TaskWise

TO DO AI AGENT

A Project Report

By

Rechael Lopes

DECLARATION

I, Rechael Vincent Lopes affirm that the attached work is entirely my own, except where the words or ideas of other writers are specifically acknowledged in accordance with accepted APA citation conventions. This project is specifically made for my personal use. I acknowledge that I have revised, edited, and proofread this paper, and I certify that I am the author of this paper. Any assistance I received in its preparation is fully and properly acknowledged and disclosed. I have also cited any sources from which I used data, ideas, theories, or words, whether quoted directly or paraphrased. I further acknowledge that this paper has been prepared by myself specifically for this project.

ABSTRACT

This document highlights the project plan for creating a personal To Do AI Agent “TaskWise” that is effective and efficient beyond just the basic applications. The plan covers a detailed Work Breakdown Structure with the technology used, workflow, and prototype. The designed project plan ensures the successful implementation of the personal To Do AI Agent with reduced risks, downtime, and discrepancies.

Table of Contents

CHAPTER 1: INTRODUCTION 1

Project Scope 2

Goals 2

Objectives 3

Work Breakdown Structure 4

CHAPTER 2: SYSTEM ANALYSIS 9

Proposed System 10

Requirement Analysis 10

Functional Requirements 10

Non-functional Requirements 11

Hardware Requirements 11

Software Requirements 11

Justification of the Platform 12

CHAPTER 3: SYSTEM DESIGN 13

Module Design 14

UML Diagrams 20

Data Flow Diagram 20

Component Diagram 22

CHAPTER 4: IMPLEMENTATION 24

CHAPTER 5: FUTURE WORK 28

Conclusion 29

Future Work 30

CHAPTER 6: REFERENCES AND APPENDIX 31

Table of Figures

Figure 1: Landing/Home Page 14
Figure 2: User Login & Task Panel 15
Figure 3: AI Assistance 15
Figure 4: Add Task 16
Figure 5: Tasks Page 16
Figure 6: Habits Page 17
Figure 7: Add Habit 17
Figure 8: Calander 18
Figure 9: Responsive Layout 19
Figure 10: Data Flow Diagram 21
Figure 11: Component Diagram 23

CHAPTER 1: INTRODUCTION

Project Scope

The **AI To-Do App "TaskWise"** aims to provide an intelligent task management system that integrates AI capabilities to enhance productivity. It allows users to create, view, update, and delete tasks efficiently, leveraging AI for task prioritization and smart recommendations.

Key Features:

- **Task Management:** CRUD operations (Create, Read, Update, Delete) for tasks.
- **AI Integration:** OpenAI API for natural language task input and smart suggestions.
- **Database Management:** PostgreSQL + Drizzle ORM for storing tasks.
- **User Interaction:** Command-line or web-based interface.
- **Authentication (Future Scope):** User login and task sync across devices.

The AI-based To Do Application will perform CRUD operations. It will communicate with AI using Natural Language Processing. AI agent should know what the user wants, and based on that the AI agent will interact with the database, and fetch data from the database.

Goals

The primary goal of this project is to develop an **AI-powered task management system** that enhances productivity through intelligent task handling, automation, and a seamless user experience.

1. **AI-Powered Task Management**
   - Enable users to add, view, update, and delete tasks.
   - Implement AI to process natural language input for task creation (e.g., "Remind me to pay my bills tomorrow").
2. **Intelligent Task Prioritization**
   - Automatically rank tasks based on urgency and importance.
   - Provide smart reminders and recommendations using AI.
3. **Efficient & Scalable Database Design**
   - Use PostgreSQL and Drizzle ORM for structured and optimized task storage.

- Ensure data consistency, reliability, and security.

4. **User-Friendly Interface**
   - Provide a CLI and/or web-based UI for task management.
   - Ensure an intuitive user experience with simple and effective interactions.
5. **Seamless AI Integration**
   - Leverage OpenAI API for intelligent task interpretation and automation.
   - Implement AI-driven suggestions for improving productivity.
6. **Error Handling & Performance Optimization**
   - Implement robust error handling for database operations and API calls.
   - Optimize response time and database queries for efficiency.
7. **Documentation & Maintainability**
   - Maintain a well-documented **Project Book** for future reference.
   - Ensure code is modular, clean, and follows best practices for maintainability.

Objectives

- Automate task entry using AI (e.g., "Remind me to submit my project next Monday").
- Provide **task prioritization** based on deadlines and importance.
- Offer a clean and efficient UI for seamless task management.
- Implement **database-driven** storage for persistence.
- Learn and document **best practices** in AI, database management, and full-stack development.

Work Breakdown Structure

WBS consists of 5 Phases each divided into multiple Levels. Each level then consists of individual Work Packages (Key Activities). Following is the breakdown for all the phases, levels, and work packages.

