SmartTask:

End-to-End Task Management Application

Documentation

Project Title: SmartTask: End-to-End Task Management Application

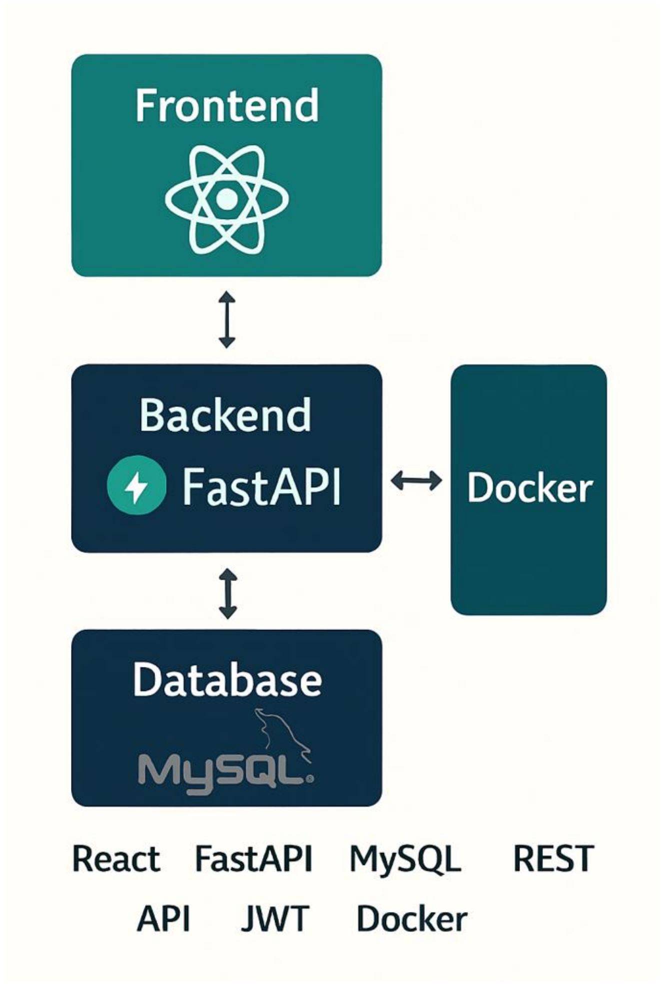
GitHub Repository:

https://github.com/Chakradhar080/SmartTask-End-to-End-Task-Management-Application.git

1. Project Objective

The Task Manager Full-Stack Application is a web-based system for managing tasks. It allows users to: - Register and log in - Create, read, update, and delete tasks - Track task status (pending, in-progress, completed)

It demonstrates: - Full-stack development using React (frontend) + FastAPI (backend) + MySQL (database) - REST API development - JWT authentication - Docker deployment for backend and database - API documentation and Postman collection



Full-stack architecture diagram

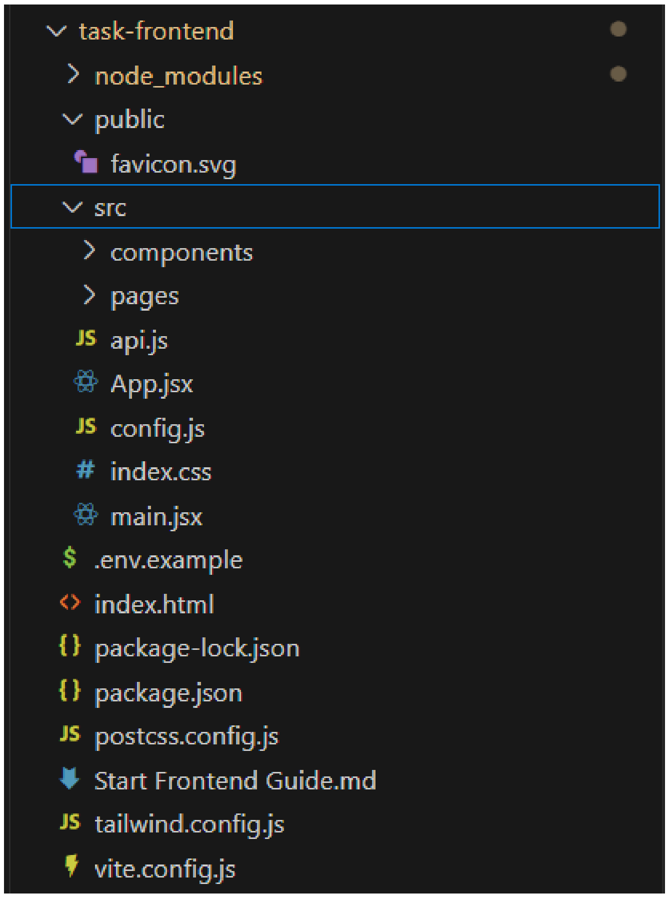
2. Technology Stack

Layer	Technology / Tools
Frontend	React, JavaScript, Axios, Tailwind CSS
Backend	Python, FastAPI, SQLAlchemy, Pydantic, JWT
Database	MySQL 8.0
Deployment	Docker, Docker Compose
Documentation	Swagger/OpenAPI, Postman

---

3. Frontend Details

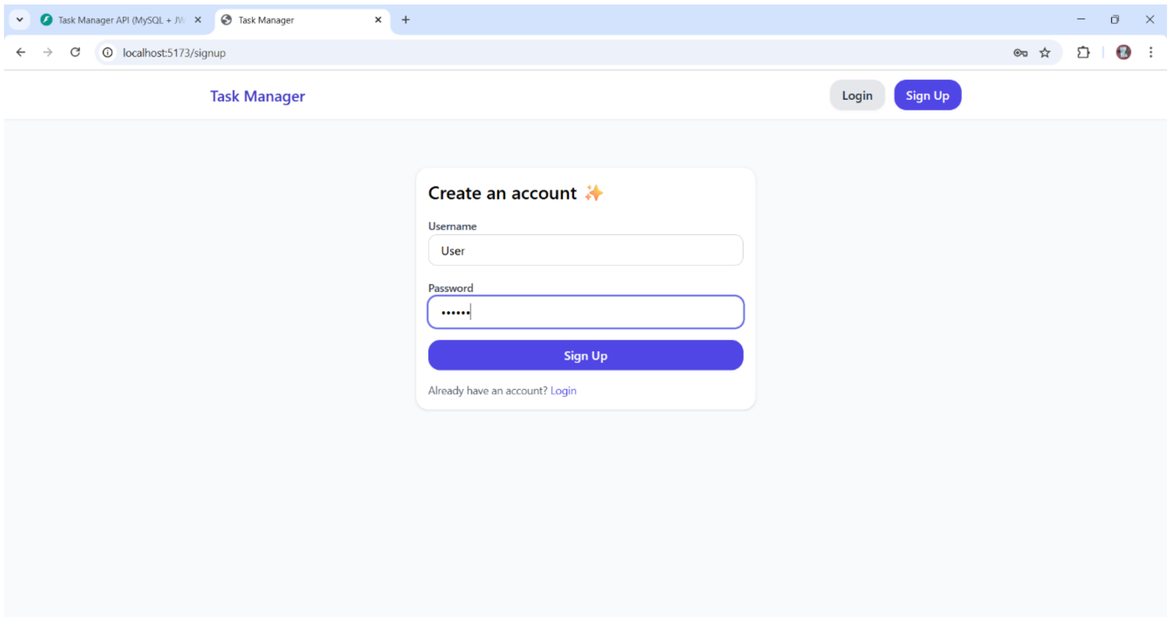
3.1 Folder Structure



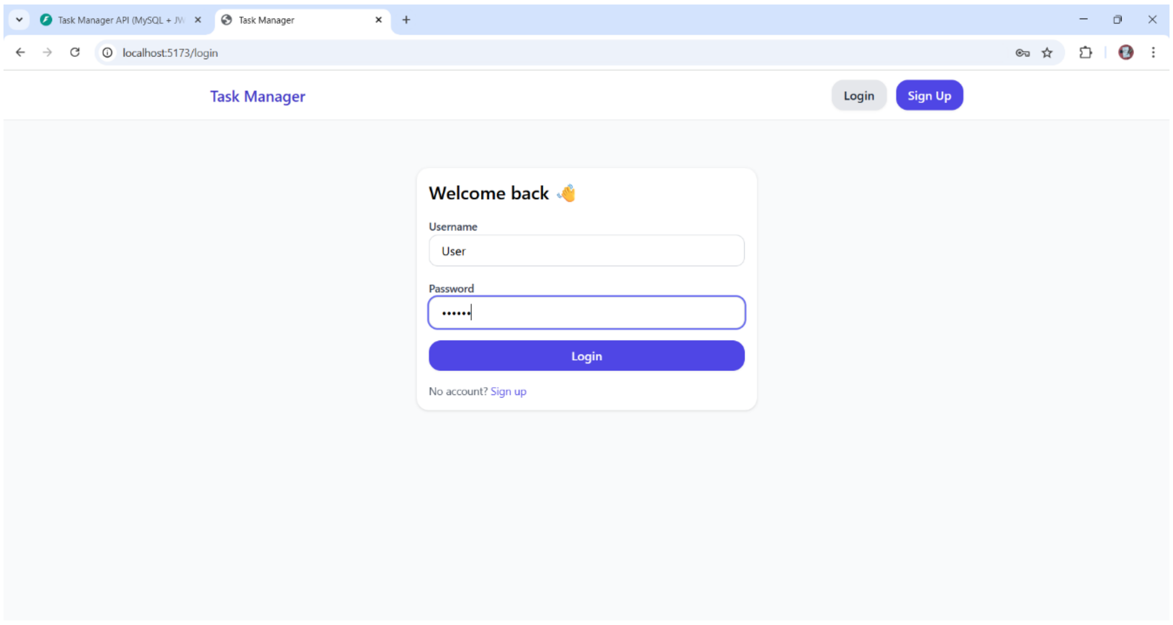
```
task-frontend
  node_modules
  public
    favicon.svg
  src
    components
    pages
    api.js
    App.jsx
    config.js
    index.css
    main.jsx
  .env.example
  index.html
  package-lock.json
  package.json
  postcss.config.js
  Start Frontend Guide.md
  tailwind.config.js
  vite.config.js
```

3.2 Key Features

1. **User Registration & Login**
   - Signup page posts to `/signup`
   - Login page posts to `/token` to get JWT
   - JWT saved in local storage for authenticated API calls
2. **Task Management**
   - Task list fetches tasks from `/tasks`
   - Task creation, update, deletion via `/tasks` endpoints
   - Real-time updates using Axios requests



