contributing to an interconnected experience. Data persistence is handled through JSON Server, ensuring that user entries remain intact even after refreshing the page or restarting the application.

SmartLife Tracker also demonstrates software engineering principles such as modular development, component reusability, organized folder structures, and scalable architecture. With future enhancements, the system can be extended to include deeper analytics, authentication features, chart expansions, and cloud-based data storage.

Overall, SmartLife Tracker bridges the gap between scattered productivity tools and provides users with a unified platform to gain better control over their daily activities, financial patterns, and personal well-being.

2. Scenario-Based Introduction

Imagine beginning your day by opening your laptop or unlocking your phone. Instead of jumping between separate apps for tasks, expenses, and health logs, you open **SmartLife Tracker**—a single dashboard that provides clarity across all your daily activities.

The Dashboard greets you with an organized summary of your pending and completed tasks, today's health progress, and a snapshot of your weekly or monthly expenses. With this overview, you instantly understand your productivity level, financial standing, and wellness balance for the day.

Next, you navigate to the Tasks Page to plan your day. You add tasks such as "Finish assignment," "Buy groceries," or "Workout at 6 PM." Each task is displayed with clear design elements, and you can mark them as completed whenever you finish them.

Later in the day, you record your expenses—taxi fare, lunch cost, or shopping bills—on the Expenses Page. The system updates your total spending automatically and compares your usage with your preset spending limits. You can see whether you're nearing your weekly or monthly budget.

In the evening, you open the Health Page to log your progress for water intake, steps walked, and hours slept. Every input updates the health summary instantly, allowing you to track daily habits and overall wellness trends.

When needed, you open your Profile Page to update your personal details such as name, age, or email ID. You may also modify your spending limits, and the changes reflect everywhere instantly.

This integrated experience eliminates the need to switch between multiple applications and provides a single platform that supports productivity, financial awareness, and personal health monitoring. SmartLife Tracker becomes a personal digital assistant that grows with your routine, making your lifestyle more organized, informed, and balanced.

3. Target Audience

SmartLife Tracker is designed to meet the needs of a broad range of users who seek better lifestyle organization through simple yet effective digital tools. The application caters to the following groups:

1. Students
Students require help managing assignments, routine tasks, health habits, and daily expenses. SmartLife Tracker enables them to keep track of academic responsibilities, maintain budget control, and develop healthy routines.

2. Working Professionals
Professionals who juggle office tasks, personal goals, financial responsibilities, and health routines benefit from a centralized platform that helps them maintain balance and plan efficiently.

3. Fitness and Wellness-Oriented Individuals
Users who track steps, sleep hours, and hydration can use the Health module to maintain a consistent wellness routine and monitor progress over time.

4. Budget-Conscious Users
Individuals who monitor their spending habits and aim to follow weekly or monthly financial limits can use the Expenses module for budgeting and financial planning.

5. Productivity-Focused Users
People who prefer a structured system to plan tasks, maintain checklists, and improve productivity will find the Tasks module highly useful.

By combining daily planning, budgeting, and wellness tracking into a unified platform, SmartLife Tracker becomes a valuable tool for maintaining an organized lifestyle.

4. Project Goals and Objectives

The primary goal of SmartLife Tracker is to provide users with a unified, modern, and highly structured platform that simplifies the management of multiple aspects of everyday life. As individuals increasingly depend on digital tools to manage activities, financial records, and wellness habits, SmartLife Tracker aims to consolidate these functionalities into a single, easy-to-use application. The system is designed not only to improve productivity but also to encourage users to develop healthier routines, maintain budget discipline, and stay aware of their daily progress.

Primary Goal

To develop a comprehensive lifestyle management system that integrates task tracking, expense monitoring, and health logging into a centralized, intuitive, and responsive web application built using React.js and JSON Server.

Objectives

1. Deliver a unified digital space for life organization
The application aims to eliminate the need for multiple separate apps by combining productivity, budgeting, and health monitoring tools into one platform.

2. Provide an intuitive and consistent user interface
SmartLife Tracker focuses on clean navigation, structured layouts, and responsive design to ensure that users of all technical backgrounds can operate the system with ease.

3. Facilitate complete CRUD operations across all modules
By integrating JSON Server, the system supports creating, reading, updating, and deleting data for tasks, expenses, health logs, and profile settings.

4. Enhance user awareness through analytical insights
Each module provides summaries, categories, totals, and progress indicators, enabling users to understand their performance and trends at a glance.

5. Ensure data persistence and reliability

All user interactions are stored in db.json, ensuring that data remains intact across sessions, refreshes, and restarts, simulating real backend functionality.

6. Implement modular and maintainable system architecture

React's component-driven structure ensures that each part of the application is isolated, reusable, and simple to modify or enhance during future updates.

7. Promote productivity and personal discipline

The Tasks module improves user accountability, the Expenses module enhances financial awareness, and the Health module encourages the development of consistent well-being habits.

8. Support extensibility for future enhancements

The project is designed in a way that additional features—such as authentication improvements, reminders, advanced graphs, or cloud storage—can be integrated without major restructuring.

5. Key Features

SmartLife Tracker consists of multiple integrated modules designed to streamline essential aspects of a user's daily routine. Each feature is structured to deliver clarity, efficiency, and meaningful insights.

1. Task Management Module

The Task Management module enables users to plan, organize, and complete tasks efficiently. Users can create new tasks, update existing tasks, delete unnecessary tasks, and mark completed tasks. This functionality allows users to structure their daily goals and remain consistent in their productivity. The list format offers clear visibility of pending and completed tasks, helping users stay focused and organized.

2. Expense Tracking Module

This module allows users to record and monitor daily expenses. Each expense entry includes a category, amount, and date, enabling users to maintain accurate financial records. The system also supports weekly and monthly spending limits, helping users manage their budget effectively. When spending approaches or exceeds the set limit, the application can display warning messages. This module enhances financial discipline and encourages informed spending habits.

3. Health Tracking Module

The Health module helps users log health-related activities such as daily water intake, sleep duration, and steps walked. All entries contribute to a summary that reflects personal health progress. By allowing users to compare their daily habits with predefined goals, the application promotes a healthier routine and consistent wellness tracking. This module is especially useful for individuals managing fitness or lifestyle improvement plans.

4. Profile Management Module

The Profile Page allows users to manage their personal information, including name, age, email, password, and spending limits. Any changes made in the profile are immediately updated across the application due to React's state management and JSON Server persistence. This module ensures personalization and accuracy of user-related data.

5. Dashboard Overview

The Dashboard serves as the central hub of the application, providing a high-level summary of all modules. It includes total counts of tasks, expense summaries, and health progress for the day. The structured dashboard allows users to assess their performance and make informed decisions about their tasks, financial activities, and health priorities. This summary-driven approach improves clarity and supports quick decision-making.

6. JSON Server Data Persistence

SmartLife Tracker uses JSON Server to simulate real backend operations. All user interactions—such as adding tasks, logging expenses, and updating health entries—are stored in db.json. This ensures that data persists between browser refreshes or application restarts, enabling long-term use and reliable retrieval of historical data. JSON Server provides REST API endpoints that mimic real-world backend behavior, making the system scalable and accurate for testing and demonstration purposes.

7. Component-Based Architecture

The application is developed following React's modular component architecture. Each component handles a specific functionality, resulting in cleaner code, easier maintenance, and efficient scalability. Components such as Navbar, DashboardCards, TaskForm, ExpenseForm, and HealthLogs promote reusability across the project.

6. PRE-REQUISITES

Developing SmartLife Tracker requires a foundational understanding of modern web development practices and familiarity with essential tools that support frontend engineering. The application is built using React.js for user interface development and JSON Server to simulate backend functionality. Together, these technologies offer a comprehensive development environment suitable for building interactive, data-driven applications.

React.js

React.js is a JavaScript library used to construct modular, component-based interfaces. It enables efficient UI rendering, state management, event handling, and dynamic updates without requiring full page reloads. Developers can create reusable components, allowing the project to remain maintainable and scalable. React's JSX syntax blends HTML and JavaScript, making interface creation intuitive.

Node.js and npm

Node.js is required to run development servers and manage project dependencies. npm (Node Package Manager) installs all necessary libraries such as React Router, Axios, and JSON Server.

Node.js ensures that the development setup runs efficiently across platforms, while npm simplifies version management and package installation.