WORK BREAKDOWN STRUCTURE FOR THE AI TODO APPLICATION		
Phases	**Levels**	**Work Packages**
1. Planning and Initiation	1.1 Project Initiation	1.1.1 Define Project Scope and Objectives
		1.1.2 Identity key features and functionalities
	1.2 Research	1.2.1 Research AI, database, and ORM technologies
		1.2.2 Set up project repository and project environment
2. System Analysis and Requirement Gathering	2.1 Requirement Analysis	2.1.1 Define functional and non-functional requirements

	2.2 Elicitation	2.2.1 Identify and document hardware/software requirements
		2.2.2 Justify platform and technology stack selection
3. System Design	3.1 System Analysis	3.1.1 Define system architecture and workflow
		3.1.2 Design module interactions and process flow
	3.2 Data Analysis	3.2.1 Create UML diagrams
		3.2.2 Develop and finalize database schema
4. Backend Development	4.1 Backend Implementation	4.1.1 Set up Node.js and Express backend
		4.1.2 Install and configure Drizzle ORM with PostgreSQL

		4.1.3 Implement CRUD operations for task management
		4.1.4 Set up environment variables and database connections
		4.1.5 Implement error handling and logging mechanism
5. AI Integration	5.1 AI Implementation	5.1.1 Implement OpenAI API for NLP-based task input
		5.1.2 Develop AI-based task prioritization logic
	5.2 AI Optimization	5.2.1 Optimize AI response time and accuracy
		5.2.2 Fine-tune AI recommendations for better task management
6. Frontend and User Interface	6.1 Design and Implement	6.1.1 Design and develop web-based UI

		6.1.2 Implement user input handling and interaction
	6.2 Optimization	6.2.1 Ensure seamless integration with backend API
		6.2.2 Optimize UI for ease of use and responsiveness
7. Testing and Debugging	7.1 System Testing	7.1.1 Conducting UI testing for AI, database, and backend functions
		7.1.2 Perform database query optimization and load testing
		7.1.3 UI/UX testing
	7.2 System Debugging	7.2.1 Debug and optimize the overall system performance
8. Deployment and Documentation	8.1 Deployment and maintenance	8.1.1 Deploy system using Docker and Cloud services
		8.1.2 Ensure database and API connectivity

	8.2 Documentation	8.2.1 Write and maintain comprehensive documentation
		8.2.2 Maintain modular, clean, and scalable code
9. Future Enhancement	9.1 Future scope	9.1.1 Implement user authentication for multi-device task sync
		9.1.2 Add voice-based task input functionality
		9.1.3 Expand AI capabilities for more intelligent task recommendations

Table 1: WBS for the AI To-do App

CHAPTER 2: SYSTEM ANALYSIS

The **System Analysis** phase focuses on evaluating the proposed AI-powered To-Do App by identifying system requirements, defining constraints, and analyzing feasibility. This ensures the solution meets user needs while optimizing performance and scalability.

Proposed System

The **AI To-Do App "TaskWise"** is designed to enhance productivity through an AI-powered task management system. The system will allow users to add, update, delete, and retrieve tasks while integrating artificial intelligence for smart task prioritization and recommendations.

Key Features

- **Task Management:** CRUD (Create, Read, Update, Delete) operations for tasks.
- **AI Integration:** Natural Language Processing (NLP) for task input and smart recommendations.
- **Database Management:** PostgreSQL + Drizzle ORM for structured and optimized data storage.
- **User Interaction:** Web-based interface for easy usability.
- **Future Enhancements:** Authentication and multi-device sync for persistent user tasks.

Requirement Analysis

Functional Requirements

The system must provide the following functionalities:

- Users can add tasks using AI-powered natural language input.
- Users can update, delete, and mark tasks as completed.
- AI should prioritize tasks based on urgency and deadlines.

- Task data should be stored securely in a structured database.
- The system should provide reminders and recommendations.

Non-functional Requirements

Performance: AI response should be optimized for quick task processing. **Scalability:** The system should support multiple users in future iterations. **Security:** User data should be protected using authentication (future scope). **Maintainability:** Code should be modular, well-documented, and easy to extend. **Usability:** The interface (CLI or Web UI) should be intuitive and user-friendly.

Hardware Requirements

Development Machine: A laptop or PC with at least 8GB RAM, and 256GB SSD. **Server (for deployment):** Cloud-hosted VM or containerized environment.

Software Requirements

Operating System: Windows, macOS, or Linux **Development Tools:** Visual Studio Code, Docker, Postman **Programming Languages:** JavaScript/TypeScript (Node.js) **Database Management:** PostgreSQL with Drizzle ORM **AI Integration:** OpenAI API for natural language understanding **Version Control:** GitHub for code repository

Justification of the Platform

Component	Technology Used	Justification
Backend	Node.js + Express	Fast, scalable, and lightweight for API development
Database	PostgreSQL + Drizzle ORM	Reliable, and efficient for structured data storage
AI Integration	OpenAI API	Advanced NLP capabilities for smart task input
Frontend	Typescript, React, CSS	Simple and intuitive user interaction
Deployment	Docker, cloud-based	Ensures scalability, easy maintenance

