Project Name:

“My Health Tracker”

Table of Contents

List of Tables

List of Figures

Executive Summary

My Health Tracker is a mobile web app which helps people keep track of their wellness routines and track them daily. The system, which was created on the basis of the MERN stack (MongoDB, Express, React, Node.js), allows one to record the meals, the amount of water consumed, mood, and physical activity. It offers an interactive graph and dashboard where users can see logs and progress over time.

Under the project, the delivery is well organised into two phases:

- Phase 1 was channelled towards concept design, value proposition, framework, and technology planning.
- Phase 2 went into end-to-end development, testing, documentation, presentation, and screencast delivery.

1. Phase 1: Conceptualisation and Planning

1.1 Project Idea

The goal of the app is to promote personal health by making them track their habits and visualise trends. The original idea was to present a user-friendly digital journal, which not only saves all the inputs but also allows the user to discover trends within his or her lifestyle.

1.2 Who will be reached?

- Adults between the ages of 18 and 45
- Fitness enthusiasts or fitness gurus, and those who are quite tech-savvy and interested in mental wellness
- Individuals who want to develop healthy habits

1.3 User Value

- Graphical trends through graphs
- Access on the move and desktops
- Information on the correlation of mood and habit
- all the data in a single clean cloud area

1.4 Technical Stack

Table 1 below summarises technologies that were selected at each of the layers of application and justifications on how they fit into the development of a scalable, full-stack health monitoring web application.

Table 1: Technical Stack (source: self-made)

Component	**Technology Used**	**Rationale**
Frontend	React.js, Chart.js	Responsive UI with interactive visuals
Backend	Node.js, Express	Lightweight RESTful API server
Database	MongoDB Atlas (Cloud)	Flexible schema, fast NoSQL storage
Hosting	(Local for dev, deploy-ready)	Easily deployable on Netlify/Render
Tools	Axios, dotenv, GitHub	API calls, config management, versioning

2.1 System Architecture

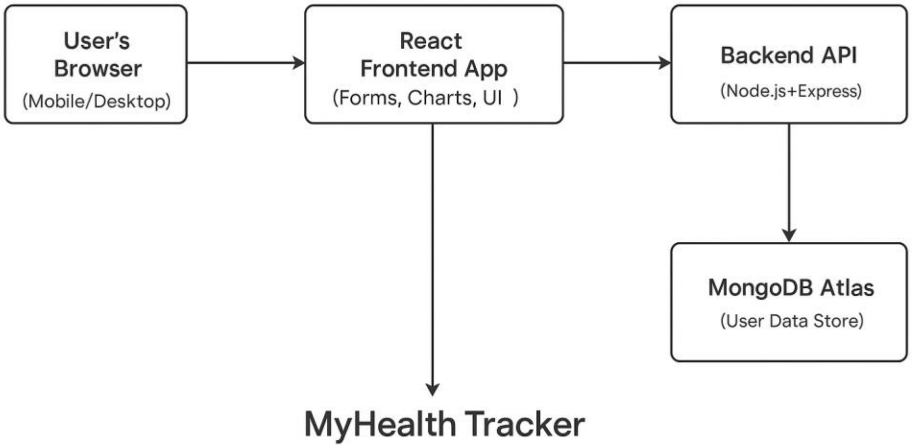

Figure 1: Architecture Overview (source: self-made)

Figure 1 shows the complete system architecture, the interaction between the cloud database services, the frontend, and the backend.

2. Phase 2: Development and Testing

2.2 Key Features Implemented

- **Health Log Form:** Capture daily input (meal, exercise, mood, water)
- **Live Chart Visualisation:** Weekly water intake visualised using Chart.js
- **Table View:** All entries are listed in a filterable, scrollable table
- **Delete Logs:** Remove entries dynamically with instant UI updates
- **Persistent Storage:** Data saved in MongoDB Atlas, retrieved via API

2.3 Code Structure

```
/MyHealthTracker
├── /client            → React frontend
│   └── /components    → LogForm, LogChart, LogTable
├── /server            → Express backend
│   ├── /routes
│   ├── /controllers
│   ├── /models
│   └── server.js
├── /docs_phase2       → Slides and Screencast
└── README.md
```

2.4 Testing and validation

- **Form Validation:** Makes filling out a form compulsory before submitting it
- **Chart Reactivity:** As logs are updated, the chart also updates immediately
- **API/Back-end Test:** All the endpoints were tested on browsers and Postman
- **MongoDB Atlas Integration:** Confirmed document Creation and destruction
- **Cross-browser checked:** Compatible with Chrome, Edge, and Firefox

3. Presentation and Evidence

- **PowerPoint:** has an 8-slide concept, feature, architecture, tools and screenshots breakdown
- **Screencast:** 2.38 minutes long, narrated walkthrough of all functionality
- **Source Code:** Well-structured GitHub repository with README, .env not included

4. Delivery Deployment

- Working files were all comprised: client, server, and docs_phase2
- An instance of MongoDB is connected and checked
- We can use the App to be deployed to Netlify (frontend, backend)
- Sold as a ready-to-give-to-someone solution when scaling in the future

5. Conclusion

MyHealth Tracker attains the objectives of developing both phases 1 and 2. In planning, development, and delivery, this end-to-end wellness app is highly aligned with contemporary web app development and a user-driven development process. The plug-ins that have been delivered can be hosted on the web or can be improved with the functions of logging in/authentication, reminders and mobile enabling. The client can now look at the attached presentation and video as a visual demonstration and use the codebase to make a technical review.

6. Phase 3: Evaluation & Future Enhancements

6.1 Summary of Evaluation

MyHealth Tracker application allows carrying out all important functions:

- Records health input (meals, mood, water, exercise)
- Pictures it in tabular and graphical format
- Allows erasing of records
- Saves all of the data securely in MongoDB Atlas

Unit testing was conducted based on cross-browser compatibility and a responsive frontend. Manual tests using API endpoints through Postman were delivered successfully, as well as real-time updates.

6.2 6.2 Limitations

- No login/authentication mechanism (it does not protect privacy, as users are in a joint space)
- None: no filtering, date-range selectors
- App is deployed online (not implemented locally)

6.3 Planned Improvements

Table 2: Planned Improvements (source: self-made)

Feature	**Future Plan**
Login System	Enable secure accounts using Firebase/Auth0.
UI Styling	Apply a modern UI with Material UI or Tailwind
Date Filters	Let users filter logs by week/month.
Deployment	Host frontend on Netlify, backend on Render
Mobile Optimization	Improve layout for smaller screens.

6.4 Reflection

This project managed to prove the use of the full-stack using the MERN stack. The important lessons were to embrace both frontend and backend logic, debug the programmatic interface, and utilise cloud services such as MongoDB Atlas. It was also an experience that enhanced skills in collaboration on GitHub, component-based designs and responsive web architecture.