JSON Server

JSON Server acts as a lightweight backend alternative that provides REST API endpoints. It reads from a db.json file and supports full CRUD operations. This allows SmartLife Tracker to simulate real backend behavior without requiring databases such as MongoDB or SQL. The use of JSON Server makes development faster, easier to test, and suitable for prototype-level applications.

Axios

Axios is a promise-based HTTP client used to communicate with JSON Server. It enables sending GET, POST, PUT, PATCH, and DELETE requests, thereby allowing all modules—Tasks, Expenses, Health, Profile—to store and retrieve user data from the simulated database.

Tailwind CSS

Tailwind CSS is utilized to style portions of the application using utility-first classes. It simplifies layout design, spacing, typography, and responsiveness. This allows SmartLife Tracker to achieve visually consistent and clean user interfaces.

JavaScript, HTML, and CSS

A strong understanding of JavaScript is required for handling component logic, state updates, API requests, and rendering dynamic data. HTML provides the structural foundation, while CSS supports layout and visual styling.

Visual Studio Code

Visual Studio Code is used as the primary development environment due to its integrated terminal, debugging capabilities, and extensive plugin support. Extensions for React, Tailwind CSS, Git, and formatting support streamline the development experience.

Git and GitHub

Git provides version control for tracking changes and managing code history. GitHub is used to host the SmartLife Tracker repository, allowing collaboration, documentation, and submission of the final project.

These prerequisites collectively support the development of a structured, responsive, and user-friendly lifestyle management system.

7. PROJECT STRUCTURE

The SmartLife Tracker project is organized into clearly defined folders and components to ensure modularity, maintainability, and smooth integration between the frontend and the simulated backend.

The following are the important files and folders in the SmartLife Tracker project:

- **index.html** → The root HTML file where the React application is mounted.
- **package.json** → Contains all major dependencies such as React, React Router, Axios, Tailwind CSS, and development tools.
- **db.json** → Stores users, tasks, expenses, and health logs through JSON Server to simulate a backend.

The directory structure is as follows:

smartlife-tracker/

├─ public/

├─ src/

│ ├─ components/ → Reusable UI components (Navbar, DashboardCharts, Cards, Forms)

│ ├─ pages/ → Dashboard, Tasks, Expenses, Health, Profile, Login, Signup

│ ├─ api/ → Handles API calls to JSON Server (tasks, expenses, health, profile)

│ ├─ context/ → AuthContext for login session & global state

│ ├─ App.jsx → Root component holding routes and global layout

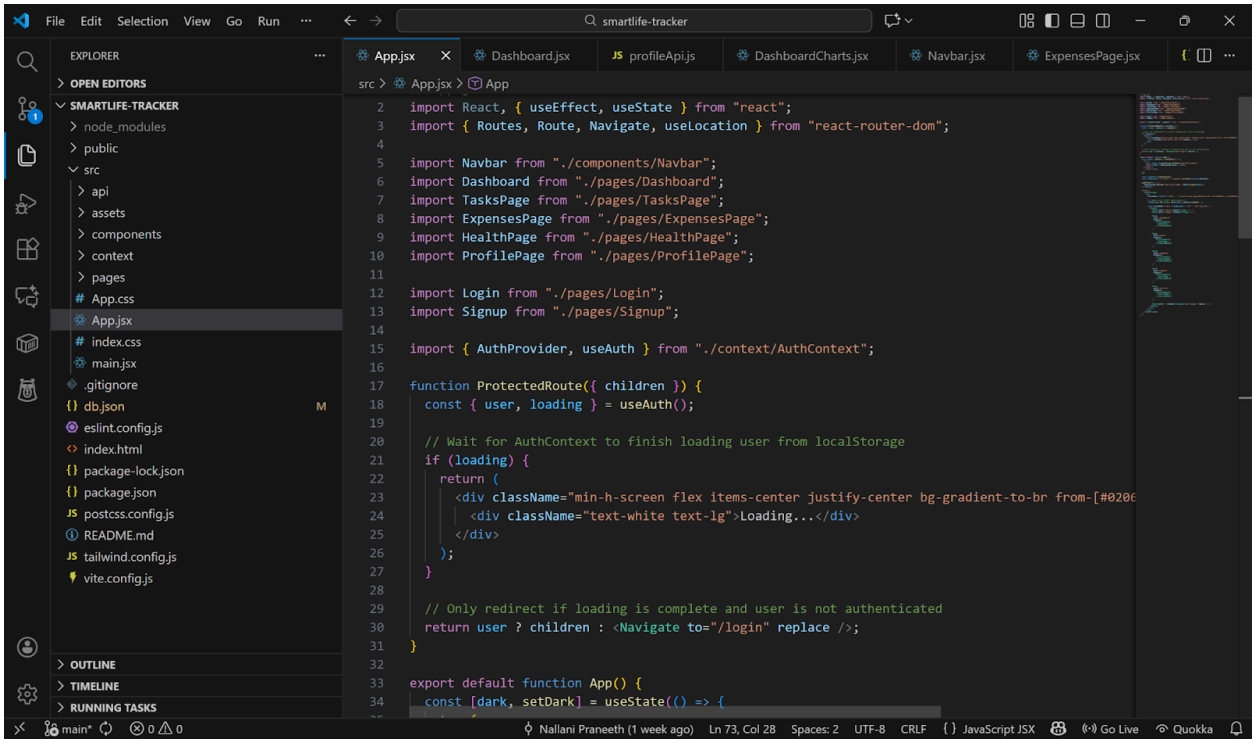
│ └─ main.jsx → Entry point that renders the entire React application

├─ db.json → Fake backend containing user data, tasks, expenses, health logs

├─ index.html → Base HTML where React is injected

├─ package.json → Dependencies and metadata

└─ vite.config.js → Vite build tool configuration



8. PROJECT FLOW

Before starting development, it is important to understand how SmartLife Tracker operates from a user's perspective.

https://github.com/Nallanipraneeth66/SmartLife-Tracker

SmartLife User Flow

1. The user opens the SmartLife Tracker application.
2. Login or Signup is performed through AuthContext and JSON Server.
3. The Dashboard loads summaries of Tasks, Expenses, and Health data from JSON Server.
4. The user navigates to the **Tasks Page** to add, update, or delete tasks.
5. The **Expenses Page** allows adding expenses, applying filters, and monitoring weekly/monthly spending limits.
6. The **Health Page** allows logging daily water intake, sleep hours, and steps.
7. The **Profile Page** displays and updates user information and budget limits.
8. All data changes are instantly reflected through Axios calls to db.json.
9. The entire interface is responsive, styled with Tailwind CSS, and navigated using React Router.

MILESTONE 1: Project Setup & Configuration

1. Install Required Tools

Installed libraries include:

- React.js – UI framework
- React Router – Page navigation
- Axios – API data handling
- JSON Server – Local backend simulation
- Tailwind CSS – Styling & responsiveness
- React Toastify – Notifications

These dependencies are visible inside **package.json**.

