API Design and Implementation with FASTAPI and Swagger

Solo Project by Kamogelo Ellen Kganakga

KamoEllen/project-management-api-v2 (github.com)

Overview

The Project Management API is a FastAPI-based application designed to facilitate the management of projects, milestones, tasks, and progress tracking. This API allows users to create, read, update, and delete data related to projects, milestones, tasks, and user progress. It also includes user authentication to ensure secure access to endpoints.

User Authentication: Secure login and token-based authentication.
Project Management: Endpoints to manage projects, milestones, tasks, and track progress.
CRUD Operations: Full support for Create, Read, Update, and Delete operations.

Current Features Checklist

- Authentication and Authorization
- Token-based authentication with JWT
- User registration
- Secure password hashing with bcrypt
- Token expiration management
- User Management
- Create a new user
- Retrieve user details by ID
- Update user details
- Delete a user
- Milestone Management
- Create a new milestone
- Retrieve milestones with pagination
- Retrieve a single milestone by ID
- Update a milestone by ID
- Delete a milestone by ID
- Task Management
- Create a new task
- Retrieve tasks with pagination
- Retrieve a single task by ID
- Update a task by ID
- Delete a task by ID
- Progress Tracking
- Create a new progress entry
- Retrieve progress entries by user ID
- Retrieve a single progress entry by ID
- Error Handling
- Standard HTTP status codes for errors
- Custom error messages for clarity

Architecture and Technologies

Technologies

FastAPI: I chose it for its speed and built-in Swagger UI, which is great for interactive design and customization. It's now my go-to framework, as it also handles the front-end aspects I need.

Pydantic: Used for data validation because it integrates well with FastAPI and simplifies handling data models.

Motor: Selected for asynchronous MongoDB operations. Initially, I faced dependency issues, which were resolved by upgrading to Python 3.11.

Passlib: Chosen for secure password hashing and management.

Client-Server Model: Client requests data via HTTP methods (GET, POST, PUT, PATCH, DELETE).

Development Report

Challenges

Securely managing user authentication and authorization.

Ensuring accurate and secure data handling.

Properly identifying and managing errors.

Solution

Used JWTs for secure token-based authentication.
Implemented password hashing with `bcrypt` for secure storage.

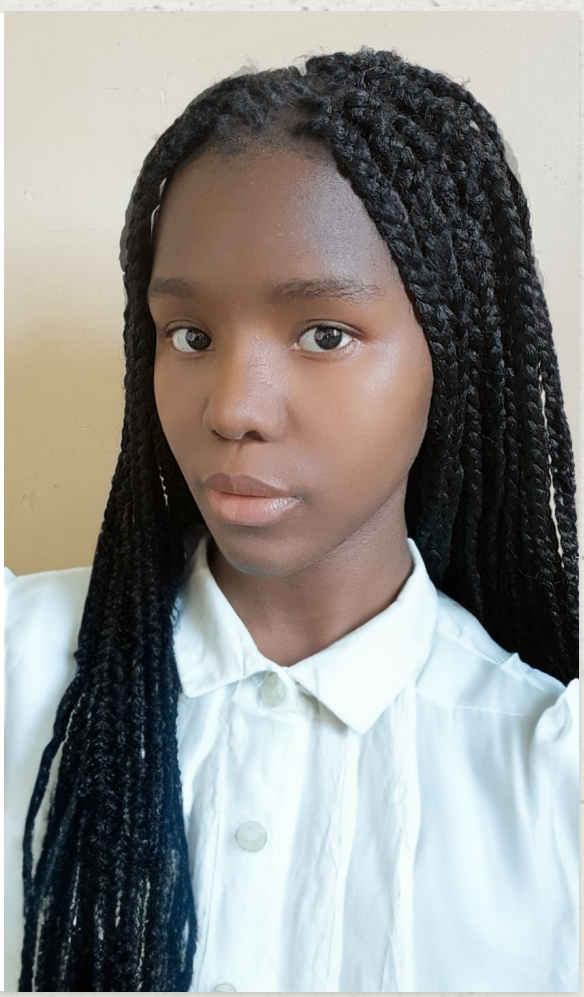
Used Pydantic models for validation and serialization.
Added custom error handling for clear responses.

Used HTTP status codes for error reporting.

Summary of Experience

Developed a functional and well-documented RESTful API.
Gained experience in API design, implementation, and documentation.
Identified key areas for future enhancement and growth.

Access my API :
Project Management API - Swagger UI
(project-management-api-v2.onrender.com)



Demo Recording - Project-Management-API Demo (youtube.com)

Project Management API 1.0.0 OAS 3.1

/openapi.json

API designed for managing projects, milestones, tasks, and tracking progress.

Authorize

Acess Token

POST /token Login For Access Token

Register Team Member

POST /users/ Register User

Projects

GET /projects/ Get Projects Endpoint

POST /projects/ Create Project Endpoint

GET /projects/{project_id} Get Project Endpoint

PUT /projects/{project_id} Update Project Endpoint

THANKS!

Do you have any questions?

kamoellenkganakga@gmail.com