Scrrenshot 3.2.1: Account Creation



Welcome back 👋

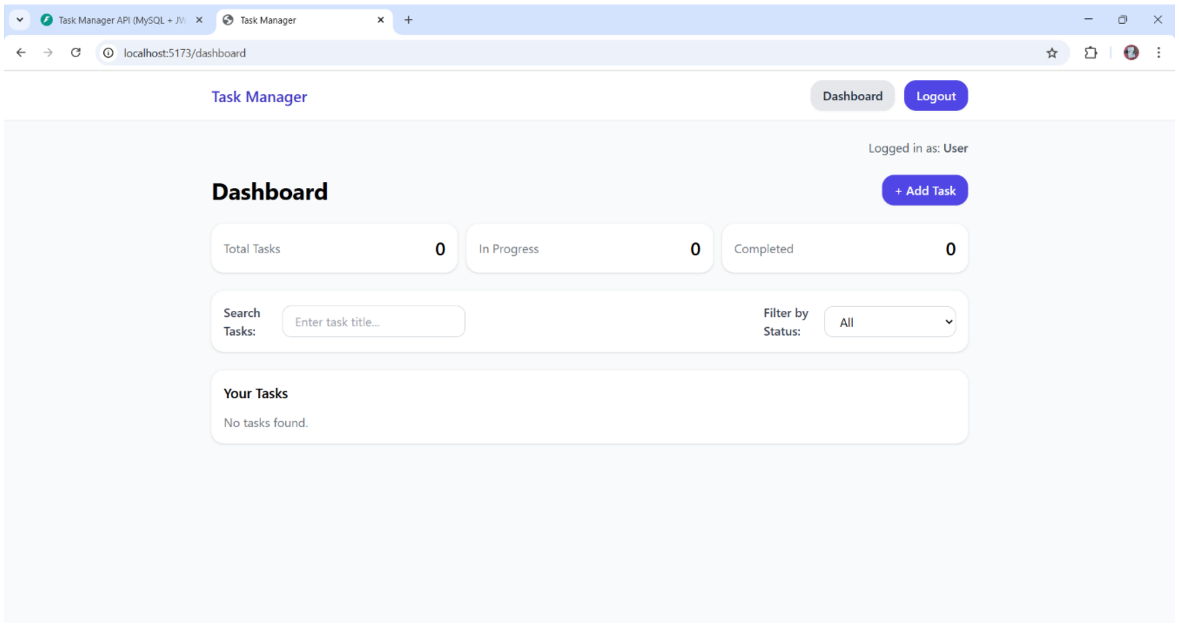
Username
User

Password
••••••

Login

No account? Sign up

Screenshot 3.2.2: Login to Created Account



Logged in as: User

Dashboard

+ Add Task

Total Tasks	0
In Progress	0
Completed	0

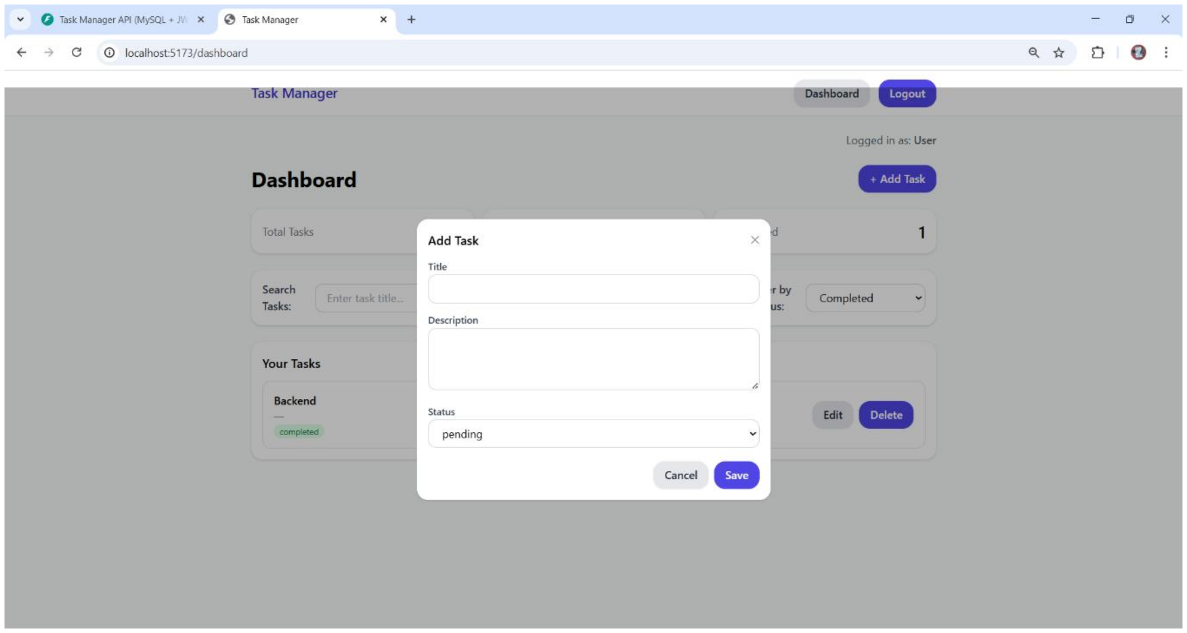
Search Tasks: Enter task title...

Filter by Status: All

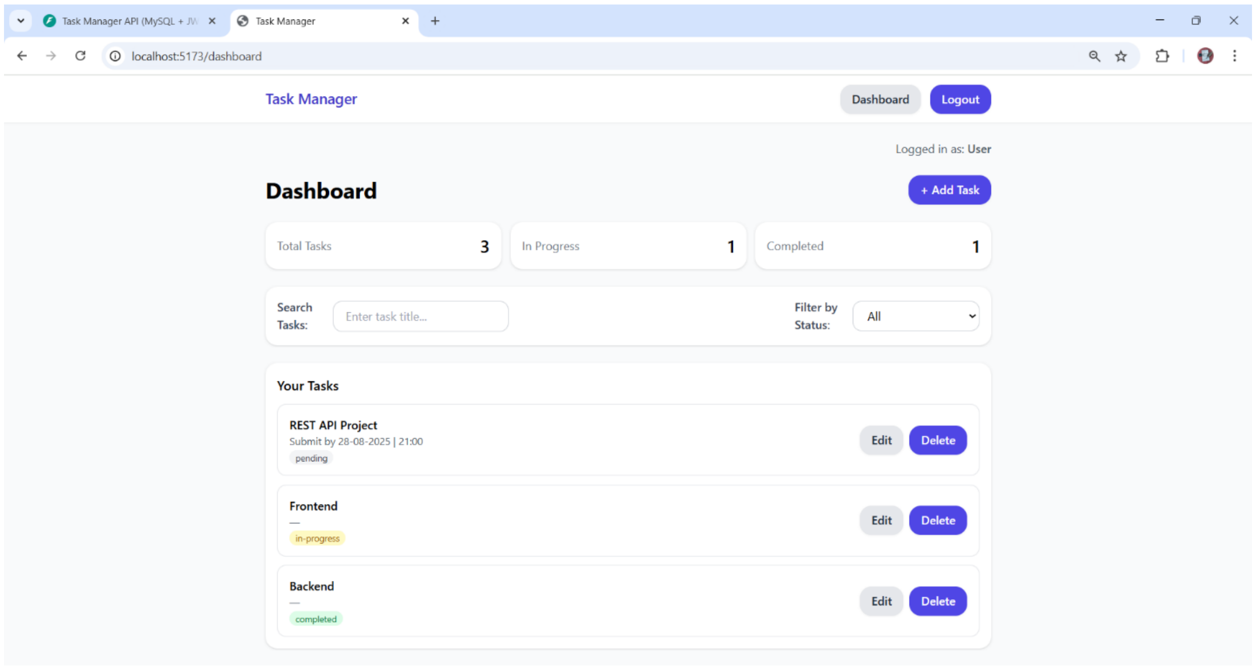
Your Tasks

No tasks found.

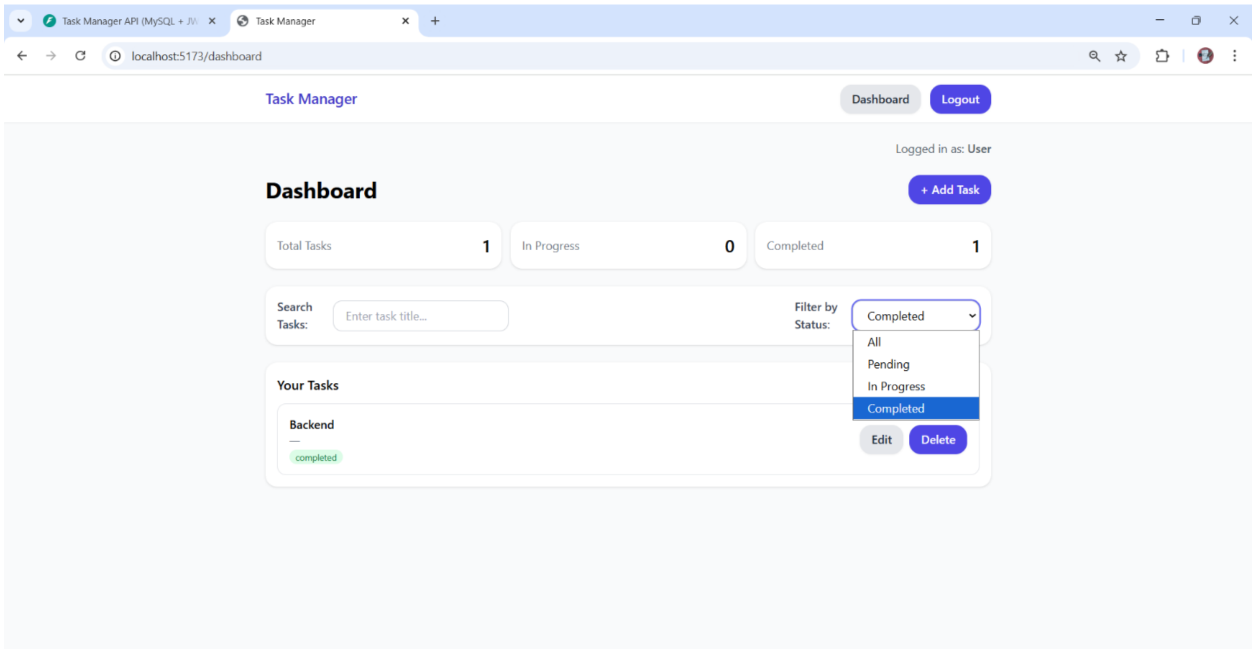
Screenshot 3.2.3: User Dashboard



Screenshot 3.2.4: Task Creation



Screenshot 3.2.5: Tasks Created


Task Manager API (MySQL + JW
Task Manager
localhost:5173/dashboard

Task Manager

Dashboard Logout

Logged in as: User

Dashboard

+ Add Task

Total Tasks 1
In Progress 0
Completed 1

Search Tasks: Enter task title...