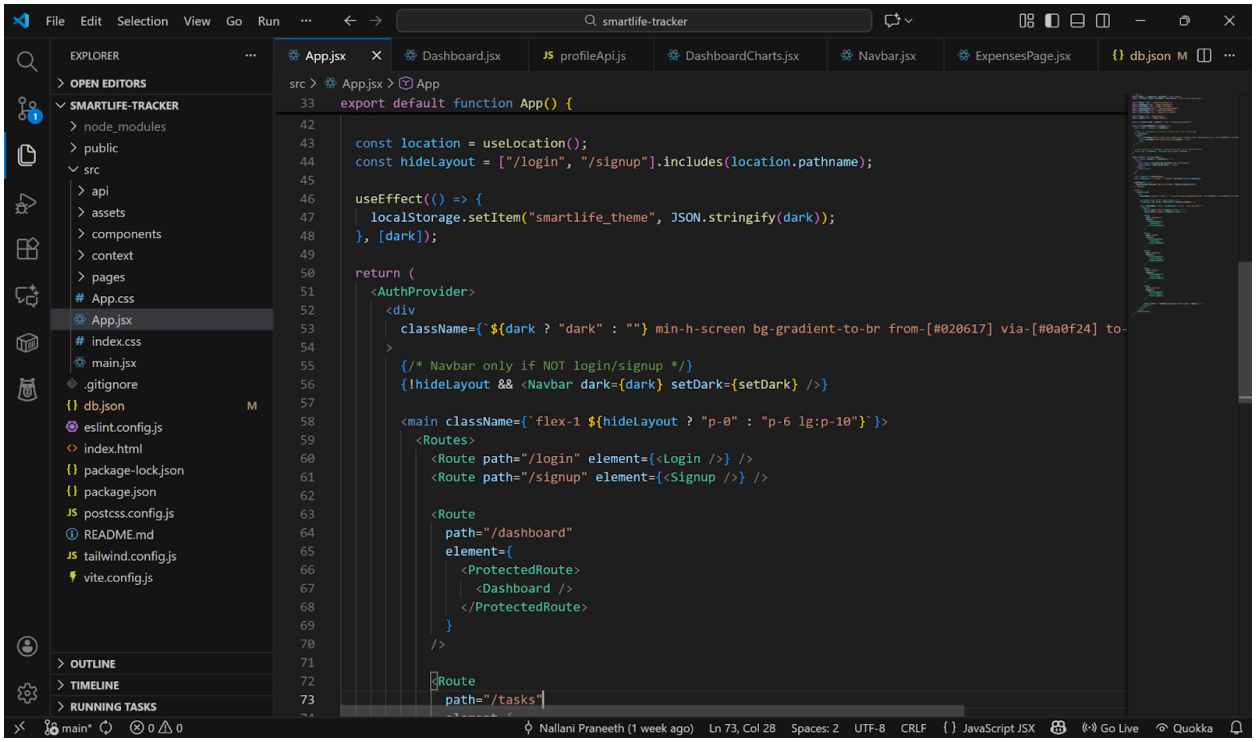
2. Recommended Documentation

- React Installation → https://react.dev/learn/installation
- React Router → https://reactrouter.com
- TailwindCSS → https://tailwindcss.com/docs/installation
- JSON Server → https://www.npmjs.com/package/json-server

MILESTONE 2: Web Development

1. Setup React Application

- Create project using Vite
- Install dependencies listed above
- Configure routing in App.jsx
- Set up db.json with users, tasks, expenses, health
- Connect frontend with JSON Server using Axios



Code Description:

- App.jsx imports React and routing utilities such as Routes, Route, Navigate, and useLocation.
- It imports all main pages including Dashboard, TasksPage, ExpensesPage, HealthPage, ProfilePage, Login, and Signup.
- The Navbar component is also imported to be displayed across all pages.
- AuthProvider and useAuth are imported from AuthContext to manage authentication.
- A ProtectedRoute function is defined to restrict access to authenticated users only.
- ProtectedRoute checks the user and loading values and displays a loading screen until authentication is verified.
- If authentication is complete and the user is not logged in, it redirects to the login page.
- If the user is authenticated, it renders the requested protected component.

- The App component returns the entire routing structure wrapped inside AuthProvider.
- Navbar is rendered above all routing content for consistent navigation.
- Routes are defined for Dashboard, Tasks, Expenses, Health, and Profile using ProtectedRoute.
- Login and Signup pages are accessible without authentication.
- The App component is exported as the default export for use as the root component of the application.

2. Design UI Components

Reusable UI elements include:
- Navbar
- Dashboard Summary Cards
- TaskItem
- ExpenseItem
- HealthLogCard
- Forms for Add/Edit (Tasks, Expenses, Health)

Pages include:
- Dashboard
- Tasks
- Expenses
- Health
- Profile
- Login
- Signup

3. Implement Frontend Logic

- Fetch tasks, expenses, and health logs from JSON Server
- Bind data to UI components
- Implement CRUD for each module:

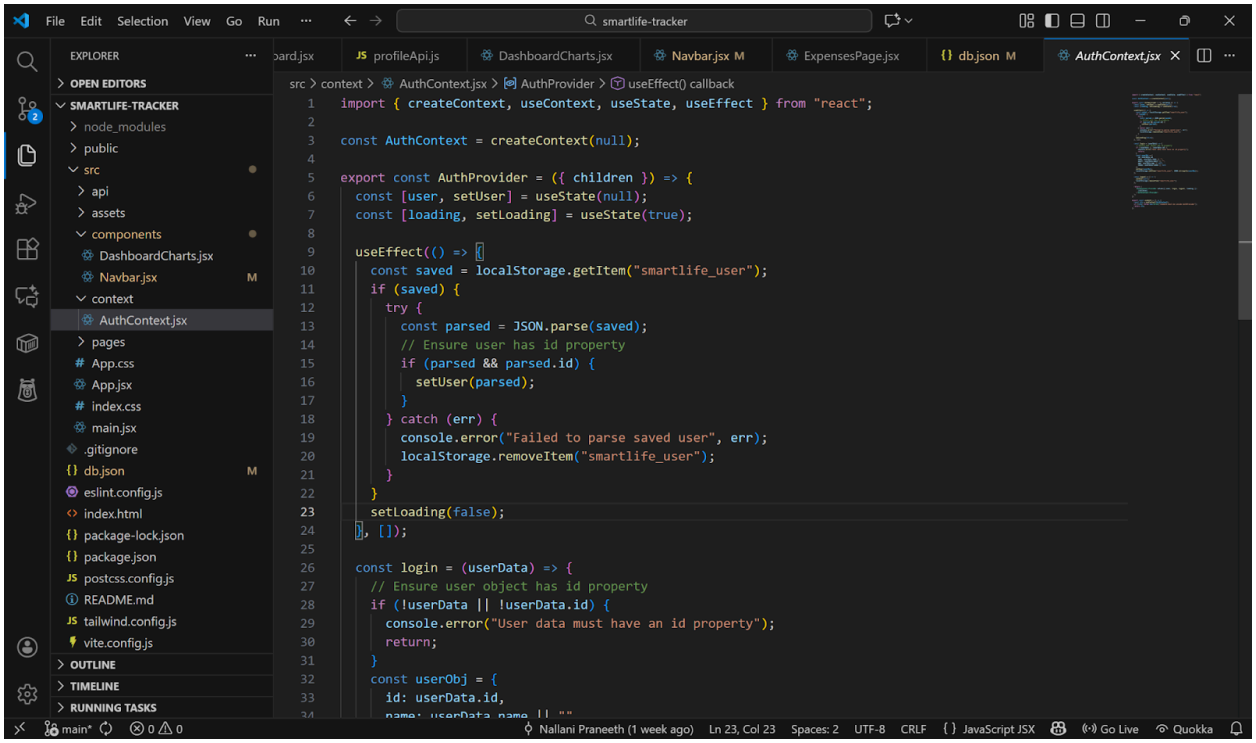
→ Add Task, Edit Task, Delete Task

→ Add Expense, Edit Expense, Delete Expense

→ Log Water/Sleep/Steps

- Implement toast notifications
- Apply filters for expenses
- Manage user session via AuthContext

Data Provider : AuthContext



Code Description:

- Creates an AuthContext using createContext(null) to store and provide authentication state globally.

- Defines an AuthProvider component that wraps around the entire app to supply authentication data.

- Uses useState hooks to store user and loading states.

- Uses useEffect to automatically restore the logged-in user from localStorage when the application loads.

- Parses and validates the stored user object before setting it into state.

- Provides a login function that saves user details in both state and localStorage for session persistence.

