StepHabit Capstone Report

A comprehensive technical and product deep dive

Prepared by: Engineering Team

Date: December 16, 2025

Abstract

StepHabit is a full-stack habit-building and productivity platform designed to help users convert long-term goals into consistent daily action. The system integrates habit tracking, task management, intelligent scheduling, and AI-assisted coaching within a unified experience. This report presents a detailed technical and product-level analysis of StepHabit, covering system architecture, data modeling, backend services, frontend workflows, and implementation decisions. The document serves both as a formal capstone submission and as a long-term reference for future contributors and system extensions.

Contents

1 Introduction **4**
1.1 System Overview 4
1.2 Objectives 4
1.3 Scope 5
1.4 Scope of the Project 5
1.5 Structure of the Report 5

2 System Architecture **6**
2.1 Overview of System Components 6
2.2 Technology Stack and Frameworks 6
2.3 Development and DevOps Tools 7

3 Database Design **9**
3.1 Database Choice 9
3.2 Requirements Gathering Process 9
3.3 Functional Requirements 9
3.4 Non-Functional Requirements 9
3.5 Success Criteria 9

4 System Architecture **10**
4.1 High-Level Architecture 10
4.2 Technology Stack 10
4.3 Backend Architecture 10
4.4 Frontend Architecture 10
4.5 AI and Intelligent Components 10
4.6 Deployment Architecture 10

5 Backend Design and Implementation **11**
5.1 Application Entry Point 11
5.2 Routing and Middleware 11
5.3 Controller Layer 11
5.4 Service Layer 11
5.5 Database Access Layer 11
5.6 Error Handling Strategy 11

6 Data Modeling **12**
6.1 Database Design Overview 12
6.2 Core Entities 12
6.3 Relationships and Associations 12
6.4 Data Integrity Constraints 12
6.5 Data Lifecycle Management 12

7 API Design **13**
7.1 REST API Principles . . . 13
7.2 Authentication and Authorization APIs . . . 13
7.3 Habit and Task APIs . . . 13
7.4 Scheduling APIs . . . 13
7.5 AI Service APIs . . . 13
7.6 Messaging and Notification APIs . . . 13

8 AI and Coaching System **14**
8.1 AI System Overview . . . 14
8.2 Prompt Engineering Strategy . . . 14
8.3 Assistant Memory Design . . . 14
8.4 Safety and Guardrails . . . 14
8.5 Future AI Extensions . . . 14

9 Frontend Design and Implementation **15**
9.1 UI Framework and Design System . . . 15
9.2 Application Layout and Navigation . . . 15
9.3 Routing and State Management . . . 15
9.4 Core Screens and User Flows . . . 15
9.5 Accessibility and Responsiveness . . . 15

10 Security and Privacy **16**
10.1 Authentication Mechanisms . . . 16
10.2 Authorization Model . . . 16
10.3 Input Validation . . . 16
10.4 Data Privacy and Compliance . . . 16
10.5 Secrets Management . . . 16

11 Scheduling and Productivity Workflows **17**
11.1 Habits vs Tasks . . . 17
11.2 Smart Scheduling . . . 17
11.3 Calendar Integration . . . 17
11.4 Progress Tracking . . . 17
11.5 Achievements and Motivation . . . 17

12 Community and Communication **18**
12.1 Friendship Model . . . 18
12.2 Group Challenges . . . 18
12.3 Messaging System . . . 18
12.4 Notifications . . . 18
12.5 Moderation Policies . . . 18

13 Testing and Quality Assurance **19**
13.1 Testing Strategy . . . 19
13.2 Unit Testing . . . 19
13.3 Integration Testing . . . 19
13.4 End-to-End Testing . . . 19
13.5 Performance Testing . . . 19

14 Deployment and Operations **20**
14.1 Environment Configuration . . . 20
14.2 Containerization . . . 20
14.3 CI/CD Pipeline . . . 20
14.4 Scaling Considerations . . . 20
14.5 Monitoring and Logging . . . 20

15 Evaluation and Metrics **21**
15.1 Product Metrics . . . 21
15.2 Technical Metrics . . . 21
15.3 AI Performance Metrics . . . 21
15.4 User Engagement Analysis . . . 21