Filter by Status: Completed

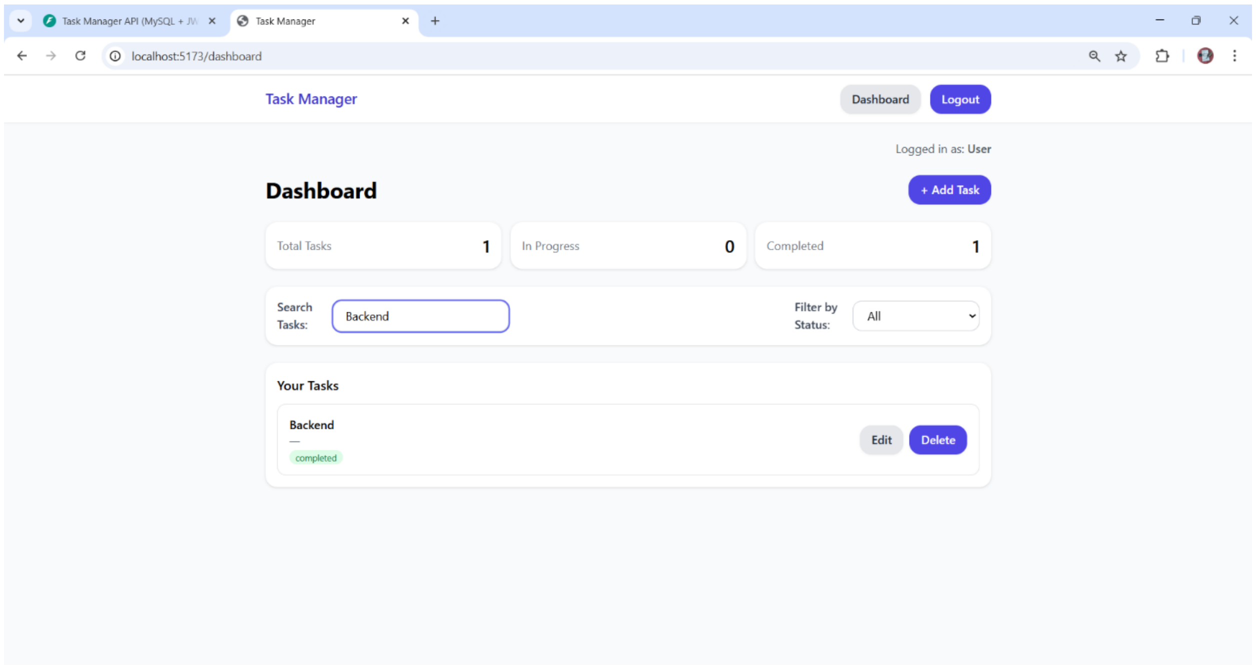
All
Pending
In Progress
Completed

Your Tasks

Backend
—
completed

Edit Delete

Screenshot 3.2.6: Filer Tasks by Status


Task Manager API (MySQL + JW
Task Manager
localhost:5173/dashboard

Task Manager

Dashboard Logout

Logged in as: User

Dashboard

+ Add Task

Total Tasks 1
In Progress 0
Completed 1

Search Tasks: Backend

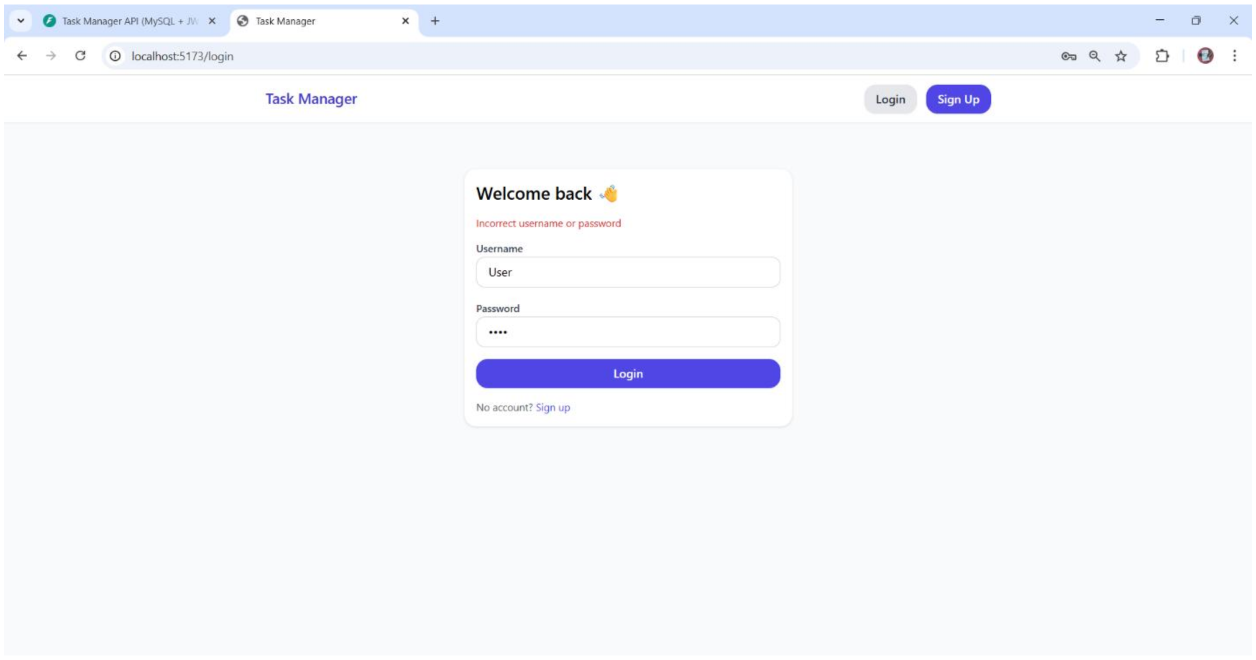
Filter by Status: All

Your Tasks

Backend
—
completed

Edit Delete

Screenshot 3.2.7: Filter Tasks by Searching



Welcome back 👋

Incorrect username or password

Username: User

Password: ••••

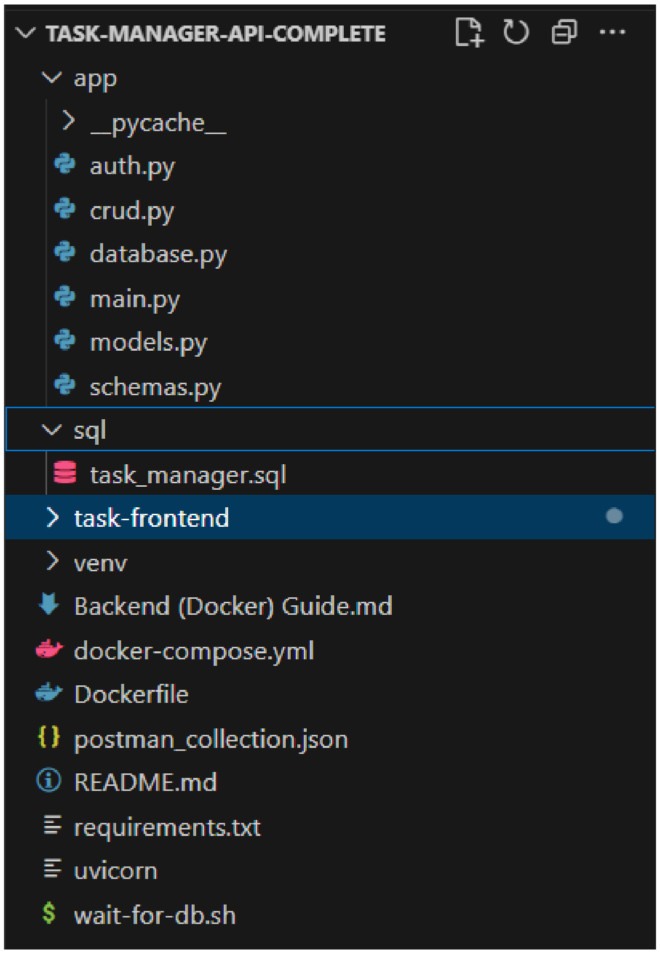
Screenshot 3.2.8: Error Message if the Credentials are Mismatch or wrong

3.3 API Integration (Frontend → Backend)

```
import axios from 'axios';

const API_URL = "http://localhost:8000";

export const signup = (data) => axios.post(`${API_URL}/signup`, data);
export const login = (data) => axios.post(`${API_URL}/token`, data);

export const getTasks = (token) => axios.get(`${API_URL}/tasks`, {
    headers: { Authorization: `Bearer ${token}` }
});

export const createTask = (task, token) => axios.post(`${API_URL}/tasks`,
task, {
    headers: { Authorization: `Bearer ${token}` }
});
```

4. Backend Details

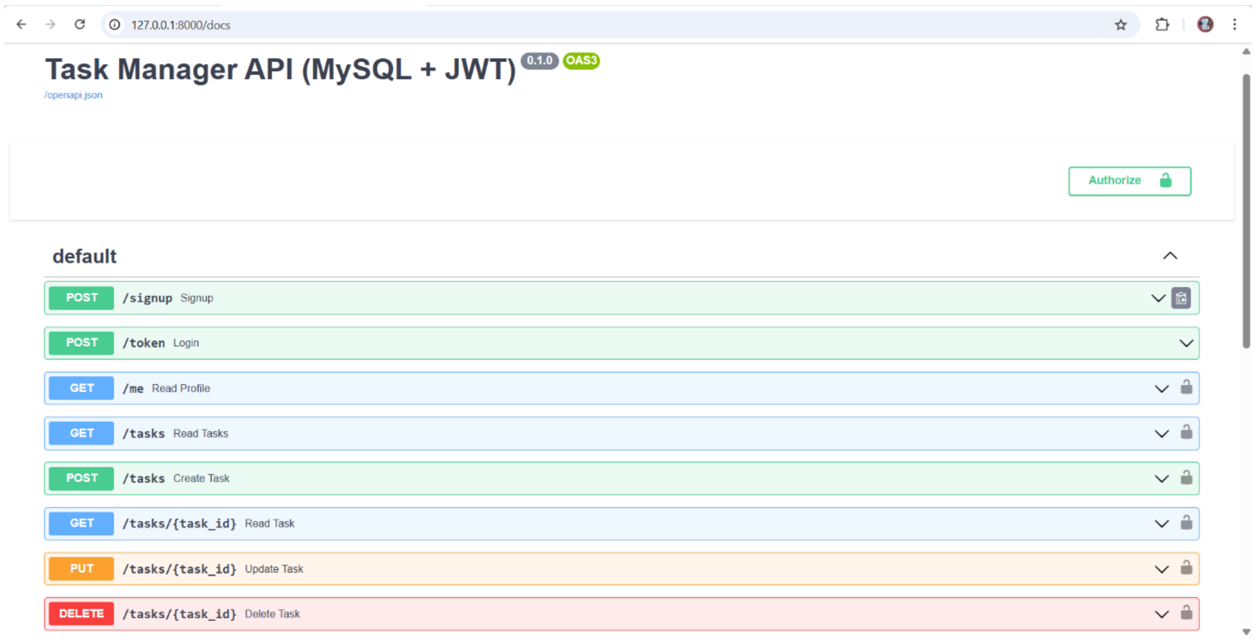
4.1 Folder Structure



```
TASK-MANAGER-API-COMPLETE
  app
    __pycache__
    auth.py
    crud.py
    database.py
    main.py
    models.py
    schemas.py
  sql
    task_manager.sql
  task-frontend
  venv
  Backend (Docker) Guide.md
  docker-compose.yml
  Dockerfile
  postman_collection.json
  README.md
  requirements.txt
  uvicorn
  wait-for-db.sh
```