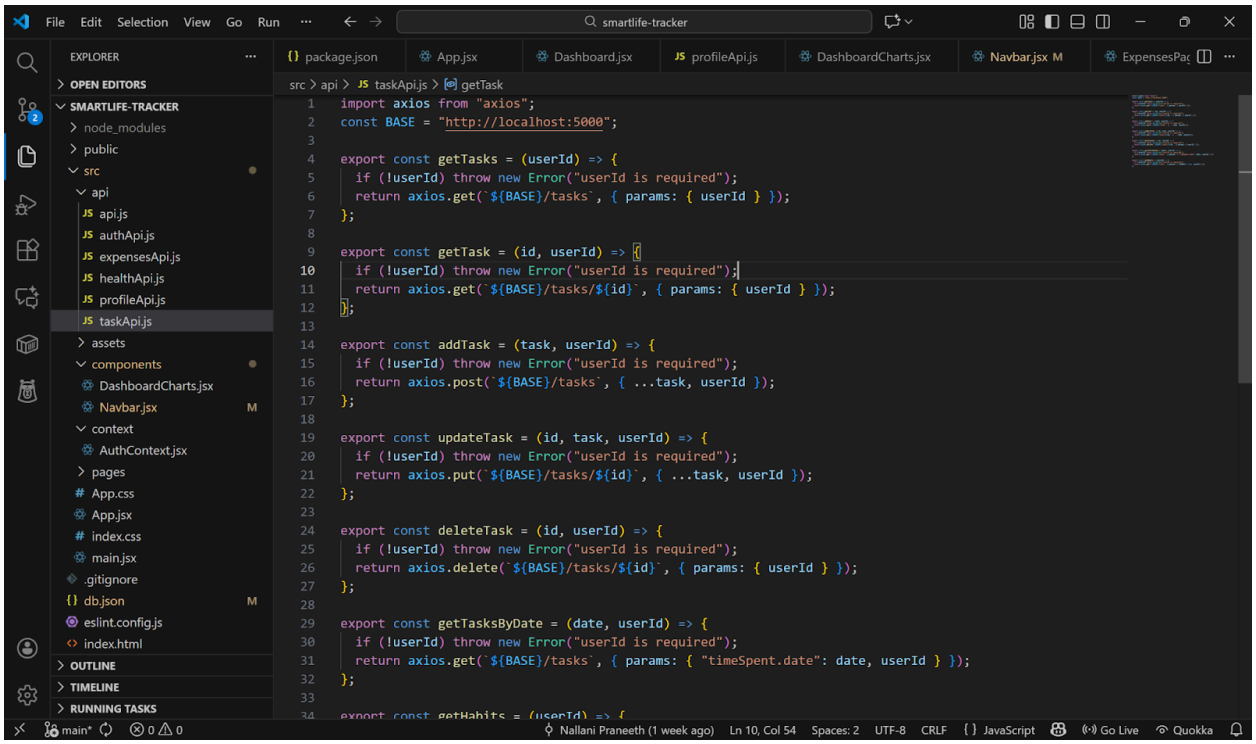
- Provides a logout function (not shown in your screenshot, but available in the file) to clear authentication data and end the session.

- Exposes user, login, logout, and loading through the context provider value, enabling global access.

- Ensures that child components can use the authentication state through the useContext() hook.

- Exports AuthProvider so it can wrap App.jsx and make authentication accessible throughout the project.

Data Provider : taskApi



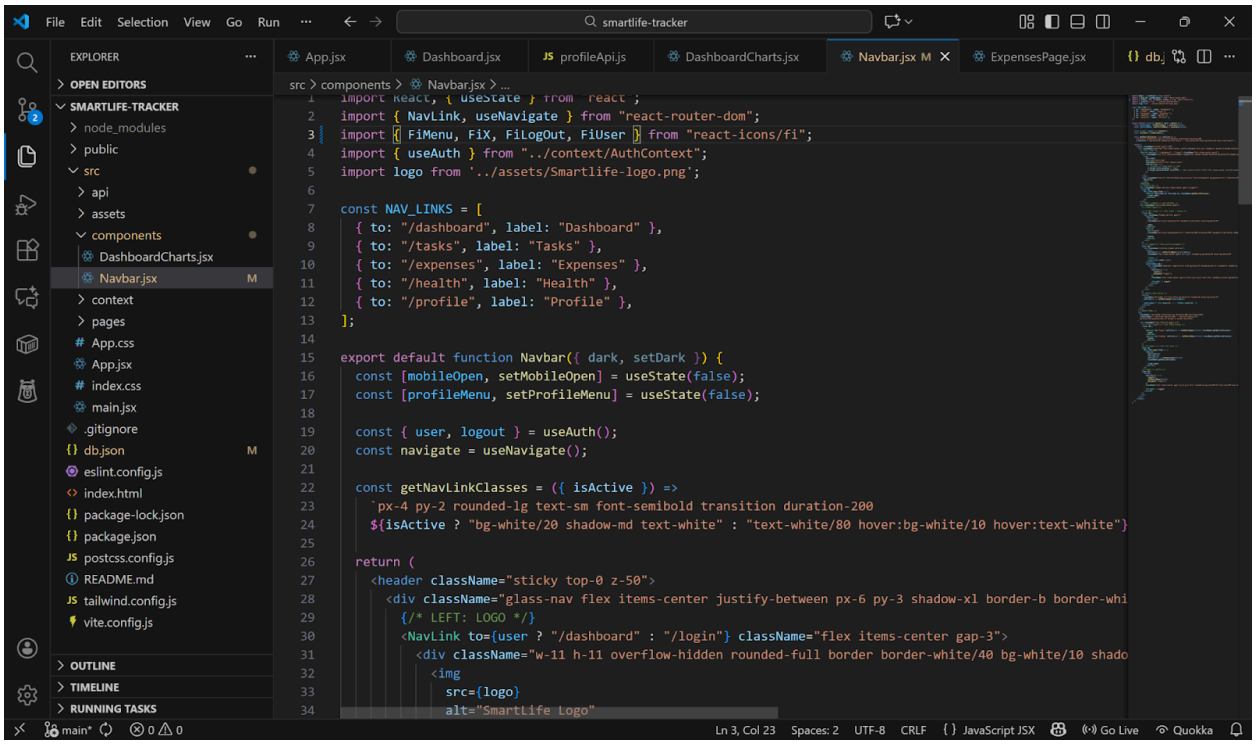
Code Description:

- Imports Axios to perform HTTP requests between the frontend and JSON Server.
- Defines a constant BASE that stores the backend URL (`http://localhost:5000`).
- Provides multiple exported functions for task-related CRUD operations.
- `getTasks(userId)` retrieves all tasks for a specific user using a query parameter.
- `getTask(id, userId)` fetches a single task by its ID while validating userId.
- `addTask(task, userId)` sends a POST request to create a new task and attach the userId to it.
- `updateTask(id, task, userId)` updates an existing task using a PUT request with updated details.

- `deleteTask(id, userId)` deletes a task by its ID and ensures the request belongs to an authenticated user.
- `getTasksByDate(date, userId)` retrieves tasks filtered by the "timeSpent.date" field.
- All functions include validation to ensure a valid userId is provided before performing any request.
- Each API function returns an Axios promise, allowing the calling component to handle results and errors.
- This file centralizes and organizes all task-related API logic for cleaner and reusable code across the application.

