Maria Desiree A. Ganohay March 04, 2025

BSIT-3R5

Todo List Application Documentation

Project Title: To-Do List App with FastAPI and React

This full-stack To-Do List application is built with FastAPI for the backend and React (Vite) for the frontend. It offers users the ability to:

- Add, edit, and delete tasks
- Mark tasks as completed
- Filter tasks by status (all, completed, or pending)
- Toggle between light and dark mode
- Persist data through a backend API and a database

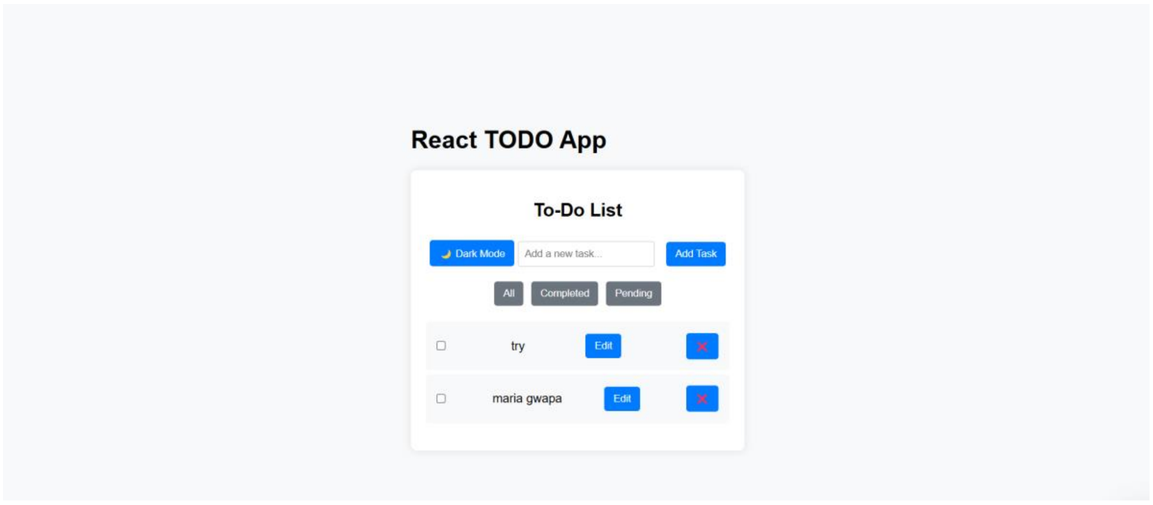
The backend features a RESTful API and is integrated with a database using SQLAlchemy with SQLite. SQLite was chosen for this project as the free PostgreSQL instance on Render had already been utilized for a previous task, and SQLite provided a lightweight, easy-to-use alternative that fits the development and small-scale deployment needs. The frontend communicates with the API using Axios and is designed for responsiveness with a theme-toggle feature.