16 Future Work **22**
16.1 Feature Enhancements . . . 22
16.2 AI Improvements . . . 22
16.3 Scalability Roadmap . . . 22
16.4 Research Directions . . . 22

17 Ethical Considerations **23**
17.1 Responsible AI Use . . . 23
17.2 User Privacy . . . 23
17.3 Bias and Fairness . . . 23
17.4 Transparency and Consent . . . 23

18 Conclusion **24**
18.1 Summary of Contributions . . . 24
18.2 Lessons Learned . . . 24
18.3 Final Remarks . . . 24

1 Introduction

1.1 System Overview

StepHabit is a secure, AI-assisted productivity and habit-building platform designed to help users transform long-term goals into consistent daily routines. Built with a strong emphasis on structure, motivation, and sustainability, the system enables users to create, manage, and track habits, tasks, and schedules within a unified environment. A secure registration and authentication process ensures account integrity, while personalized dashboards allow users to monitor progress, streaks, and achievements over time.

Once authenticated, users can define habits with daily goals, manage tasks with durations and deadlines, and organize their time through calendar-based planning and busy-block scheduling. The platform provides visual feedback through planners, charts, and progress logs, helping users clearly understand how their daily actions align with broader objectives. Notifications, reminders, and achievement milestones reinforce engagement and encourage consistency.

Beyond traditional productivity tools, StepHabit integrates artificial intelligence to provide personalized coaching and guidance. An AI-driven assistant analyzes user input and context to generate structured habit plans, refine habit ideas, and offer actionable suggestions. By maintaining assistant memory tied to individual users, the system delivers increasingly relevant feedback and supports reflective, goal-oriented decision making over time.

User data is stored securely and is accessible only to authenticated users, with all core features scoped to individual accounts to ensure privacy. Social features such as friendships, messaging, and group challenges are implemented within controlled boundaries, allowing users to engage in accountability-driven interactions without compromising personal data. Calendar integrations further enhance the experience by preventing scheduling conflicts between habits, tasks, and external commitments.

In summary, StepHabit combines secure account management, structured planning, social accountability, and AI-powered coaching to deliver a comprehensive and intelligent productivity experience. The platform demonstrates how modern web technologies and artificial intelligence can be thoughtfully integrated to support sustainable habit formation and long-term personal growth.

1.2 Objectives

The primary objective of this project is to design and develop a secure, AI-assisted productivity and habit-building platform that enables users to translate long-term goals into consistent daily actions. The system aims to provide a structured yet flexible environment in which users can create, manage, and track habits, tasks, and schedules, while receiving intelligent guidance to improve planning and execution. Through AI-powered coaching, the platform supports users in refining habit ideas, breaking goals into actionable steps, and maintaining motivation over time.

In addition, the project seeks to deliver an intuitive and user-friendly interface, secure account management through authenticated access, and personalized dashboards that visu-

alize progress, streaks, and achievements. By combining scheduling tools, progress tracking, and social accountability mechanisms, StepHabit encourages sustainable behavior change rather than short-term productivity bursts.

Secondary objectives include building a scalable full-stack web application using modern frameworks, ensuring data security and user privacy, and establishing a robust architectural foundation that can support future extensions such as mobile clients, advanced analytics, or enhanced AI-driven personalization.

1.3 Scope

The primary objective of this project is to design and develop a secure, AI-assisted productivity and habit-building platform that enables users to translate long-term goals into consistent daily actions. The system aims to provide a structured yet flexible environment in which users can create, manage, and track habits, tasks, and schedules, while receiving intelligent guidance to improve planning and execution. Through AI-powered coaching, the platform supports users in refining habit ideas, breaking goals into actionable steps, and maintaining motivation over time.

In addition, the project seeks to deliver an intuitive and user-friendly interface, secure account management through authenticated access, and personalized dashboards that visualize progress, streaks, and achievements. By combining scheduling tools, progress tracking, and social accountability mechanisms, StepHabit encourages sustainable behavior change rather than short-term productivity bursts.

Secondary objectives include building a scalable full-stack web application using modern frameworks, ensuring data security and user privacy, and establishing a robust architectural foundation that can support future extensions such as mobile clients, advanced analytics, or enhanced AI-driven personalization.