CHAPTER 3: SYSTEM DESIGN

Module Design

1. Landing/Home Page

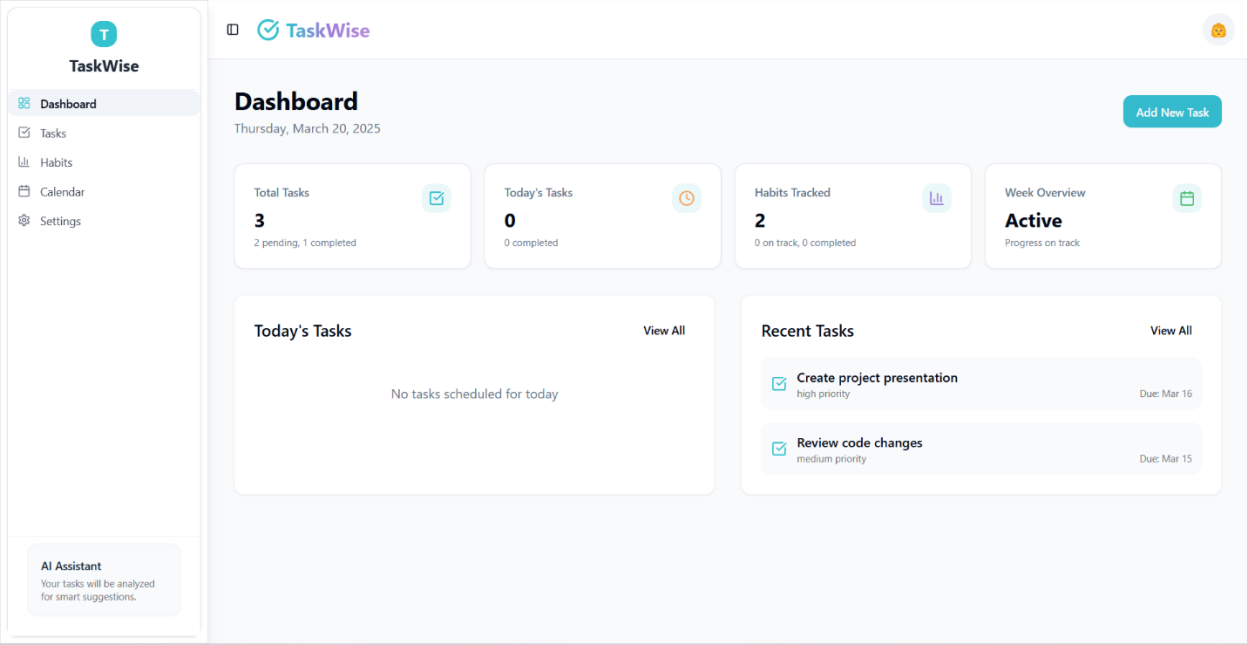

Figure 1: Landing/Home Page

2. User Login & Task Panel

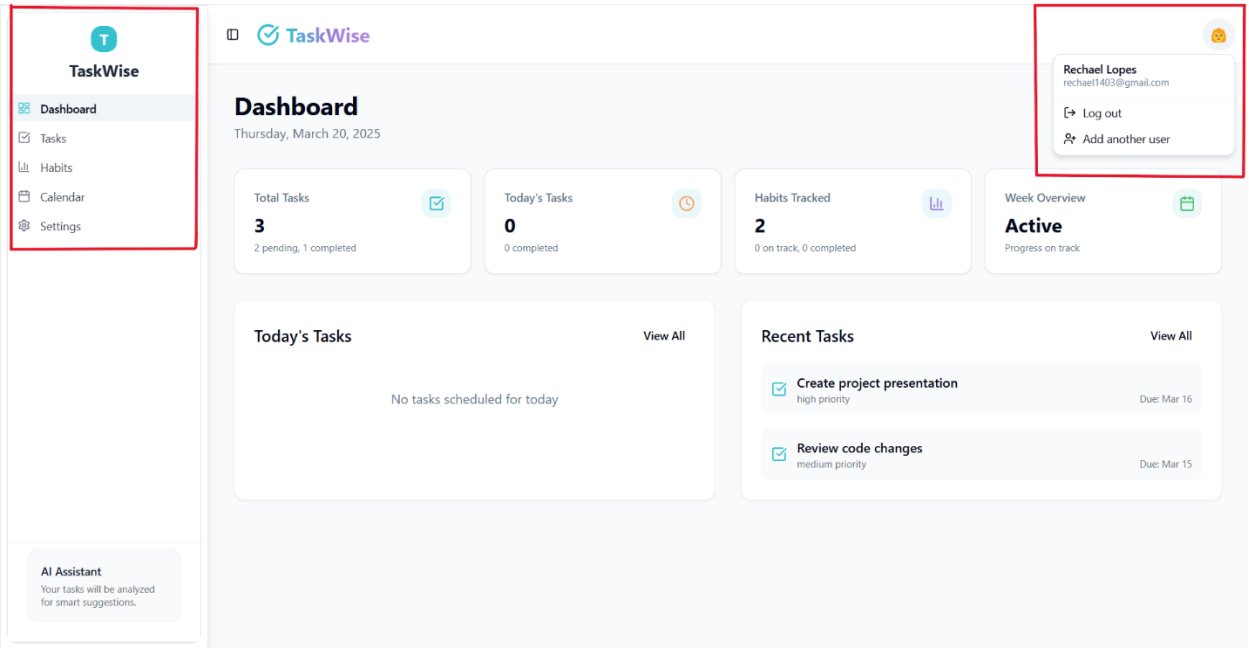

Figure 2: User Login & Task Panel

3. AI Assistance

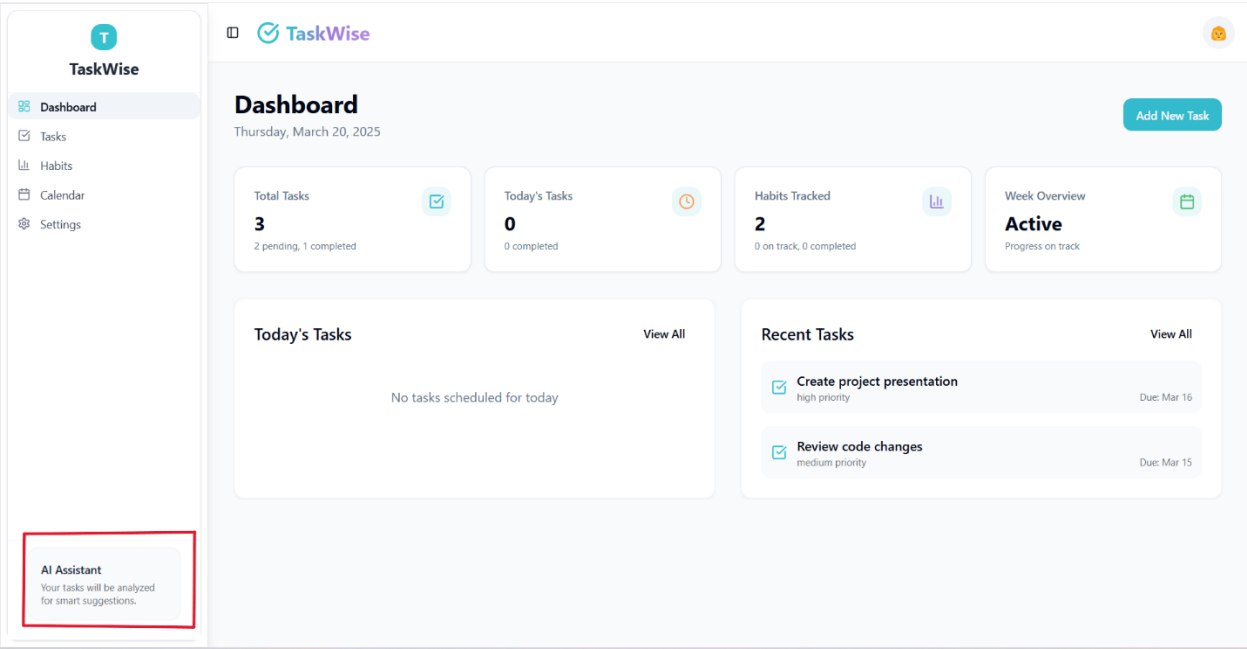

Figure 3: AI Assistance

4. Add New Task

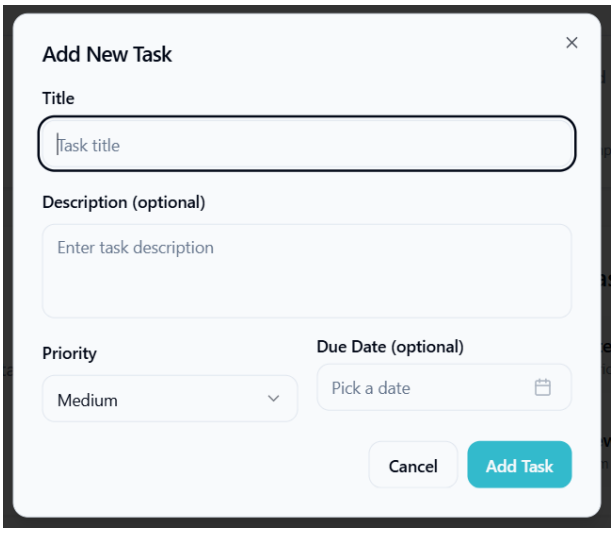

Figure 4: Add Task