Code Component – UI Components

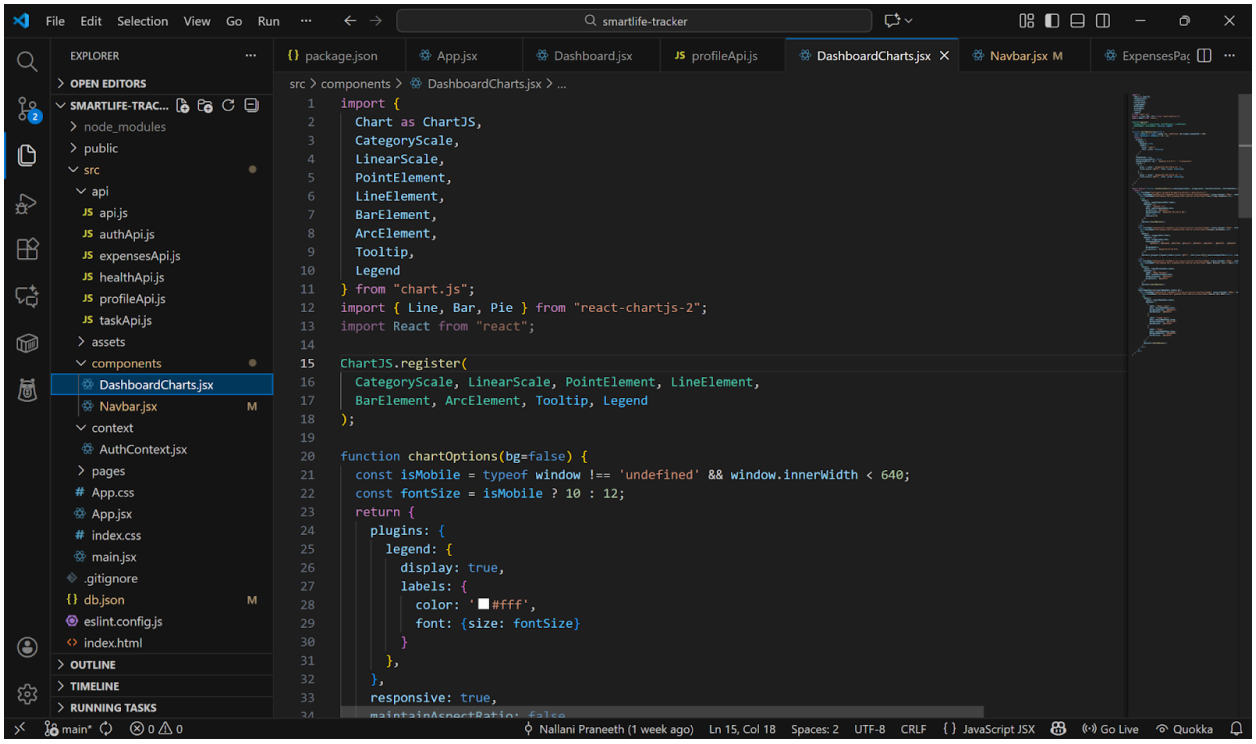
1. Navbar Component

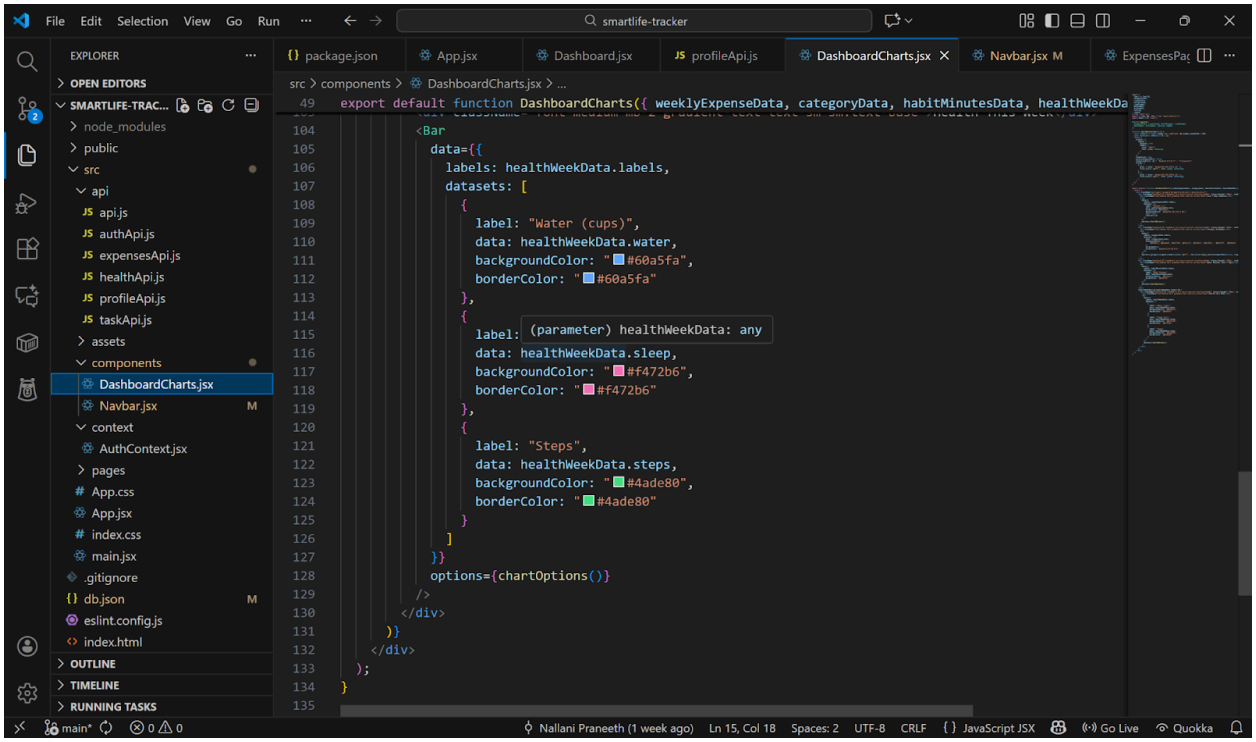


Code Description:

- Imports React and hooks (useState, useEffect)
- Imports Link and useNavigate from React Router
- Displays navigation links: Dashboard, Tasks, Expenses, Health, Profile
- Shows logged-in user's name fetched from AuthContext
- Provides logout functionality
- Responsive design using Tailwind CSS

2. DashboardCharts Component





Code Description:

- Uses Chart.js or simple statistical UI
- Displays summaries of tasks completed, total expenses, and health progress
- Fetches data from API on component load
- Renders graphs and cards

9. Project Execution

After completing development, SmartLife Tracker can be executed as follows:

Running the Frontend

npm run dev

This launches the application at:

http://localhost:5173/

Running JSON Server

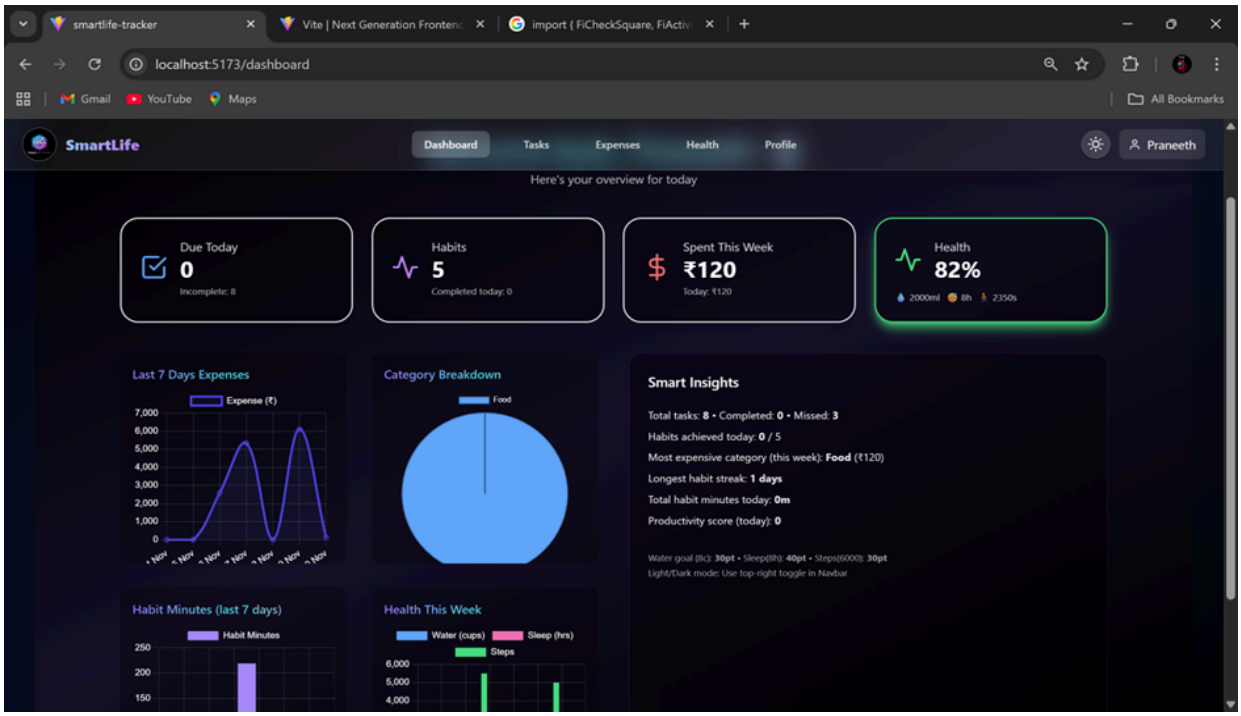
npm run server (or) json-server --watch db.json --port 5000

Backend API endpoints include:

- /users
- /tasks
- /expenses
- /health

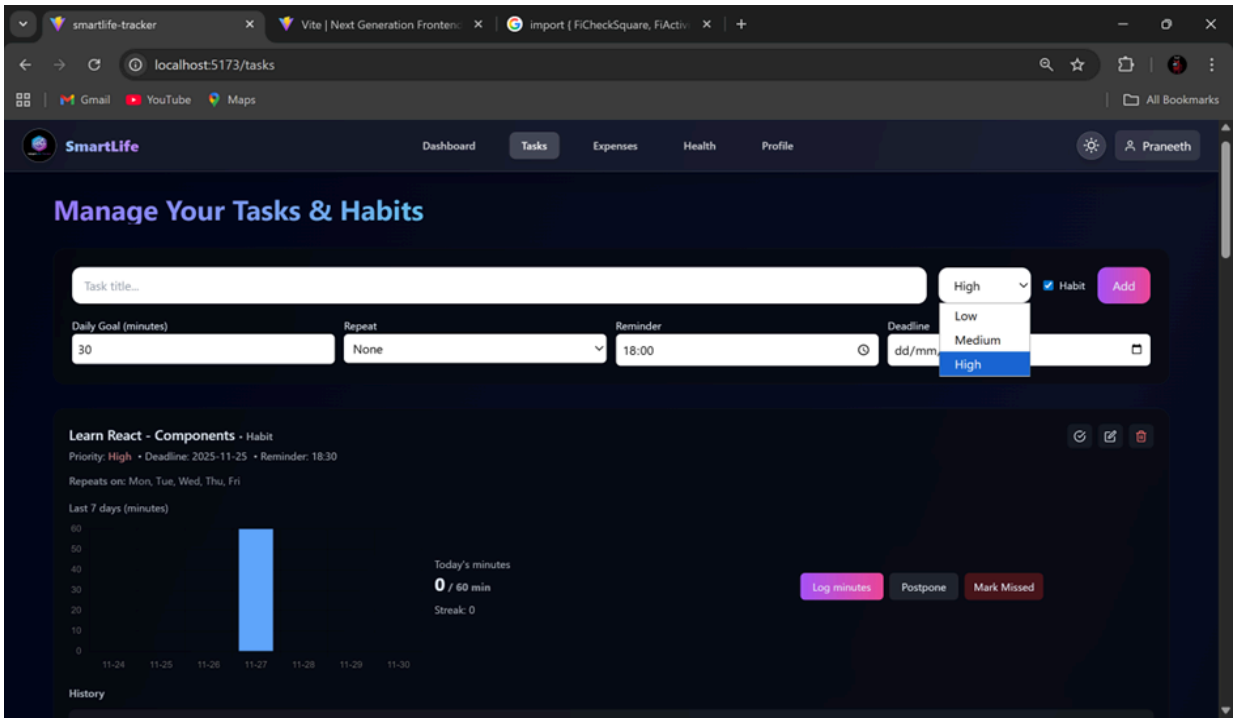
9.1 Result Screenshots

Dashboard Overview



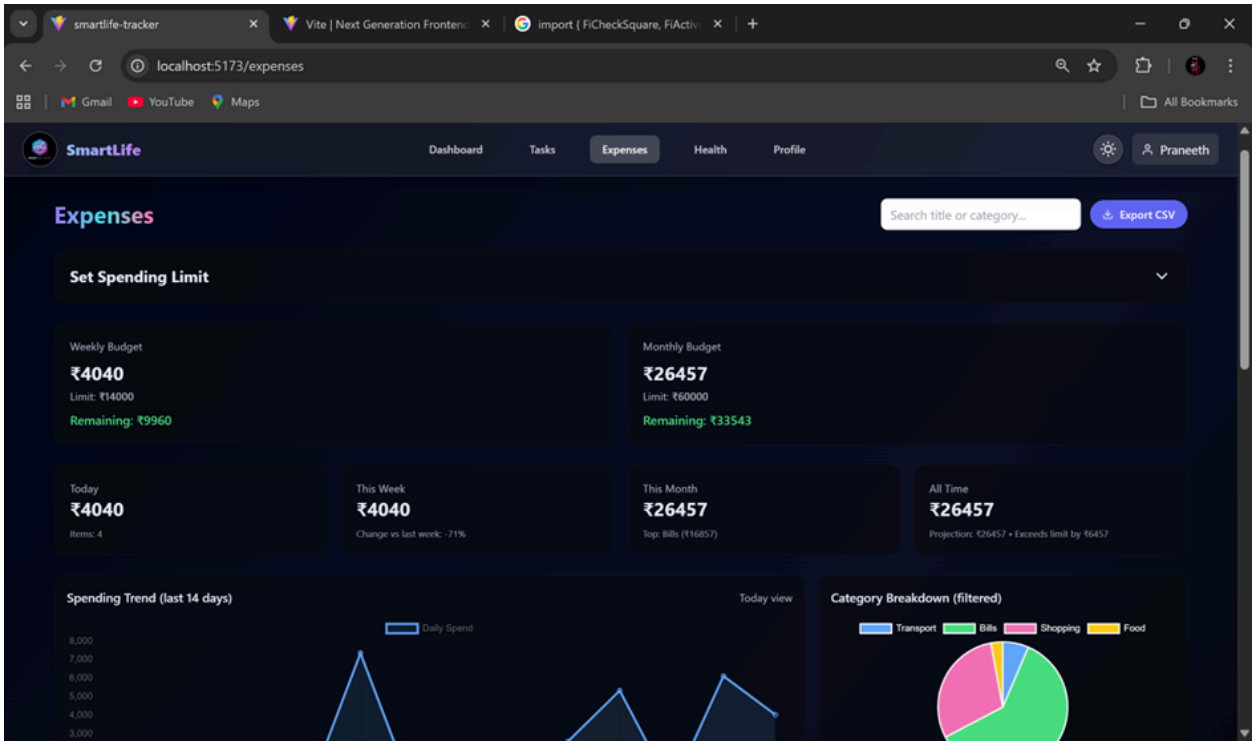
Caption: Displays a summary of the user's day including tasks, habits, expenses, and health insights.

Tasks Page



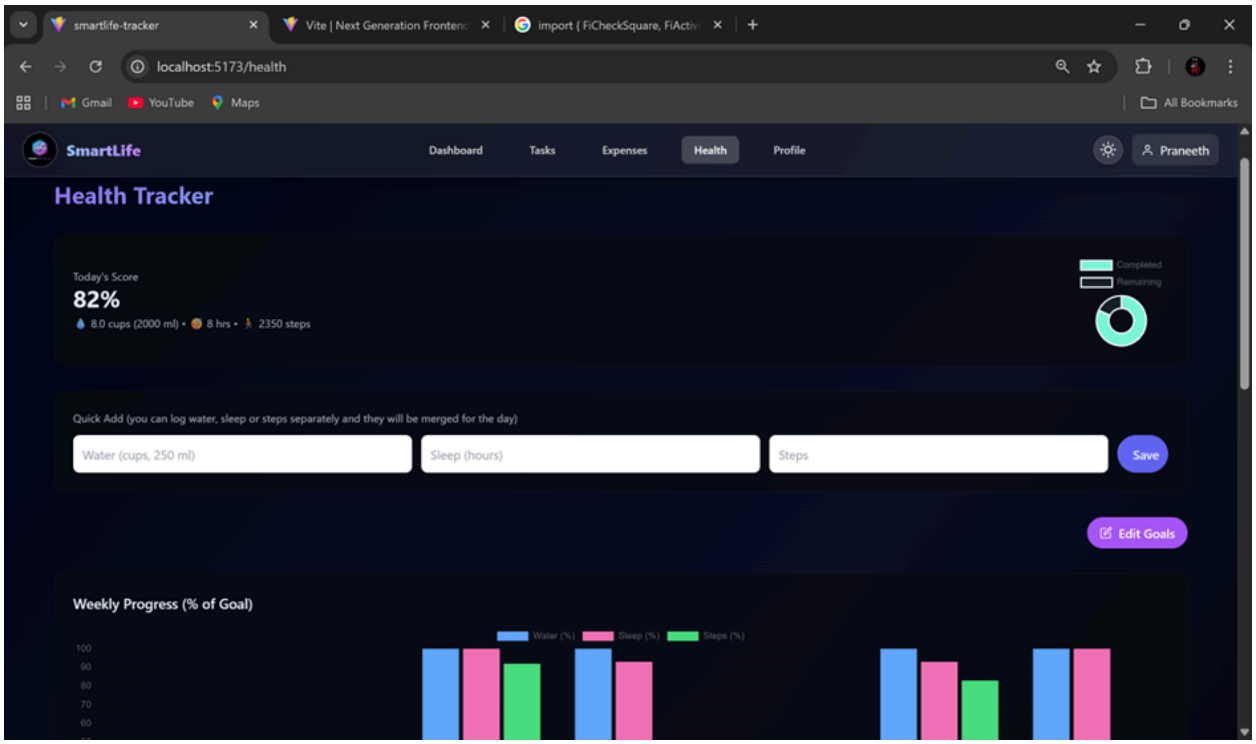
Caption: Shows task creation, priority selection, deadlines, and task completion management.

