# Project Documentation: TaskVerse

## 1. Introduction

• Project Title: TaskVerse: A Collaborative Task Management Platform

• Team Members: [List team members and their roles]

## 2. Project Overview

• Purpose: TaskVerse is a platform designed to streamline task management and enhance collaboration among team members. The goal is to provide users with an intuitive interface and robust backend for tracking tasks, assigning responsibilities, and monitoring progress.

• Features:

- Task creation, assignment, and prioritization

- Real-time updates and notifications

- User authentication and role-based access control

- Dashboard with analytics and progress tracking

## 3. Architecture

• Frontend: The frontend is built with React.js, utilizing components, hooks, and state management for a dynamic user interface.

• Backend: The backend is powered by Node.js with Express.js, offering a robust REST API for data handling and business logic.

• Database: MongoDB serves as the database, storing user, task, and project data in a structured schema for efficient queries.

## 4. Setup Instructions

• Prerequisites: Ensure the following are installed:

- Node.js (version 14 or higher)

- MongoDB (local or cloud instance)

• Installation:

1. Clone the repository: `git clone [repository URL]`

2. Navigate to the project folder: `cd TaskVerse`

3. Install dependencies for both client and server: `npm install` in respective directories

4. Set up environment variables: Create a `.env` file with necessary keys (e.g., MongoDB URI, JWT secret).

## 5. Folder Structure

• Client: Contains React frontend code organized into components, pages, and services.

• Server: Houses Node.js backend with routes, models, controllers, and middleware.

## 6. Running the Application

• Frontend: Run `npm start` in the client directory.

• Backend: Run `npm start` in the server directory.

## 7. API Documentation

• Endpoints:

- POST /api/auth/login: User login endpoint

- POST /api/auth/register: User registration endpoint

- GET /api/tasks: Retrieve tasks for the authenticated user

- POST /api/tasks: Create a new task

## 8. Authentication

• Authentication and authorization are managed using JSON Web Tokens (JWT). Tokens are issued upon login and validated for each protected route.

## 9. User Interface

• Screenshots or GIFs showcasing various UI components such as dashboards, task creation forms, and analytics can be included here.

## 10. Testing

• Testing Strategy: Utilized Jest and React Testing Library for frontend tests, and Mocha with Chai for backend API testing.

## 11. Screenshots or Demo

• Screenshots or a link to the deployed demo: [Insert screenshots or link here]

## 12. Known Issues

• [List any known bugs or areas that need improvement]

## 13. Future Enhancements

• Potential future improvements include:

- Adding support for project templates

- Enhancing analytics with machine learning insights

- Implementing a mobile-friendly design