**Task Management Web Application**

Table of Contents

Introduction

Features

Tech Stack

API Documentation

Authentication

Setup and Installation

Frontend

Backend

Database

Deployment

Contributing

**Introduction**

This is a simple Task Management Web Application that allows users to manage their tasks through basic CRUD operations. The app supports adding new tasks, updating existing tasks, viewing task details, and deleting tasks. Additionally, it includes user authentication, so only logged-in users can create, update, or delete tasks.

**Features**

* Add, Edit, Delete, and View tasks.
* User authentication (register, login, logout).
* Persistent data storage.
* Responsive frontend design using [your frontend framework of choice].
* RESTful API for task management.

**Tech Stack**

* **Backend**: Python (Django)
* **Frontend**: React/Vue/Angular
* **Database**: MySQL

**API Documentation**

**Endpoints**

1. **GET /tasks**

Fetch all tasks.

[

{

"id": 1,

"title": "Sample Task",

"description": "This is a sample task",

"status": "Pending",

"due\_date": "2024-10-31"

}

]

**GET /tasks/**

Fetch a task by its ID.

Response:

{

"id": 1,

"title": "Sample Task",

"description": "This is a sample task",

"status": "Pending",

"due\_date": "2024-10-31"

}

**POST /tasks**

Add a new task.

{

"title": "New Task",

"description": "Task description",

"status": "Pending",

"due\_date": "2024-11-01"

}

**PUT /tasks/**

Update a task by ID.

Request body

{

"title": "Updated Task",

"description": "Updated description",

"status": "Completed",

"due\_date": "2024-10-31"

}

Response

{

"message": "Task updated successfully",

"task": {

"id": 1,

"title": "Updated Task",

"description": "Updated description",

"status": "Completed",

"due\_date": "2024-10-31"

}

}

**DELETE /tasks/**

Delete a task by ID.

{

"message": "Task deleted successfully"

}

**Error Handling**

* **400 Bad Request**: Returned when invalid data is sent in a request (e.g., missing fields, invalid formats).
* **404 Not Found**: Returned when a task with the specified ID is not found.
* **500 Internal Server Error**: General server-side errors.

**Frontend**

The frontend is built using [React/Vue/Angular] with the following views:

* **List View**: Displays all tasks and includes a delete button.
* **Details View**: Shows the details of a single task.
* **Add/Edit View**: A form to create or update tasks.

**Responsive Design**

The frontend uses Bootstrap to ensure responsiveness across different screen sizes.

**Backend**

The backend provides a RESTful API to handle task management, built using Django with the following features:

* Persistent data storage using MySQL
* Error handling for incorrect routes and invalid input.

**Database**

The application uses MySQL to store task and user information. The database schema includes:

1. **User Table**: Stores user credentials (username, email, password hash).
2. **Task Table**: Stores task information (title, description, status, due date, user ID).