Expenses with Spending Limit



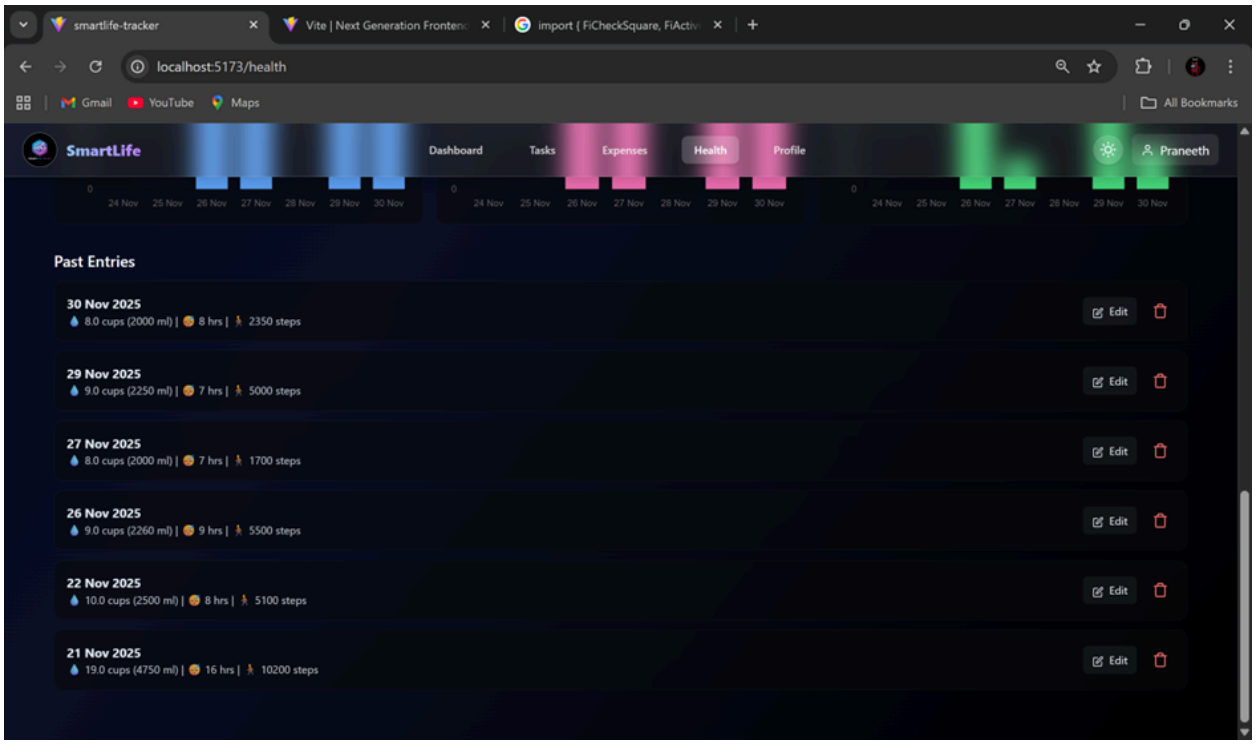
Caption: Displays expense logging, categorization, and weekly/monthly spending limit alerts.

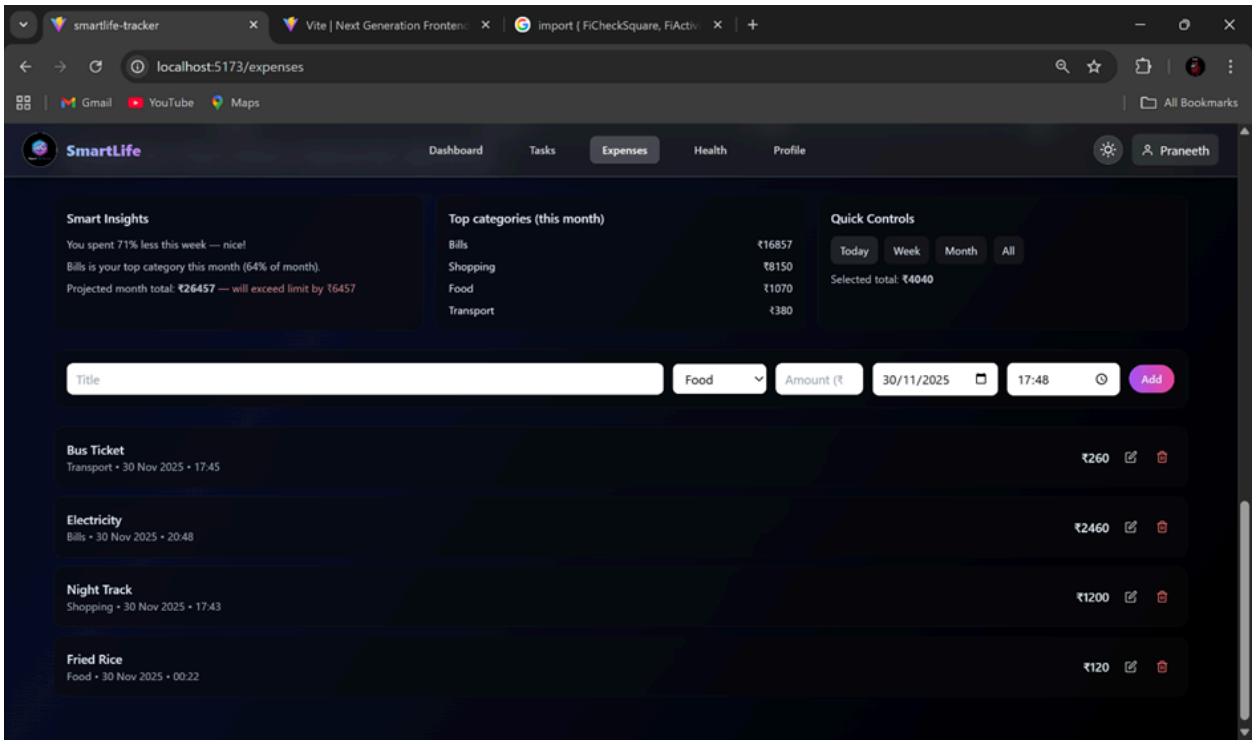
Health Tracker



Caption: Shows daily health inputs such as water intake, sleep duration, and step count.

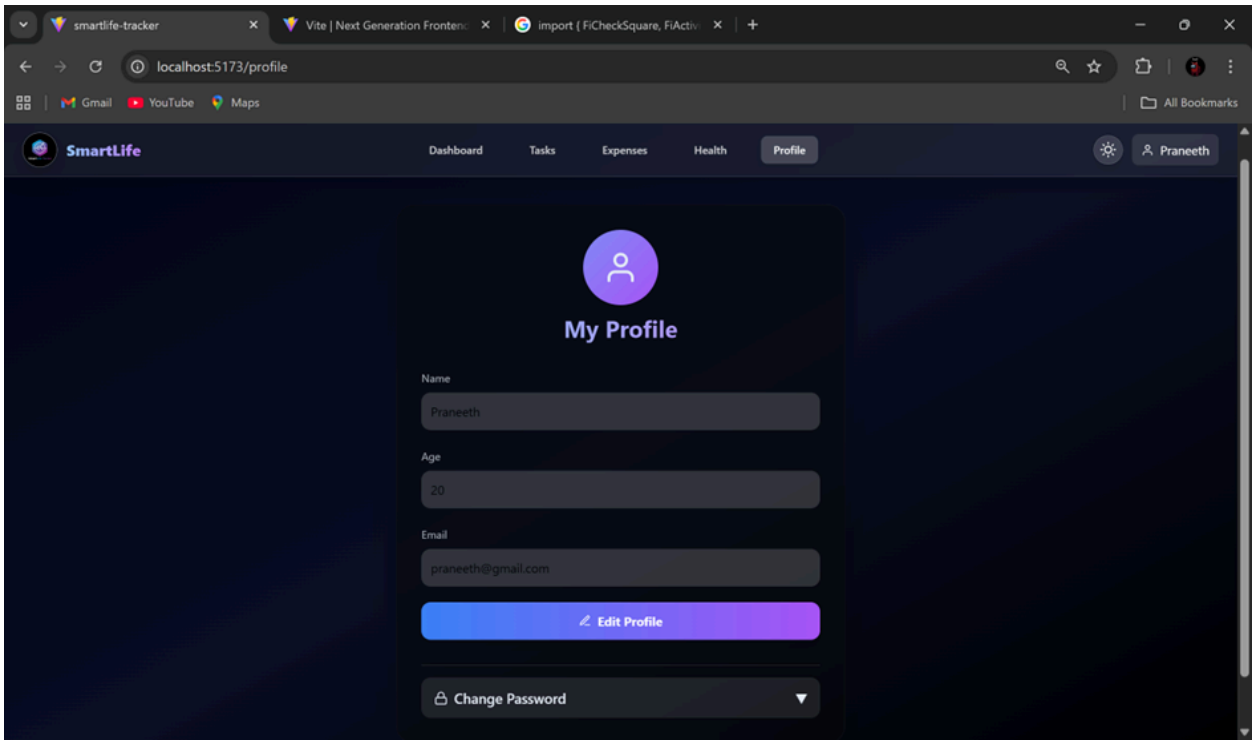
History Page (Expenses & Health)