1.4 Scope of the Project

1.5 Structure of the Report

2 System Architecture

2.1 Overview of System Components

The StepHabit platform is designed using a modular system architecture, in which each major component is responsible for a clearly defined set of responsibilities. This architectural approach improves clarity, maintainability, and scalability, while allowing individual components to evolve independently. Together, these components form a cohesive, secure, and intelligent productivity system that supports habit formation, task management, and goal-oriented behavior.

The frontend application serves as the primary interaction layer between users and the system. It provides interfaces for user registration and authentication, habit and task creation, calendar-based planning, and progress tracking. Through dashboards, planners, and visual analytics, users can monitor habits, streaks, achievements, and upcoming tasks. Social features such as messaging, group challenges, and notifications are also accessible through the frontend, enabling accountability-driven engagement in a user-friendly and responsive interface.

The backend server implements the core business logic and coordinates communication between the frontend, the database, and external services. It manages authentication and authorization, enforces data ownership and privacy rules, and processes all user actions, including habit updates, task scheduling, progress logging, notifications, and community interactions. The backend also exposes RESTful APIs that provide a stable and extensible interface for current and future clients.

The database layer is responsible for persistent data storage and integrity. It stores user profiles, habits, tasks, schedules, progress records, achievements, notifications, assistant memory, and social relationships. The relational schema is designed to support clear ownership boundaries between users and their data, while enabling efficient queries for analytics, dashboards, and planner views.

An AI-powered coaching component enhances the platform by providing intelligent guidance and personalization. This service analyzes user-provided habit ideas and contextual data to generate structured habit plans, refine goal descriptions, and offer actionable suggestions. Assistant memory is maintained on a per-user basis, allowing the system to deliver increasingly relevant and consistent coaching over time. AI outputs are evaluated and stored where appropriate to support reflection and future interactions.

Together, these components form a robust and extensible system that integrates structured planning, progress visibility, social accountability, and AI-driven guidance. StepHabit combines modern web technologies with intelligent feedback mechanisms to deliver a productivity platform that supports sustainable habit formation and long-term personal growth.

2.2 Technology Stack and Frameworks

The StepHabit platform utilizes a modern, modular technology stack designed to ensure performance, scalability, maintainability, and ease of future extension across all layers of the system. Each technology was selected to support rapid development while maintaining architectural clarity and long-term sustainability.

Frontend React (Vite): A modern JavaScript library used to build responsive and interactive user interfaces. Vite is employed as the build tool to enable fast development cycles and optimized production bundles. CoreUI: A component-based UI framework used to implement consistent layouts, navigation structures, dashboards, and data visualization elements across the application.

Backend Node.js with Express: Node.js serves as the runtime environment for the backend, while Express provides a lightweight and flexible framework for handling HTTP requests, routing, and middleware composition. Sequelize ORM: Used to manage relational data interactions with PostgreSQL, providing model definitions, associations, and schema synchronization. JWT (JSON Web Tokens): Employed for secure authentication and authorization, ensuring that protected API endpoints are accessible only to authenticated users. Bcrypt: Used for secure password hashing and credential protection. Socket-Based Messaging (Planned): The architecture supports real-time features such as notifications and messaging, allowing future integration of WebSocket-based communication if required.

Database PostgreSQL: A robust open-source relational database used to store structured data, including user profiles, habits, tasks, schedules, progress logs, achievements, notifications, and social relationships. PostgreSQL's reliability and relational capabilities support data integrity and complex querying needs.

AI Integration LangChain with Large Language Models (Anthropic): Used to implement AI-assisted habit coaching, including habit plan generation, idea refinement, and contextual guidance. LangChain provides an abstraction layer that simplifies prompt management, model configuration, and future AI provider replacement.

Environment-Based Configuration: Sensitive credentials and environment-specific variables are managed through environment files, supporting secure and flexible configuration across development and production environments.

2.3 Development and DevOps Tools

The development of the StepHabit platform relied on a set of modern development and DevOps tools to support efficient implementation, testing, collaboration, and deployment. These tools contributed to code quality, system reliability, and developer productivity throughout the project lifecycle.

Visual Studio Code (VS Code): Used as the primary integrated development environment for both frontend and backend development. VS Code provides rich language support, debugging tools, and extensions that streamline full-stack development.