4.2 API Endpoints

Method	Endpoint	Description	Auth Required
POST	`/signup`	Create new user	No
POST	`/token`	Login and receive JWT	No
GET	`/me`	Get current user profile	Yes
GET	`/tasks`	Get all tasks	Yes
POST	`/tasks`	Create new task	Yes
GET	`/tasks/{id}`	Get task by ID	Yes
PUT	`/tasks/{id}`	Update task by ID	Yes
DELETE	`/tasks/{id}`	Delete task by ID	Yes


127.0.0.1:8000/docs

Task Manager API (MySQL + JWT) 0.1.0 OAS3

/openapi.json

Authorize

default

POST /signup Signup

POST /token Login

GET /me Read Profile

GET /tasks Read Tasks

POST /tasks Create Task

GET /tasks/{task_id} Read Task

PUT /tasks/{task_id} Update Task

DELETE /tasks/{task_id} Delete Task

Screenshot 4.2.1: Swagger UI screenshot with endpoints

4.3 Example Postman Requests

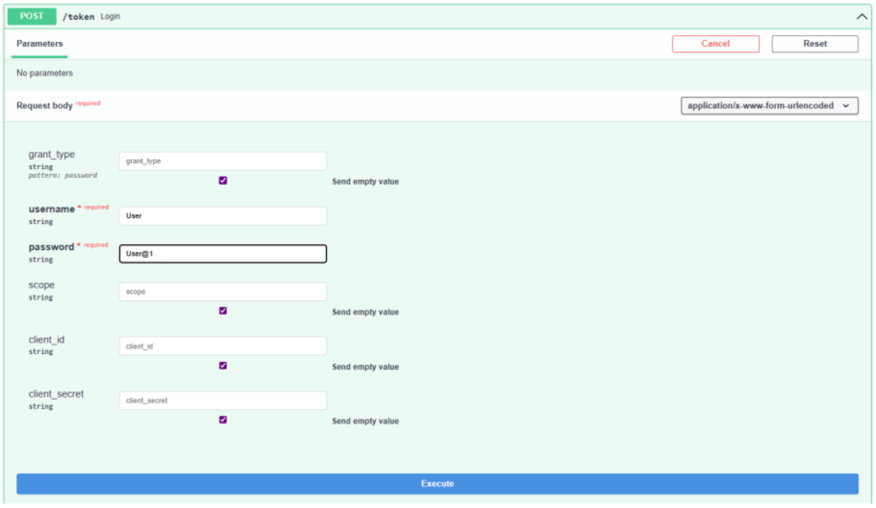
Login:

- POST /token
- Body (x-www-form-urlencoded):

```
username=john&password=123456&grant_type=password
```

- Response:

```
{
  "access_token": "<JWT_TOKEN>",
  "token_type": "bearer"
}
```


POST /token Login

Parameters — Cancel — Reset

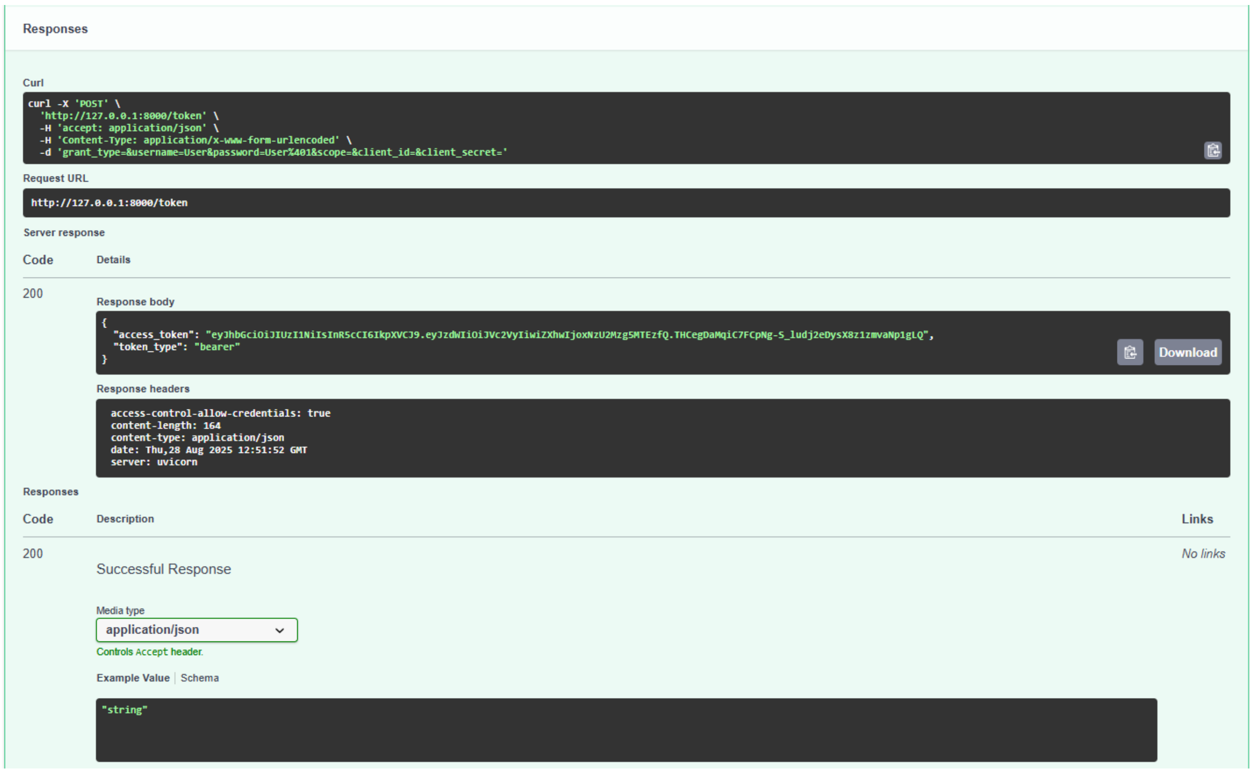
No parameters

Request body required — application/x-www-form-urlencoded

grant_type string pattern: password	grant_type	✓ Send empty value
username * required string	User	
password * required string	User@1	
scope string	scope	✓ Send empty value
client_id string	client_id	✓ Send empty value
client_secret string	client_secret	✓ Send empty value

Execute

Screenshot 4.3.1: Postman login request


Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/token' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'grant_type=&username=User&password=User%401&scope=&client_id=&client_secret='
```

Request URL

```
http://127.0.0.1:8000/token
```

Server response

Code | Details

200

Response body

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJVc2VyIiwiZXhwIjoxNzU2Mzg5MTEzfQ.THCegDaMqiC7FCpNg-S_ludj2eDysX8z1zmvaNp1gLQ",
  "token_type": "bearer"
}
```

Response headers

```
 access-control-allow-credentials: true
 content-length: 164
 content-type: application/json
 date: Thu,28 Aug 2025 12:51:52 GMT
 server: uvicorn
```

Responses

Code | Description | Links

200 | Successful Response | *No links*

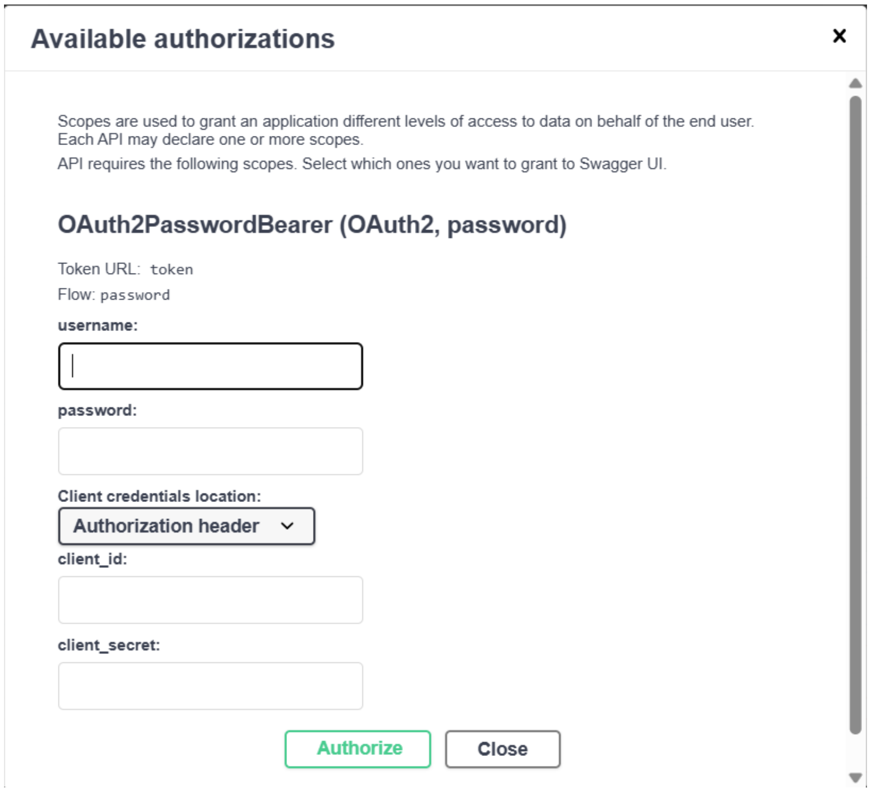
Media type: application/json

Controls `Accept` header.

Example Value | Schema

```
"string"
```

Screenshot 4.3.2: Response for Login request


Available authorizations

Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.

API requires the following scopes. Select which ones you want to grant to Swagger UI.