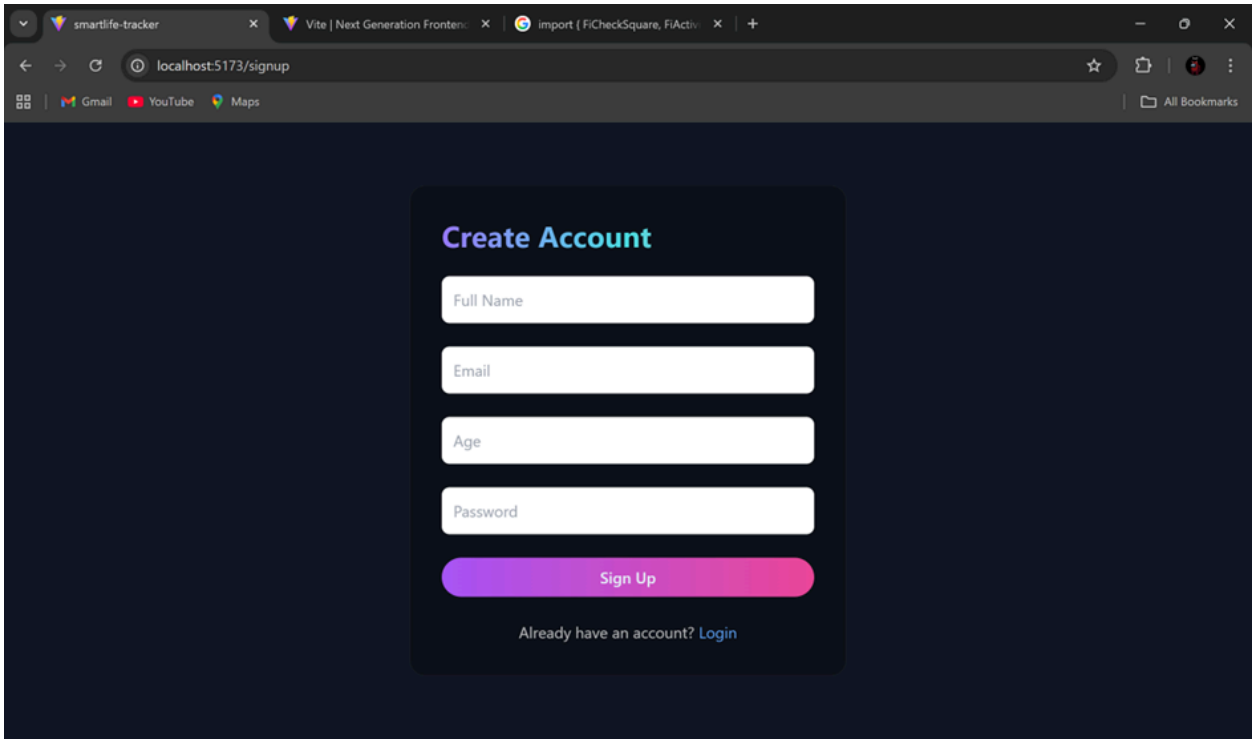
Caption: Provides detailed history of past expenses and health logs stored over time.

Profile Page



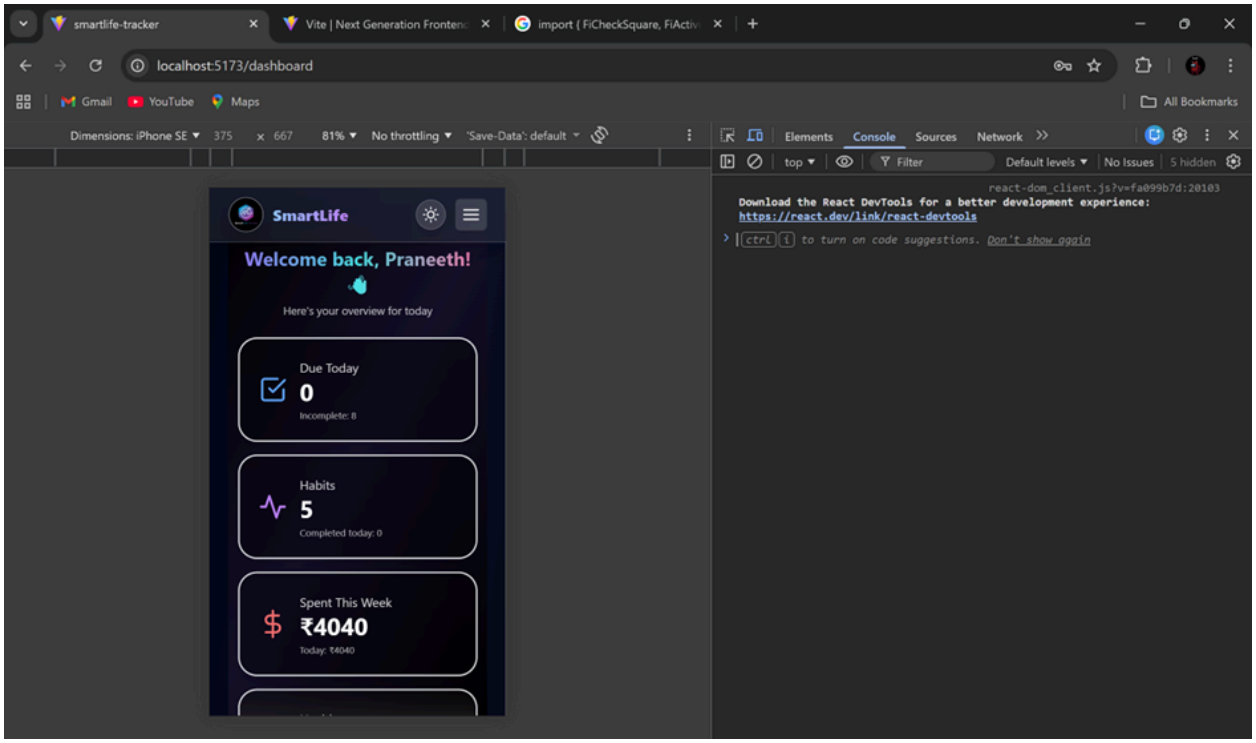
Caption: Includes user information, profile editing options, and account management features.

SignUp Page



Caption: Displays the user registration interface for creating a new account.

Mobile Responsive Interface



Caption: Shows how the SmartLife Tracker layout adapts to smaller screen sizes for mobile devices.

10. Project Demo links

Github Repo: https://github.com/Nallanipraneeth66/SmartLife-Tracker

Code Explanation Video Link:
https://github.com/Nallanipraneeth66/SmartLife-Tracker/Team-5_SmartLife Tracker_Code Explanation.mp4

Project Demonstration Video Link:
https://github.com/Nallanipraneeth66/SmartLife-Tracker/Team-5_SmartLife Tracker_Demonstration.mp4