Postman: Employed extensively during backend development for testing RESTful API endpoints, validating request and response payloads, and debugging authentication and business logic workflows.

Git and GitHub: Used for version control, source code management, and collaboration. Git enables systematic tracking of code changes, while GitHub supports repository hosting, issue tracking, and pull-request-based development workflows.

Docker and Docker Desktop: Used to containerize the application's backend services, frontend client, and database, ensuring consistent environments across local development and deployment. Docker Compose simplifies orchestration of multi-container setups.

Navicat: Utilized as a database administration and visualization tool for managing the

PostgreSQL database, inspecting tables and relationships, and validating data integrity during development and testing.

ESLint and Prettier: Applied to enforce consistent coding standards, detect potential issues early, and maintain readability and maintainable code across the project.

Email and Notification Monitoring Tools: Used to test and monitor email-based features such as account verification and notification delivery, ensuring reliability and correctness of user-facing communication workflows.

API Documentation Tools (Swagger UI): Used to document and interactively test API endpoints, improving transparency of the API surface and simplifying development, debugging, and future system integration.

3 Database Design

3.1 Database Choice

PostgreSQL was chosen for this project because of its excellent support for relational data, adherence to ACID principles, and support for TypeORM integration. PostgreSQL's support for complex querying and indexing provides an effective solution to managing the complex relationships between users, posts, and interactions (likes, comments).

3.2 Requirements Gathering Process

3.3 Functional Requirements

3.4 Non-Functional Requirements

3.5 Success Criteria

4 System Architecture

4.1 High-Level Architecture

4.2 Technology Stack

4.3 Backend Architecture

4.4 Frontend Architecture

4.5 AI and Intelligent Components

4.6 Deployment Architecture

5 Backend Design and Implementation

5.1 Application Entry Point

5.2 Routing and Middleware

5.3 Controller Layer

5.4 Service Layer

5.5 Database Access Layer

5.6 Error Handling Strategy

6 Data Modeling

6.1 Database Design Overview

6.2 Core Entities

6.3 Relationships and Associations

6.4 Data Integrity Constraints

6.5 Data Lifecycle Management

7 API Design

7.1 REST API Principles

7.2 Authentication and Authorization APIs

7.3 Habit and Task APIs

7.4 Scheduling APIs

7.5 AI Service APIs

7.6 Messaging and Notification APIs

8 AI and Coaching System

8.1 AI System Overview

8.2 Prompt Engineering Strategy

8.3 Assistant Memory Design

8.4 Safety and Guardrails

8.5 Future AI Extensions

9 Frontend Design and Implementation

9.1 UI Framework and Design System

9.2 Application Layout and Navigation

9.3 Routing and State Management

9.4 Core Screens and User Flows

9.5 Accessibility and Responsiveness

10 Security and Privacy

10.1 Authentication Mechanisms

10.2 Authorization Model

10.3 Input Validation

10.4 Data Privacy and Compliance

10.5 Secrets Management

11 Scheduling and Productivity Workflows

11.1 Habits vs Tasks

11.2 Smart Scheduling

11.3 Calendar Integration

11.4 Progress Tracking

11.5 Achievements and Motivation

12 Community and Communication

12.1 Friendship Model

12.2 Group Challenges

12.3 Messaging System

12.4 Notifications

12.5 Moderation Policies

13 Testing and Quality Assurance

13.1 Testing Strategy

13.2 Unit Testing

13.3 Integration Testing

13.4 End-to-End Testing

13.5 Performance Testing

14 Deployment and Operations

14.1 Environment Configuration

14.2 Containerization

14.3 CI/CD Pipeline

14.4 Scaling Considerations

14.5 Monitoring and Logging

15 Evaluation and Metrics

15.1 Product Metrics

15.2 Technical Metrics

15.3 AI Performance Metrics

15.4 User Engagement Analysis

16 Future Work

16.1 Feature Enhancements

16.2 AI Improvements

16.3 Scalability Roadmap

16.4 Research Directions

17 Ethical Considerations

17.1 Responsible AI Use

17.2 User Privacy

17.3 Bias and Fairness

17.4 Transparency and Consent

18 Conclusion

18.1 Summary of Contributions

18.2 Lessons Learned

18.3 Final Remarks