OAuth2PasswordBearer (OAuth2, password)

Token URL: `token`

Flow: `password`

username:

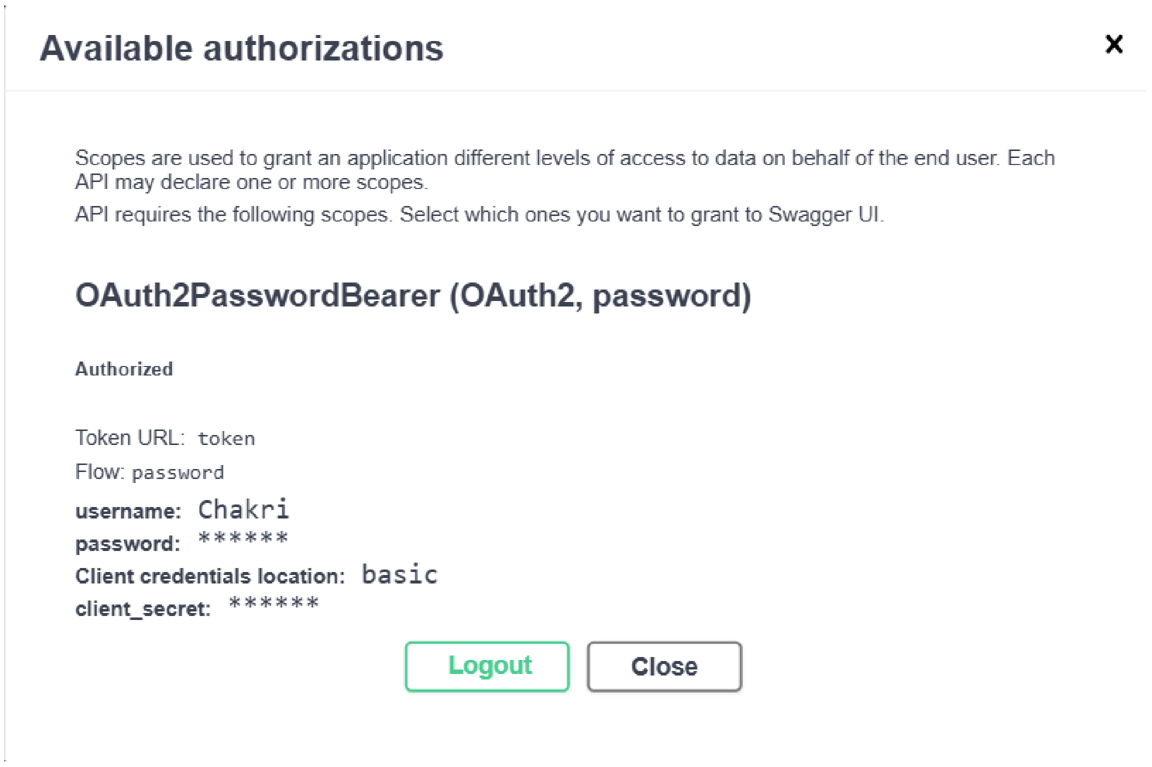
password:

Client credentials location: Authorization header

client_id:

client_secret:

Screenshot 4.3.3: Login Request (To Access all operations in Backend)



Available authorizations

Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.
API requires the following scopes. Select which ones you want to grant to Swagger UI.

OAuth2PasswordBearer (OAuth2, password)

Authorized

Token URL: `token`
Flow: `password`
username: Chakri
password: ******
Client credentials location: basic
client_secret: ******

Screenshot 4.3.4: Response if it is authorized

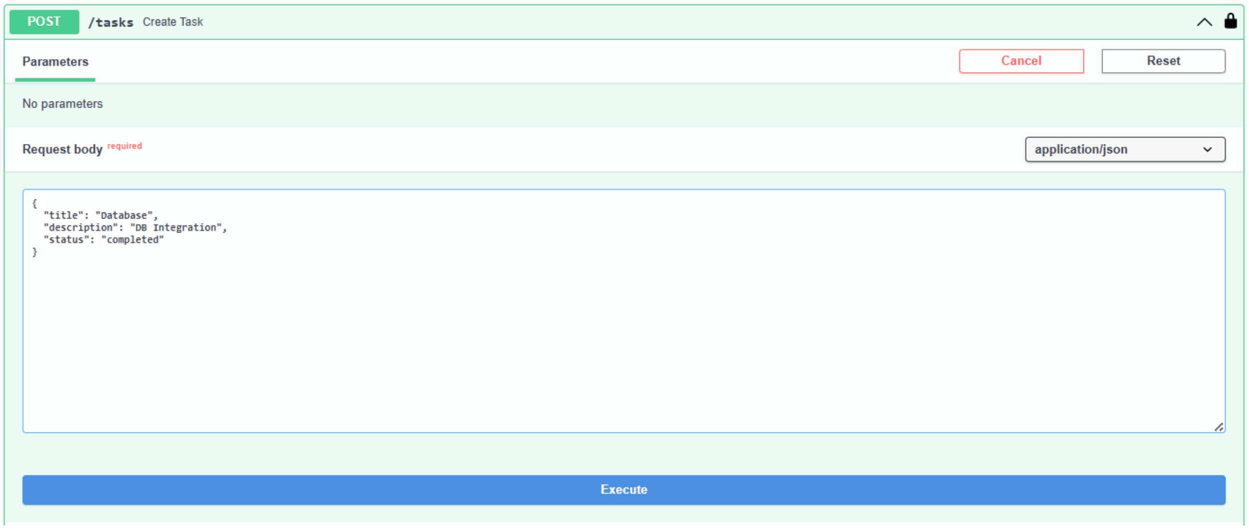
Create Task:

- POST `/tasks`
- Header: `Authorization: Bearer <JWT_TOKEN>`
- Body:

```
{
  "title": "Database",
  "description": "DB Integration",
  "status": "completed"
}
```

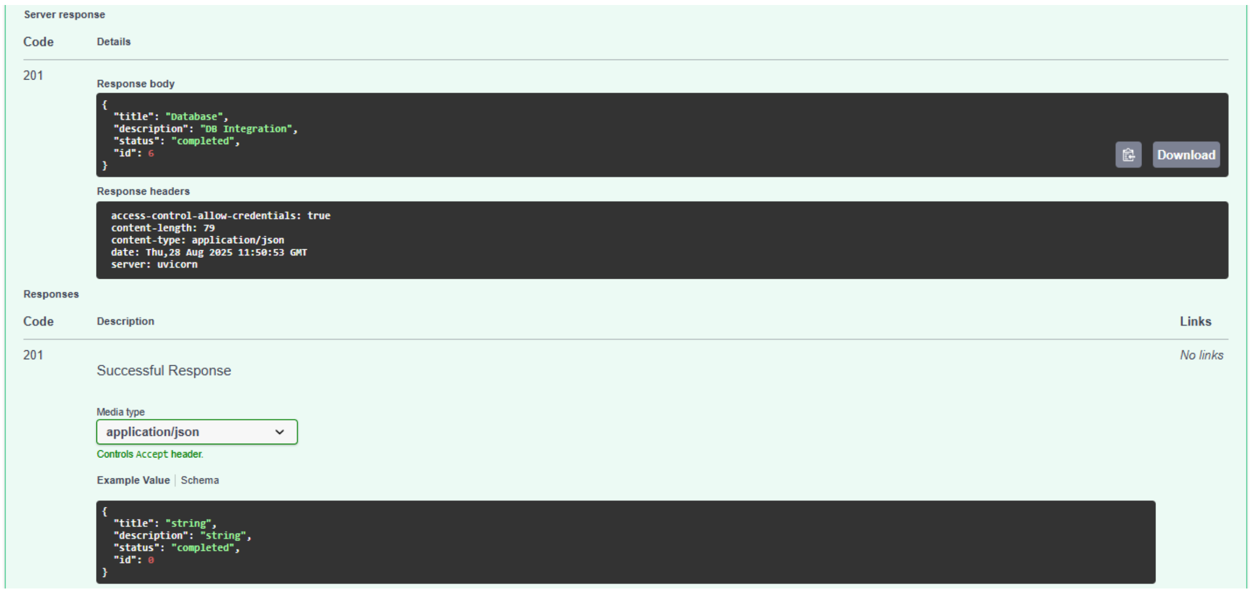
- Response:

```
{
  "id": 6,
  "title": "Database",
  "description": "DB Integration",
  "status": "completed"
}
```



```
{
  "title": "Database",
  "description": "DB Integration",
  "status": "completed"
}
```

Screenshot 4.3.5: Task Creation



201

```
{
  "title": "Database",
  "description": "DB Integration",
  "status": "completed",
  "id": 6
}
```

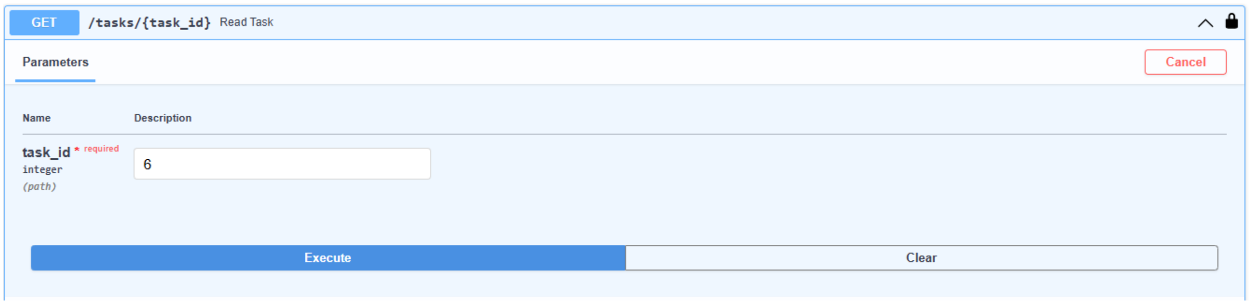
```
access-control-allow-credentials: true
content-length: 79
content-type: application/json
date: Thu,28 Aug 2025 11:50:53 GMT
server: uvicorn
```

201 Successful Response No links

```
{
  "title": "string",
  "description": "string",
  "status": "completed",
  "id": 0
}
```

