# Technical Documentation for

# Task Management Software

## Table of Contents

### [Introduction](" \l "introduction)

### [Architecture](" \l "architecture)

### [Technologies Used](" \l "technologies-used)

### [Installation](" \l "installation)

### [Configuration](" \l "configuration)

### [API Endpoints](" \l "api-endpoints)

### [User Registration](" \l "user-registration)

### [User Login](" \l "user-login)

### [Task Management](" \l "task-management)

### [Database Schema](" \l "database-schema)

### [Testing](" \l "testing)

### [Contributing](" \l "contributing)

### 1. Introduction

The Task Management Software is a web application designed to help users manage tasks efficiently. Users can create, update, delete, and retrieve tasks, along with user registration and authentication functionalities. The application utilizes JWT (JSON Web Tokens) for secure user sessions.

### 2. Architecture

The software is built using the FastAPI framework, which provides high performance and easy integration with Python. The application follows a microservices architecture with separate components for user management and task handling, allowing for scalability and maintainability.

### 3. Technologies Used

* **Backend**: FastAPI
* **Database**: MongoDB
* **Authentication**: JWT (JSON Web Tokens)
* **ORM**: PyMongo for MongoDB interaction
* **Hashing**: Bcrypt for password hashing
* **Testing**: Pytest for unit testing
* **Environment Management**: Dotenv for managing environment variables

### 4. Installation

To install the application, follow these steps:

Clone the repository:

bash

Copy code

git clone <repository-url>cd task\_management

Create a virtual environment:

bash

Copy code

python -m venv venvsource venv/bin/activate # On Windows use `venv\Scripts\activate`

Install the required packages:

bash

Copy code

pip install fastapi[all] pymongo bcrypt python-dotenv

### 5. Configuration

Create a .env file in the root directory of the project to store sensitive information such as the secret key for JWT. Example content for the .env file:

makefile

Copy code

SECRET\_KEY=your\_secret\_key

### 6. API Endpoints

#### User Registration

* **Endpoint**: POST /register
* **Description**: Registers a new user.
* **Request Body**:

json

Copy code

{

"username": "new\_user",

"email": "new\_user@example.com",

"password": "user\_password",

"is\_admin": false}

* **Response**:
  + 200: User successfully registered.
  + 400: Bad request if validation fails.

#### User Login

* **Endpoint**: POST /login
* **Description**: Authenticates a user and returns a JWT.
* **Request Body**:

json

Copy code

{

"username": "user",

"password": "user\_password"}

* **Response**:
  + 200: Returns the access token.
  + 400: Invalid credentials.

#### Task Management

**Create Task**:

* + **Endpoint**: POST /tasks
  + **Request Body**:

json

Copy code

{

"title": "Task Title",

"details": "Task Details",

"due\_date": "2024-12-31T12:00:00"}

* + **Response**: Returns the task ID.

**Get Task**:

* + **Endpoint**: GET /tasks/{task\_id}
  + **Response**: Returns task details.

**Update Task**:

* + **Endpoint**: PUT /tasks/{task\_id}
  + **Request Body**: Same as Create Task.
  + **Response**: Confirmation of the update.

**Delete Task**:

* + **Endpoint**: DELETE /tasks/{task\_id}
  + **Response**: Confirmation of the deletion.

### 7. Database Schema

#### Users Collection

json

Copy code

{

"\_id": "ObjectId",

"username": "string",

"email": "string",

"password": "string",

"is\_admin": "boolean"}

#### Tasks Collection

json

Copy code

{

"\_id": "ObjectId",

"title": "string",

"details": "string",

"due\_date": "datetime",

"created\_by": "string"}

### 8. Testing

To run the test suite, navigate to the directory containing the test\_tasks.py file and execute:

bash

Copy code

pytest test\_tasks.py

### 9. Contributing

Contributions are welcome! Please open an issue or submit a pull request for any enhancements or bug fixes.