5. Tasks Page

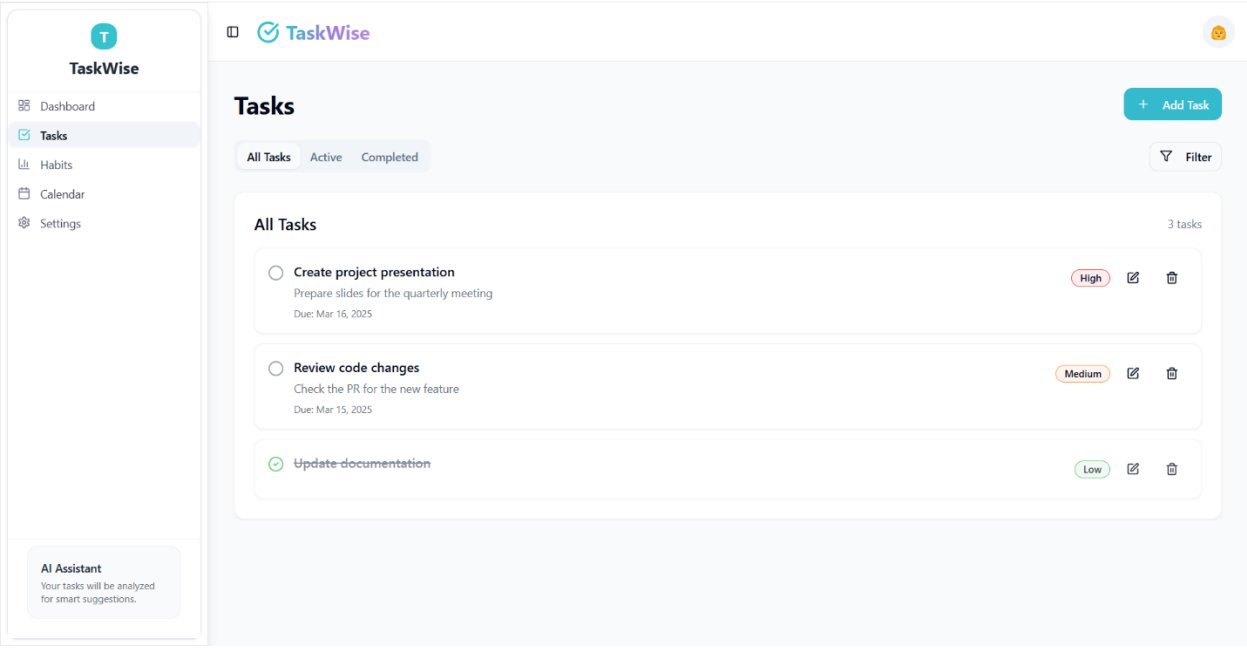

Figure 5: Tasks Page

6. Habits Page

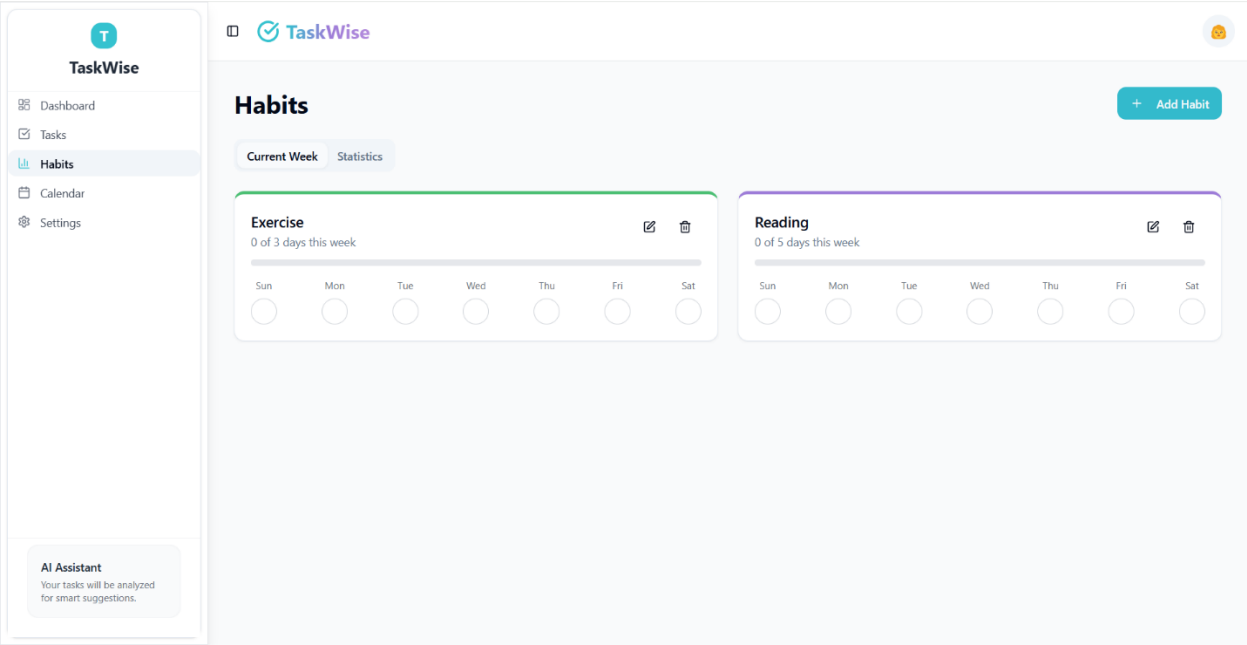

Figure 6: Habits Page

7. Add New Habit

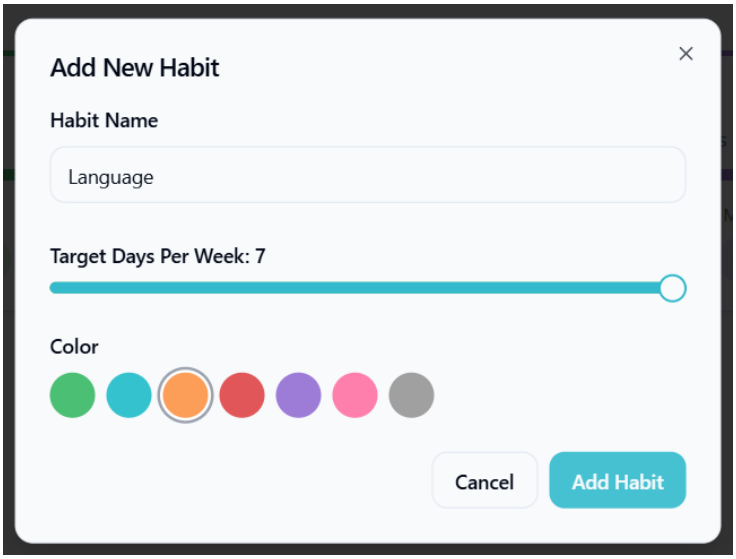

Figure 7: Add Habit

8. Calander

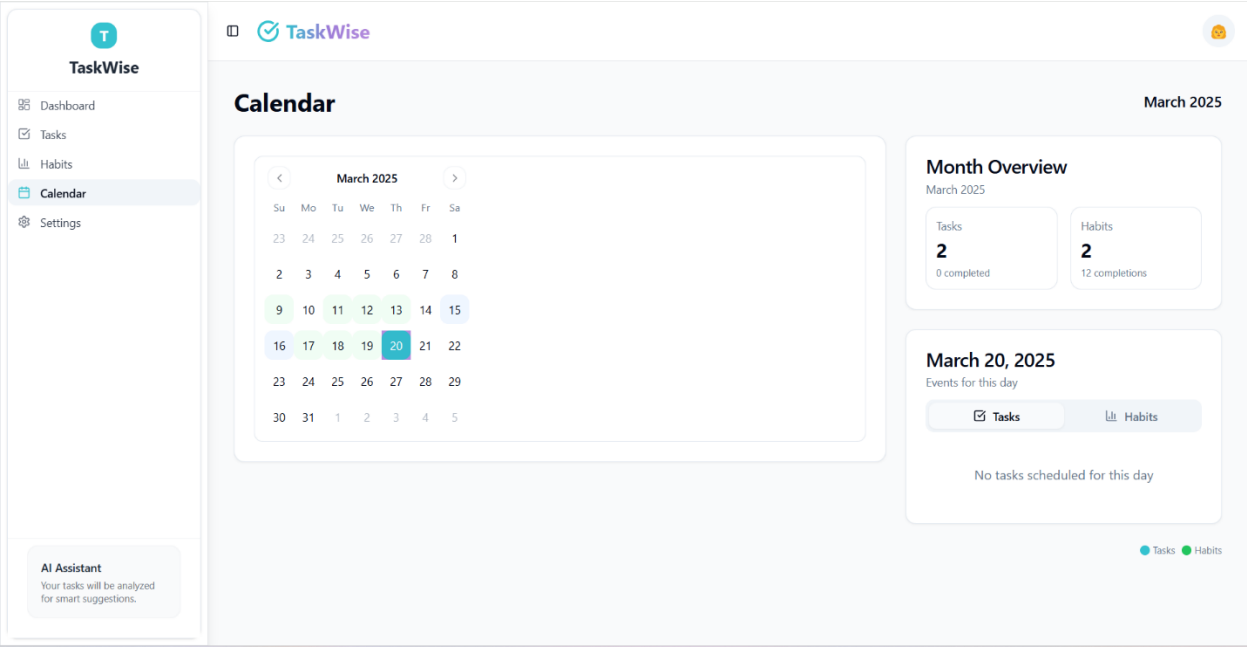

Figure 8: Calander

9. Responsive Layout

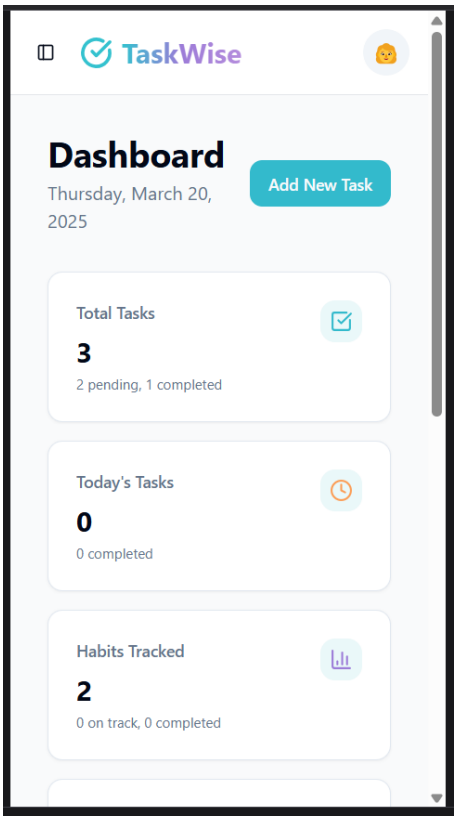

Figure 9: Responsive Layout

UML Diagrams

Data Flow Diagram

Components:

- User: Inputs tasks, updates, or retrieves them.
- AI Agent: Processes natural language inputs and prioritizes tasks.
- Task Management Module: Handles CRUD operations (Create, Read, Update, Delete).
- Database (PostgreSQL): Stores task data securely.
- OpenAI API: Provides AI-powered recommendations.

Data Flow Steps:

1. The user enters a task using natural language → AI Agent interprets the input.
2. The AI Agent interacts with the Task Management Module to create/update tasks.
3. Task Management Module stores tasks in the PostgreSQL database.
4. The user retrieves tasks → AI provides recommendations based on priorities.
5. User updates or deletes tasks → The system modifies the database accordingly.