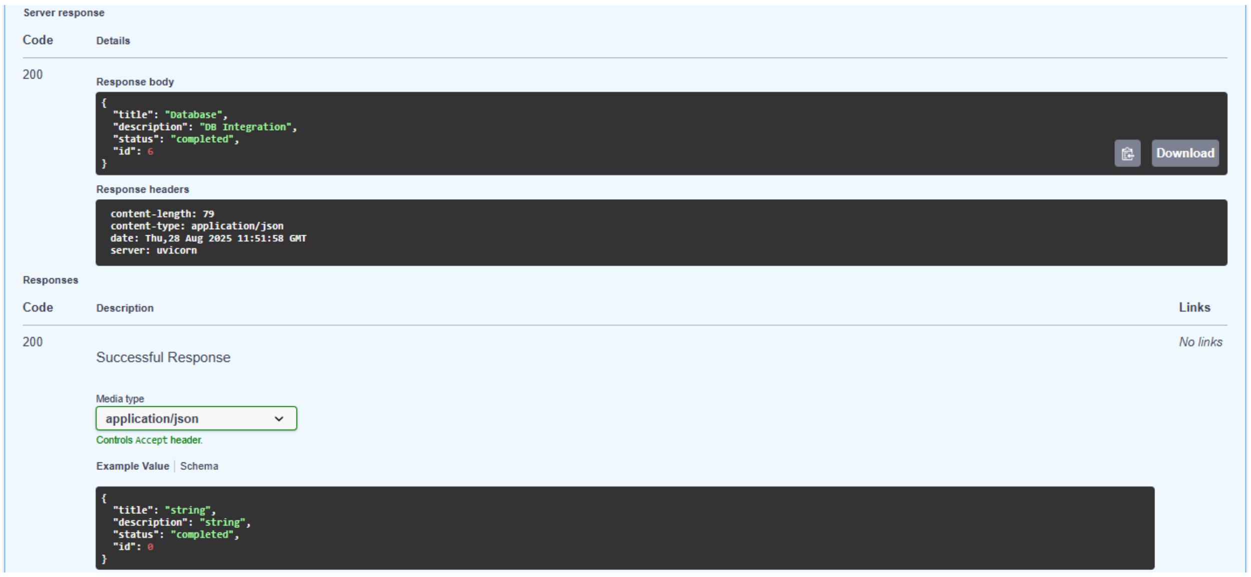
Screenshot 4.3.6: Response for Task Creation

Get Task:



task_id * required integer (path): 6

Screenshot 4.3.7: Verifying the Task with ID



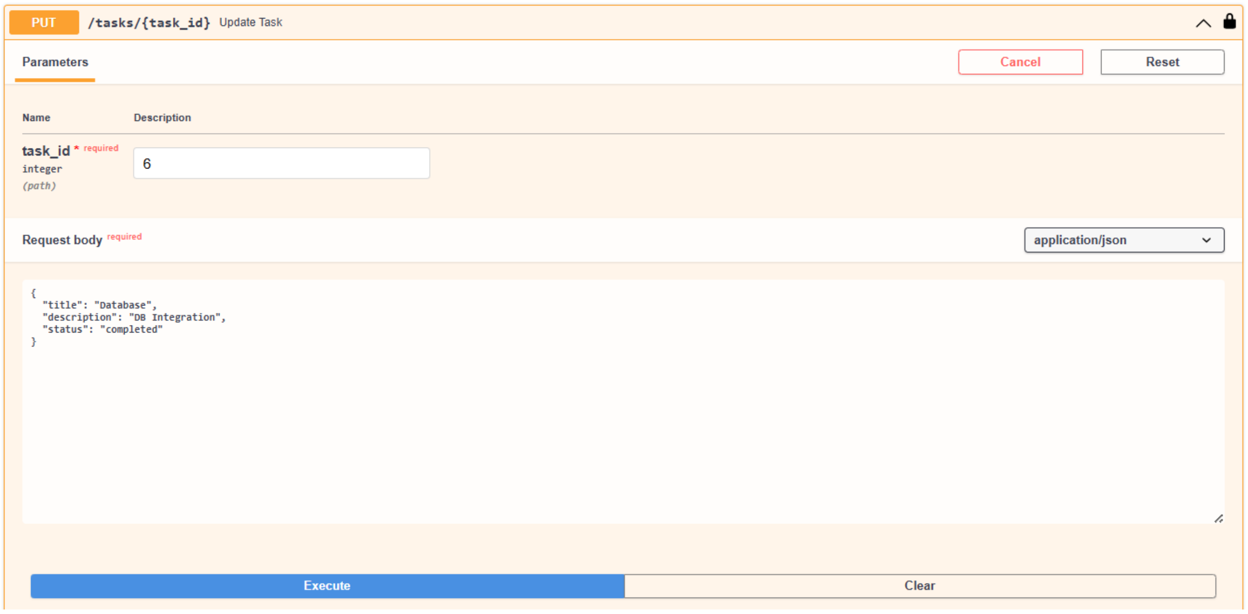
```
{
  "title": "Database",
  "description": "DB Integration",
  "status": "completed",
  "id": 6
}
```

```
content-length: 79
content-type: application/json
date: Thu,28 Aug 2025 11:51:58 GMT
server: uvicorn
```

```
{
  "title": "string",
  "description": "string",
  "status": "completed",
  "id": 0
}
```

Screenshot 4.3.8: Response when ID is present

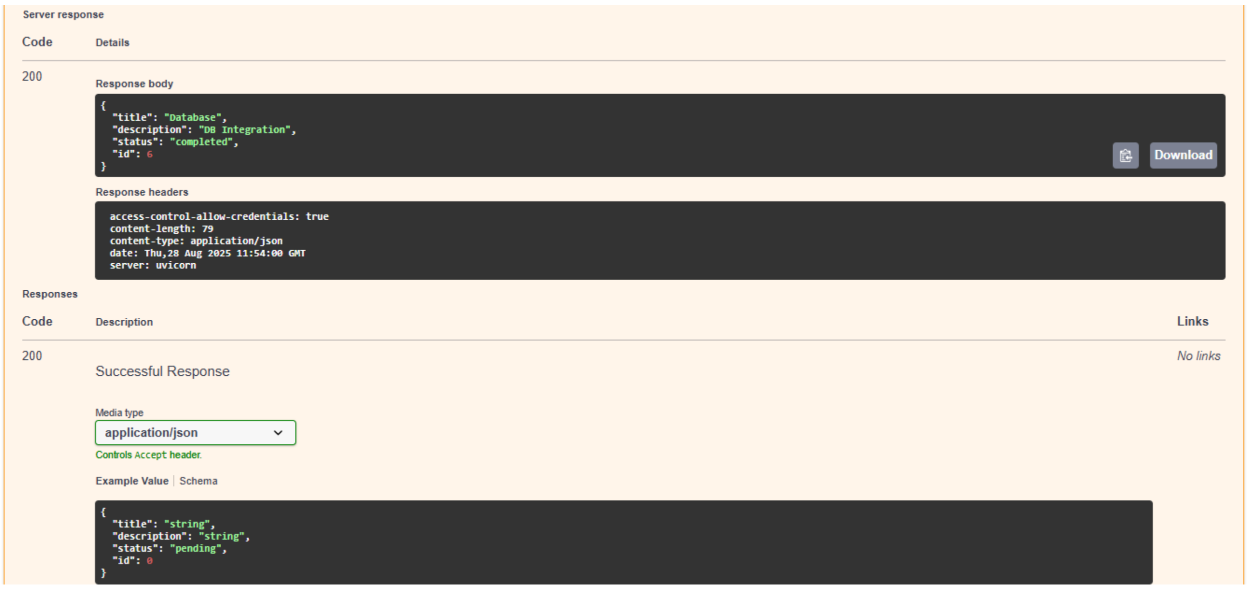
Task Update:



task_id * required
integer
(path)
6

```
{
  "title": "Database",
  "description": "DB Integration",
  "status": "completed"
}
```

Screenshot 4.3.9: Task Update with ID


Server response

Code | Details

200

Response body

```
{
  "title": "Database",
  "description": "DB Integration",
  "status": "completed",
  "id": 6
}
```

Response headers

```
access-control-allow-credentials: true
content-length: 79
content-type: application/json
date: Thu,28 Aug 2025 11:54:00 GMT
server: uvicorn
```

Responses

Code | Description | Links

200 | Successful Response | No links

Media type: application/json

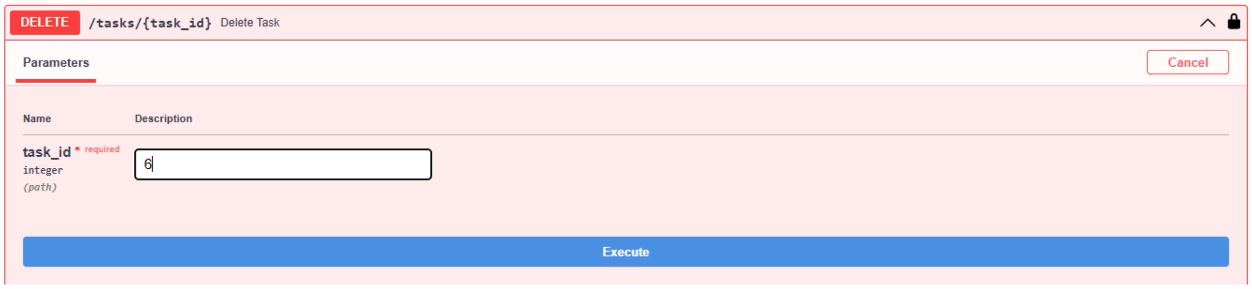
Controls Accept header.

Example Value | Schema

```
{
  "title": "string",
  "description": "string",
  "status": "pending",
  "id": 0
}
```

Screenshot 4.3.10: Response for Updated task

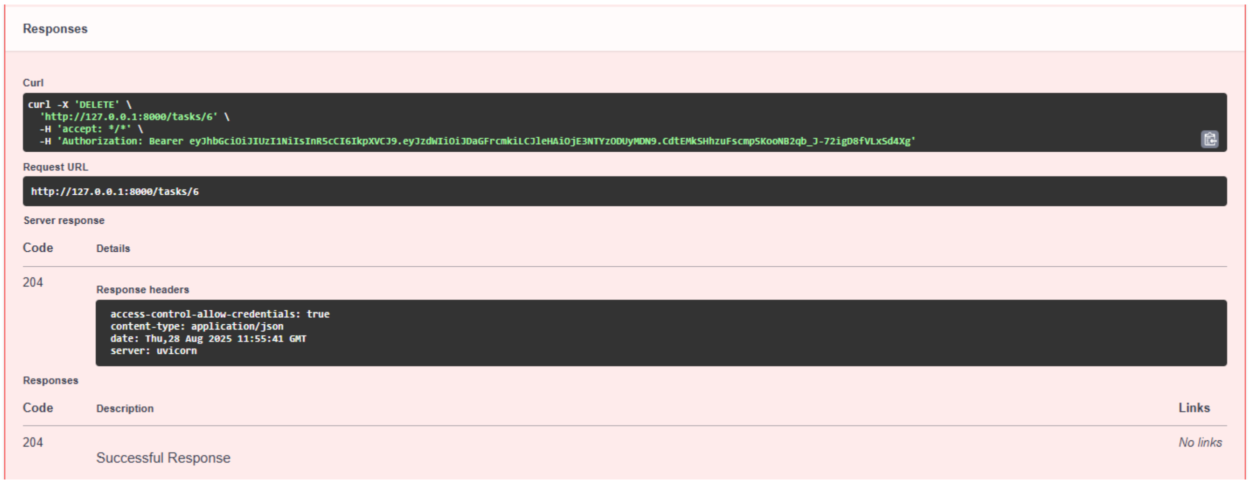
Task Deletion:


DELETE /tasks/{task_id} Delete Task

Parameters

Name	Description
task_id * required integer (path)	6

Screenshot 4.3.11: Task Deletion with ID


Responses

Curl

```
curl -X 'DELETE' \
  'http://127.0.0.1:8000/tasks/6' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJDaGFrcmkiLCJleHAiOjE3NTYzODUyMDN9.CdtEMkSHhzuFscmpSKooNB2qb_J-72igD8fVLxSd4Xg'
```

Request URL

```
http://127.0.0.1:8000/tasks/6
```

Server response

Code | Details

204

Response headers

```
access-control-allow-credentials: true
content-type: application/json
date: Thu,28 Aug 2025 11:55:41 GMT
server: uvicorn
```

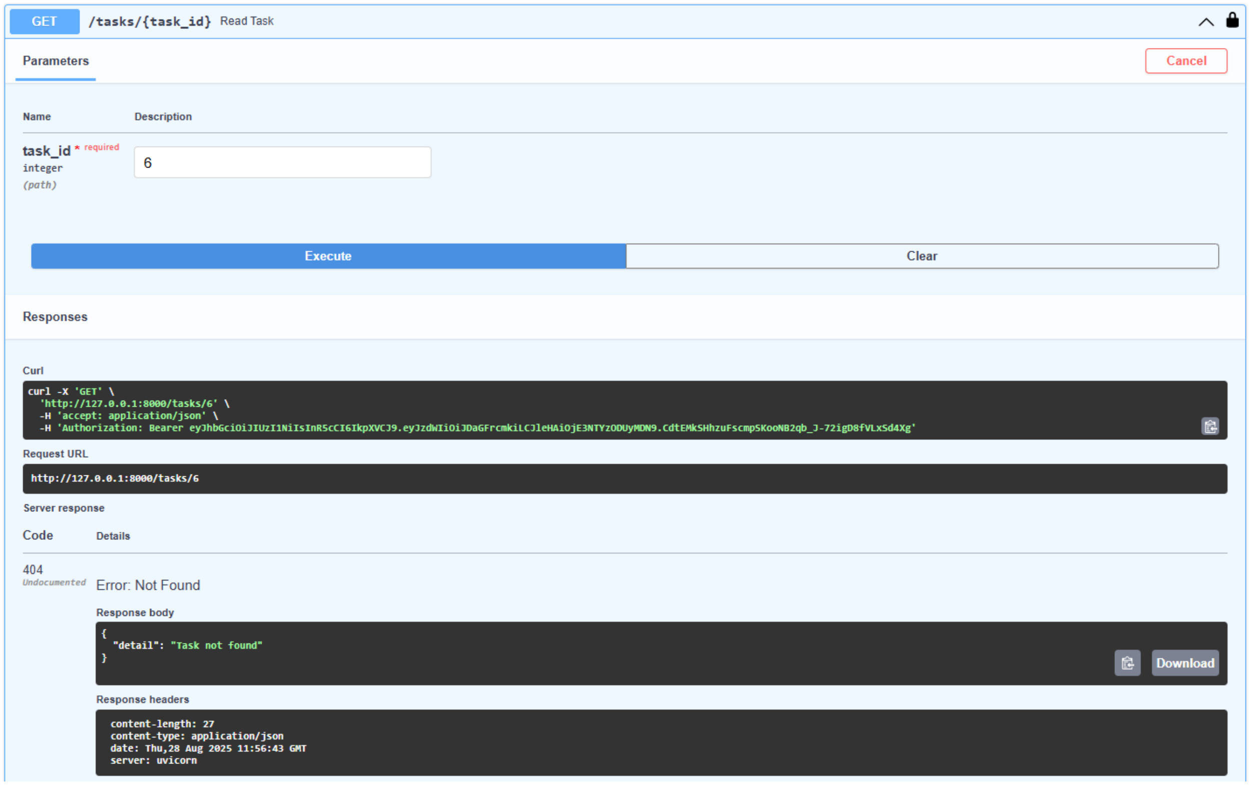
Responses

Code | Description | Links

204 | Successful Response | No links

Screenshot 4.3.12: Response for task Deletion

Get Tasks:



Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/tasks/6' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJDaGFrcmkiLCJleHAiOjE3NTYzODUyMDN9.CdtEMkSHhzuFscmpSKooNB2qb_J-72igD8fVLxSd4Xg'
```

Request URL

```
http://127.0.0.1:8000/tasks/6
```

Server response

Code Details

404 *Undocumented* Error: Not Found

Response body

```
{
  "detail": "Task not found"
}
```

Response headers

```
content-length: 27
content-type: application/json
date: Thu,28 Aug 2025 11:56:43 GMT
server: uvicorn
```

Screenshot:4.3.13: Task Searching with ID, It’s not present response 404 error

Database Details

5.1 Database Configuration

- **MySQL 8.0**
- **Database Name:** `task_manager`
- **User:** `root`
- **Password:** `password`

5.2 Tables

Users Table

COLUMN	TYPE	NOTES
ID	INT	PK, Auto Increment
USERNAME	VARCHAR 50	Unique
PASSWORD_HASH	VARCHAR 255	Hashed password

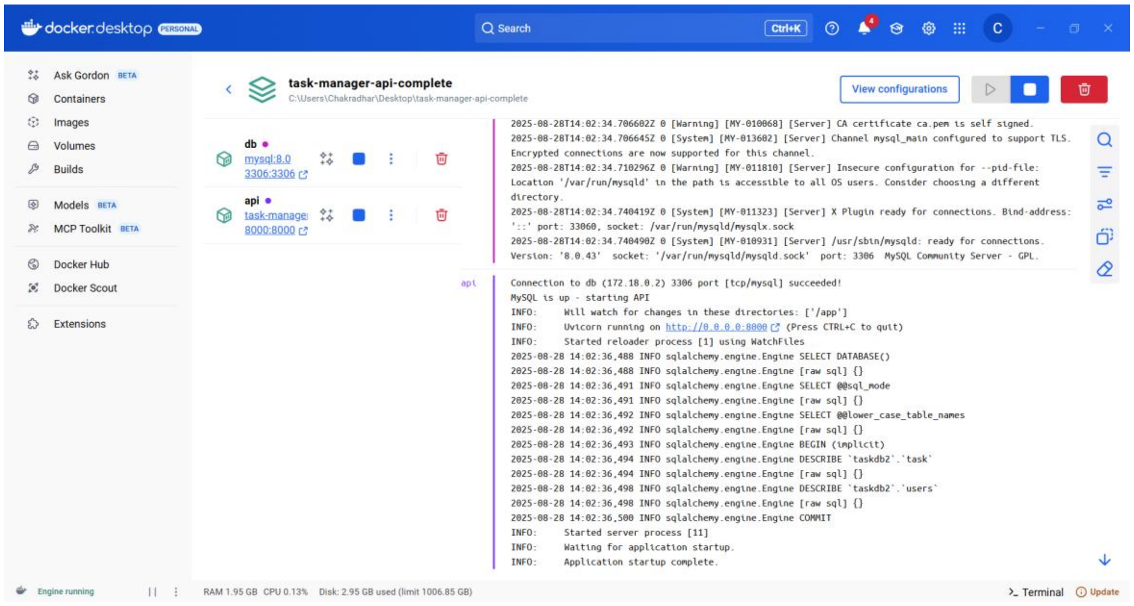
Tasks Table

COLUMN	TYPE	NOTES
ID	INT	PK, Auto Increment
TITLE	VARCHAR 100	Required
DESCRIPTION	VARCHAR 500	Optional
STATUS	ENUM	pending/in-progress/completed
USER_ID	INT	FK → users(id)

6. Docker Deployment

6.1 Backend + Database

```
# Build backend image
docker build -t task-manager-backend .

# Run backend container
docker run -d -p 8000:8000 --name task-backend task-manager-backend

# Run MySQL container
docker run -d -p 3306:3306 --name task-db -e MYSQL_ROOT_PASSWORD=password -e
MYSQL_DATABASE=task_manager mysql:8.0

# Check running containers
docker ps
```



```
2025-08-28T14:02:34.706602Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2025-08-28T14:02:34.706645Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS.
Encrypted connections are now supported for this channel.
2025-08-28T14:02:34.710296Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file:
Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different
directory.
2025-08-28T14:02:34.740419Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address:
'::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
2025-08-28T14:02:34.740490Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections.
Version: '8.0.43'  socket: '/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server - GPL.
```