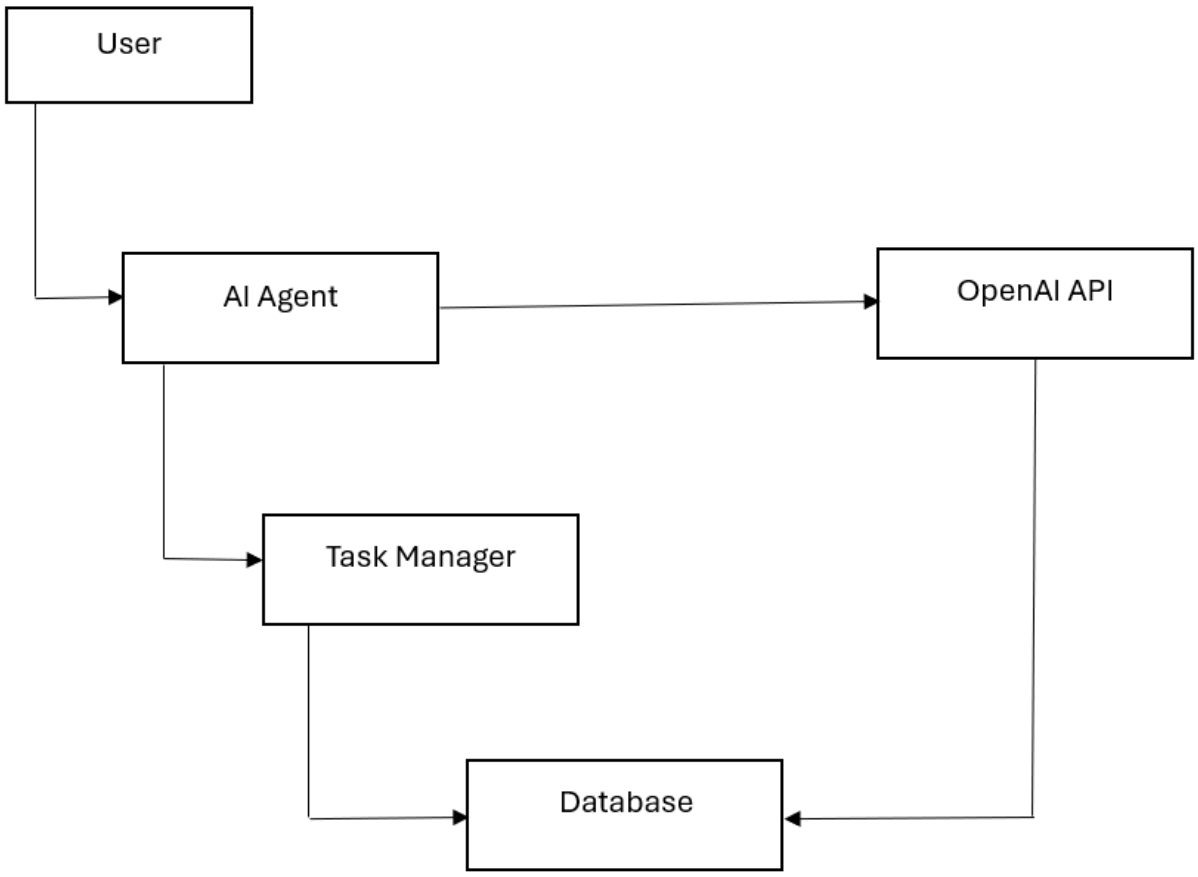

Figure 10: Data Flow Diagram

Component Diagram

Components:

- User Interface: Interacts with the user.
- Task Manager: Handles task creation, updates, and deletions.
- AI Agent: Interprets natural language and prioritizes tasks.
- Database (PostgreSQL): Stores task data.
- OpenAI API: Provides AI-powered recommendations.

Component Relationships:

1. The User Interface sends task requests to the Task Manager.
2. The Task Manager communicates with the AI Agent and the Database.
3. The AI Agent interacts with OpenAI API for task prioritization.
4. The Database stores and retrieves tasks as requested.

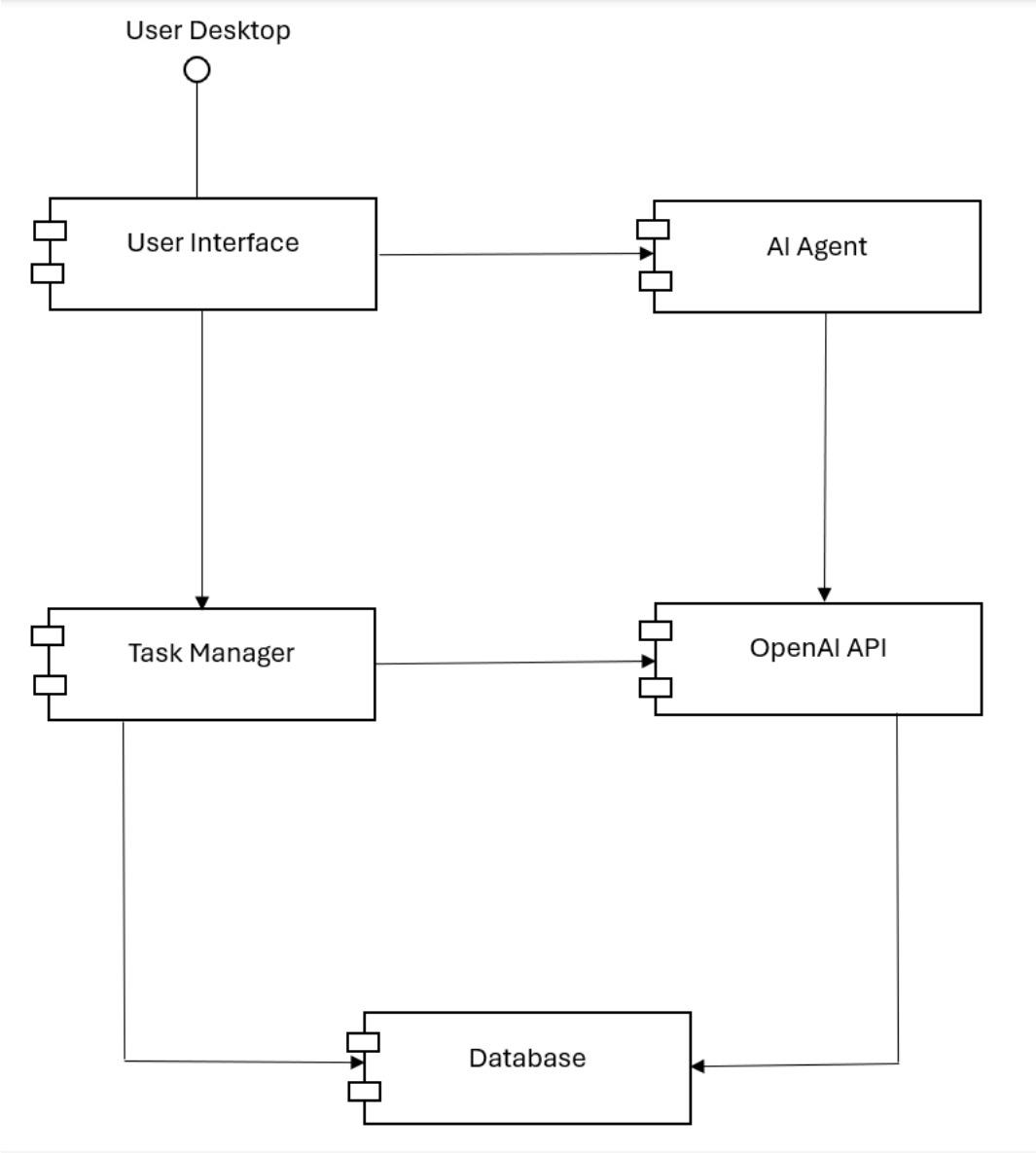

Figure 11: Component Diagram

CHAPTER 4: IMPLEMENTATION

The implementation of the project is broken down into 5 parts, they are as follows:

PART 1: Create a Docker File

Step 1: Create a folder “ai-todo-app” in your system.

Step 2: Create a JSON file

Run the following command to initialize a new Node.js project by creating a package.js file:

npm init

Once the package.js file is created, run the following command to install the TypeScripte type definition package for Node.js

npm i @types/node -D

Step 3: Create a Docker File

Create a **docker-compose.yaml** file to containerize and orchestrate your development environment.

PART 2: Create Schema

Step 1: Create a New Folder

Create a new folder **db** in the root directory. Inside the db folder create two folders:

10. Create a **schema.js** file that contains the database schema definition for the PostgreSQL database.
11. Create an **index.js** file to initialize the server, connect to the database, and load environment variables.

PART 3: Connect with PostgreSQL

Step 1: Integrating OpenAI (LLM)

To interact with the Postgres database, we will use Drizzle ORM. Run the following commands to integrate drizzle.

npm i drizzle-orm pg dotenv

npm i -D drizzle-kit tsx @types/pg

Step 2: Create .env File

Create an **.env** file in the root directory to store environment variables. Run the following commands:

npm I drizzle-kit -D

docker compose up -d

Step 3: Create a Query Page

Run the following command to create a new SQL query page:

npm run generate

Step 4: Create Drizzle Studio

Once the database is set up, run the following command to create a table in Drizzle Studio.

npm run studio

you will get a link for the studio
Drizzle Studio is up and running on https://local.drizzle.studio

If the port is already in use do the following:

netstat -ano | findstr :4983
taskkill /PID 23692 /F (replace the PID)

Then run
npm run studio

PART 4: Create CRUD Functions

Now let's create CRUD functions.
Create an index.js file in the root directory

Once you're done writing all the functions, let's integrate Open AI. Run the following commands.

npm i openai

npm I readline-sync

Finally, execute:
node index.js

PART 5: Create the UI

Create a web-based UI using Typescript, React, and CSS for simple and intuitive user interaction.

Import the following dependencies:

CHAPTER 5: FUTURE WORK

Conclusion

The development of TaskWise, an AI-powered To-Do agent, has successfully demonstrated how artificial intelligence can enhance productivity through intelligent task management. By integrating AI-driven task prioritization, CRUD operations, and seamless database interactions using PostgreSQL and Drizzle ORM, the system ensures efficient and user-friendly task handling. The project followed a structured development lifecycle, including requirement analysis, system design, implementation, and testing, resulting in a functional prototype that aligns with the initial objectives.

TaskWise effectively automates task entry, improves task prioritization, and enhances user experience through AI-driven recommendations. The system's scalability and modular design ensure that it can be extended with additional features in future iterations. Overall, this project provides a strong foundation for intelligent task management and showcases the potential of AI in streamlining everyday productivity.

Future Work

While TaskWise offers essential task management capabilities, several enhancements can be implemented to further improve its functionality and user experience:

1. **User Authentication & Multi-Device Sync** – Implementing user authentication will allow multi-device synchronization, ensuring users can access their tasks from different devices securely.
2. **Voice-Based Task Input** – Integrating speech recognition will enable users to add tasks using voice commands, improving accessibility and convenience.
3. **Advanced AI Recommendations** – Enhancing AI capabilities to offer smarter task suggestions based on user behavior and historical data.
4. **Mobile Application Development** – Creating a mobile version of TaskWise for Android and iOS to provide users with on-the-go task management.
5. **Integration with Calendar & Productivity Tools** – Connecting TaskWise with Google Calendar, Microsoft Outlook, and other productivity tools will allow seamless workflow integration.
6. **Performance Optimization & Scalability** – Optimizing database queries and AI processing for better speed and efficiency as the user base grows.

These future enhancements will make TaskWise a more comprehensive and user-friendly AI-powered task management system, further pushing the boundaries of intelligent automation in daily productivity.

CHAPTER 6: REFERENCES AND APPENDIX

- AI and Natural Language Processing (NLP)
  OpenAI Building Agents, OpenAI Platform
  Reference link: https://platform.openai.com/docs/guides/agents
  What are AI Agents? Anna Gutowska, IBM
  Reference link: https://www.ibm.com/think/topics/ai-agents

- Database Management
  Drizzle ORM PostgreSQL, Drizzle
  Reference link: https://orm.drizzle.team/docs/get-started-postgresql

- Drizzle Studio
  Meet Drizzle Studio, Drizzle
  Reference link: https://orm.drizzle.team/drizzle-studio/overview

- Web and UI Development
  Prompt Engineering, Lovable
  Reference link: https://docs.lovable.dev/tips-tricks/prompting

- Development Lifecycle
  GitHub Integration, Lovable
  Reference link: https://docs.lovable.dev/integrations/git-integration