```
api  Connection to db (172.18.0.2) 3306 port [tcp/mysql] succeeded!
     MySQL is up - starting API
     INFO:     Will watch for changes in these directories: ['/app']
     INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
     INFO:     Started reloader process [1] using WatchFiles
     2025-08-28 14:02:36,488 INFO sqlalchemy.engine.Engine SELECT DATABASE()
     2025-08-28 14:02:36,488 INFO sqlalchemy.engine.Engine [raw sql] {}
     2025-08-28 14:02:36,491 INFO sqlalchemy.engine.Engine SELECT @@sql_mode
     2025-08-28 14:02:36,491 INFO sqlalchemy.engine.Engine [raw sql] {}
     2025-08-28 14:02:36,492 INFO sqlalchemy.engine.Engine SELECT @@lower_case_table_names
     2025-08-28 14:02:36,492 INFO sqlalchemy.engine.Engine [raw sql] {}
     2025-08-28 14:02:36,493 INFO sqlalchemy.engine.Engine BEGIN (implicit)
     2025-08-28 14:02:36,494 INFO sqlalchemy.engine.Engine DESCRIBE `taskdb2`.`task`
     2025-08-28 14:02:36,494 INFO sqlalchemy.engine.Engine [raw sql] {}
     2025-08-28 14:02:36,498 INFO sqlalchemy.engine.Engine DESCRIBE `taskdb2`.`users`
     2025-08-28 14:02:36,498 INFO sqlalchemy.engine.Engine [raw sql] {}
     2025-08-28 14:02:36,500 INFO sqlalchemy.engine.Engine COMMIT
     INFO:     Started server process [11]
     INFO:     Waiting for application startup.
     INFO:     Application startup complete.
```

Docker Desktop Container contains Backend + MySQL

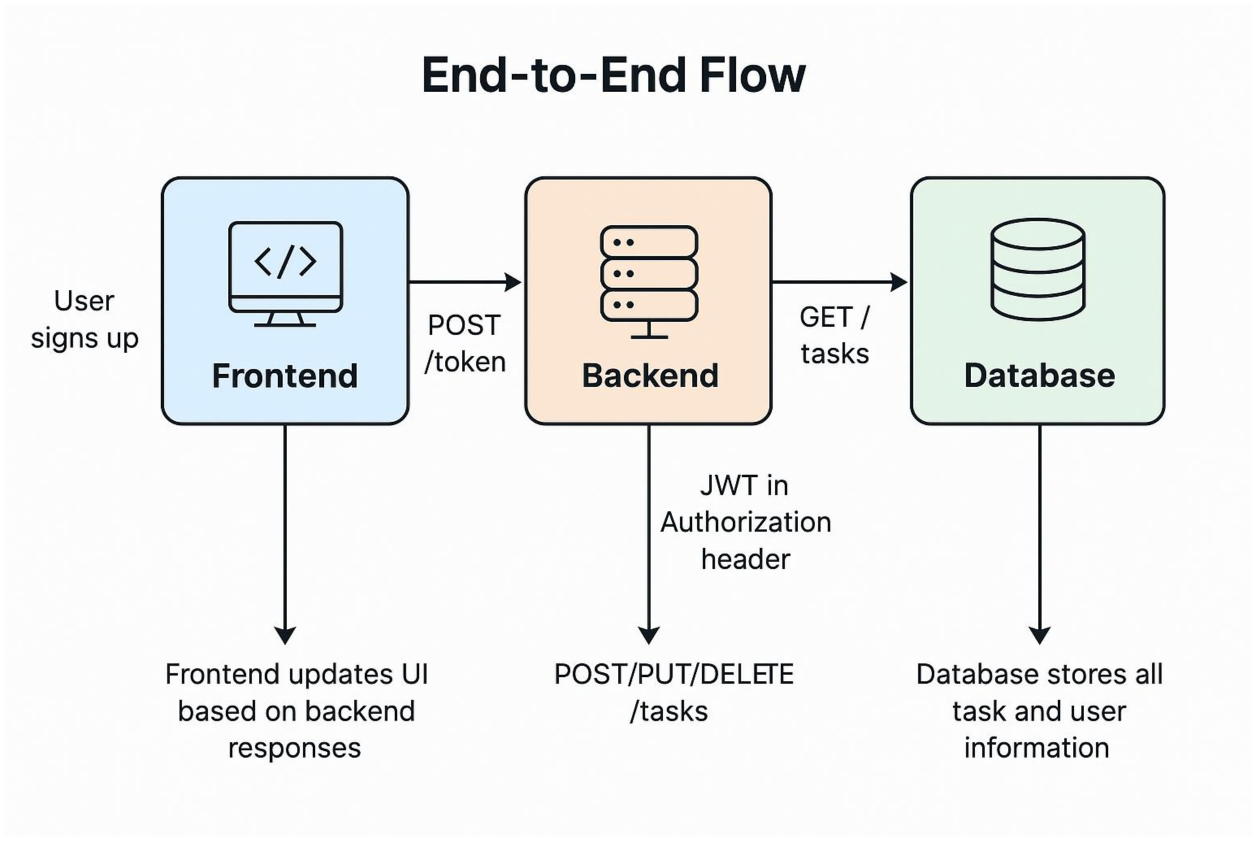
6.2 Frontend

```
# Navigate to frontend folder
cd task-manager-frontend

# Install dependencies
npm install

# Start frontend
npm run dev
```

- Access frontend: `http://localhost:5173/`
- Frontend communicates directly with backend via `http://localhost:8000`

7. End-to-End Flow

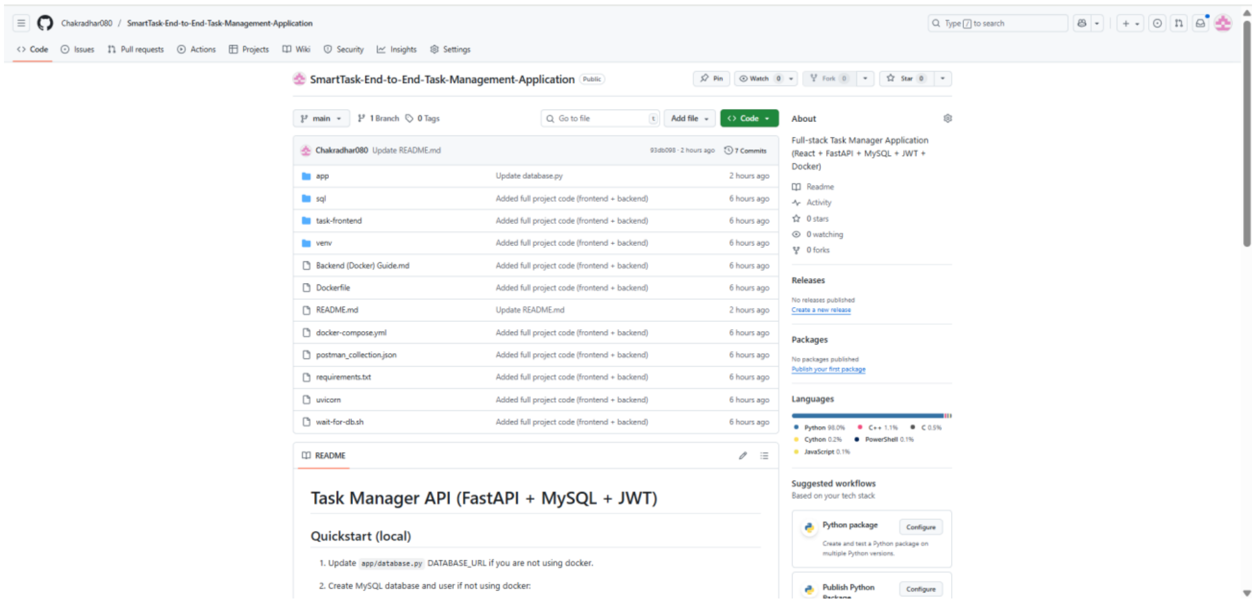
1. User signs up → POST `/signup`
2. User logs in → POST `/token` → receives JWT
3. User accesses task list → GET `/tasks` (JWT in Authorization header)
4. User creates/updates/deletes task → POST/PUT/DELETE `/tasks`
5. Frontend updates UI based on backend responses
6. Database stores all task and user information


End-to-End Flow
User signs up
Frontend
POST /token
Backend
GET / tasks
Database
JWT in Authorization header
Frontend updates UI based on backend responses
POST/PUT/DELETE /tasks
Database stores all task and user information

Screenshot 7.1: frontend ↔ backend ↔ database communication

8. GitHub Repository

- Repository contains **frontend**, **backend**, **database scripts**, **Docker setup**, **Postman collection**, and **documentation**
- GitHub URL: `https://github.com/Chakradhar080/SmartTask-End-to-End-Task-Management-Application.git`



SmartTask-End-to-End-Task-Management-Application Public

Chakradhar080 Update README.md 93db098 · 2 hours ago 7 Commits

app	Update database.py	2 hours ago
sql	Added full project code (frontend + backend)	6 hours ago
task-frontend	Added full project code (frontend + backend)	6 hours ago
venv	Added full project code (frontend + backend)	6 hours ago
Backend (Docker) Guide.md	Added full project code (frontend + backend)	6 hours ago
Dockerfile	Added full project code (frontend + backend)	6 hours ago
README.md	Update README.md	2 hours ago
docker-compose.yml	Added full project code (frontend + backend)	6 hours ago
postman_collection.json	Added full project code (frontend + backend)	6 hours ago
requirements.txt	Added full project code (frontend + backend)	6 hours ago
uvicorn	Added full project code (frontend + backend)	6 hours ago
wait-for-db.sh	Added full project code (frontend + backend)	6 hours ago

README

Task Manager API (FastAPI + MySQL + JWT)

Quickstart (local)

1. Update `app/database.py` DATABASE_URL if you are not using docker.
2. Create MySQL database and user if not using docker:

About

Full-stack Task Manager Application (React + FastAPI + MySQL + JWT + Docker)

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● Python 98.0% ● C++ 1.1% ● C 0.5%
● Cython 0.2% ● PowerShell 0.1%
● JavaScript 0.1%

Suggested workflows
Based on your tech stack

Python package — Configure
Create and test a Python package on multiple Python versions.

Publish Python — Configure

9. Deliverables

1. Full-stack project source code (frontend + backend + DB scripts)
2. Docker deployment files (Dockerfile, docker-compose.yml)
3. Swagger/OpenAPI JSON or Postman collection
4. Professional documentation (this file)
5. Screenshots showing frontend, backend, database, Postman, Docker

10. Conclusion

The SmartTask application successfully demonstrates a robust full-stack implementation of a task management system. It integrates a React-based frontend with a FastAPI backend and MySQL database, offering seamless task operations—creation, tracking, updating, and deletion. With JWT-based authentication and Dockerized deployment, the project showcases modern development practices and scalable architecture. The inclusion of Swagger and Postman documentation ensures clarity and ease of API testing, making it a comprehensive solution for task management.

---

11. Future Enhancements

To elevate the application further, consider implementing the following features:

- **User Roles & Permissions**: Introduce admin and user roles to manage access levels.
- **Task Deadlines & Reminders**: Add due dates and automated email or push notifications.
- **Collaborative Tasks**: Enable task sharing among multiple users for team-based workflows.
- **Priority Levels**: Allow users to set task priorities (e.g., High, Medium, Low).
- **Activity Logs**: Track changes made to tasks for better accountability.
- **Mobile Responsiveness**: Optimize UI for mobile devices or build a dedicated mobile app.
- **Analytics Dashboard**: Visualize task completion rates, user activity, and productivity metrics.
- **Multilingual Support**: Expand accessibility by supporting multiple languages.

12. References

1. React. (n.d.). *React – A JavaScript library for building user interfaces*. Retrieved from https://reactjs.org
2. Tiangolo, S. (n.d.). *FastAPI – Modern, fast (high-performance) web framework for building APIs with Python*. Retrieved from https://fastapi.tiangolo.com
3. Oracle Corporation. (n.d.). *MySQL 8.0 Reference Manual*. Retrieved from https://dev.mysql.com/doc
4. Docker Inc. (n.d.). *Docker Documentation*. Retrieved from https://docs.docker.com
5. JWT.io. . (n.d.). *Introduction to JSON Web Tokens*. Retrieved from https://jwt.io
6. Swagger.io. . (n.d.). *OpenAPI Specification*. Retrieved from https://swagger.io/specification
7. Postman Inc. (n.d.). *Postman API Platform*. Retrieved from https://www.postman.com
8. Python Software Foundation. (n.d.). *Python 3 Documentation*. Retrieved from https://docs.python.org/3/
9. REST API Tutorial. (n.d.). *RESTful API Design Guide*. Retrieved from https://restfulapi.net