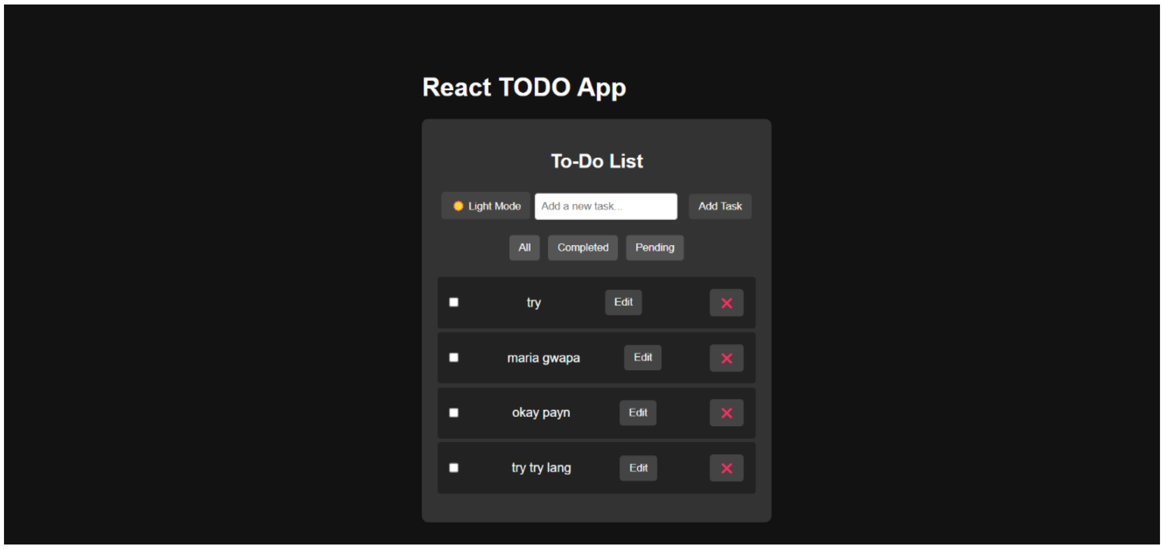
Feature	**FastAPI**	**Django REST Framework (DRF)**
Framework Type	Web framework focused on APIs	A toolkit for building APIs on top of Django
Performance	Very fast, asynchronous, and high-performance	Generally slower than FastAPI, synchronous
Asynchronous Support	Fully supports asynchronous programming	Limited async support, primarily synchronous
Ease of Use	Easy to use with automatic validation and docs	Easy to use but requires Django knowledge
Learning Curve	Shorter learning curve due to simplicity	Steeper, as it builds on Django's complexity
Validation	Automatic and built-in data validation	Needs serializers and custom validation

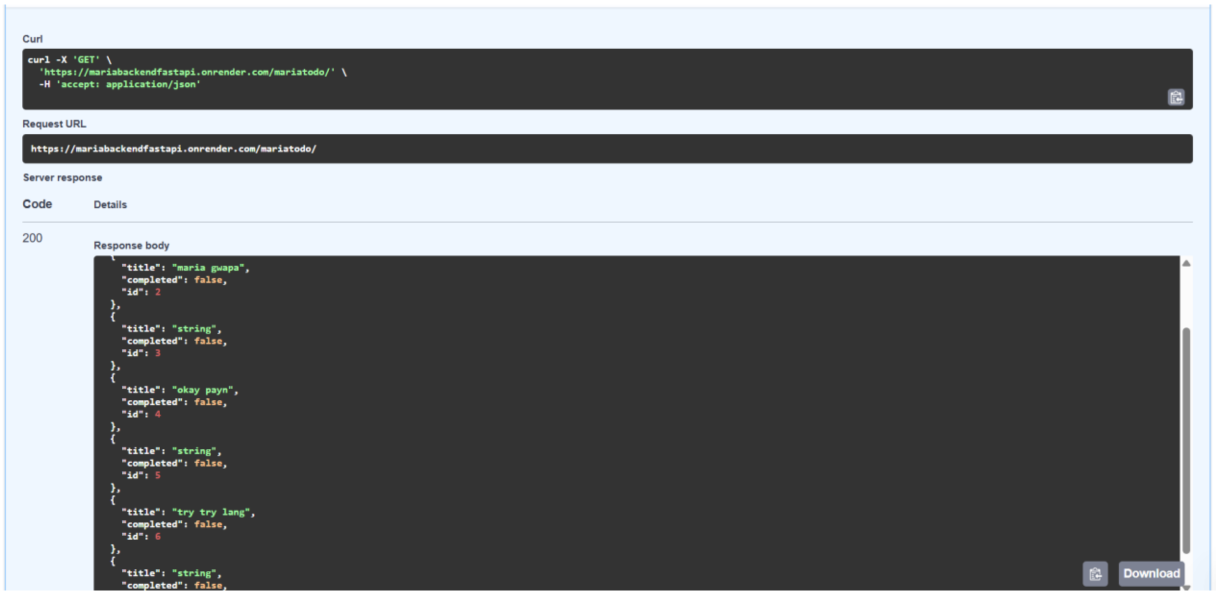
Technologies Used:

- Frontend: React with Vite
- Backend: FastAPI with SQLAlchemy
- Database: SQLite (since the free Postgres instance on Render was already used for a previous project)
- Deployment:
    - Backend deployed on Render
    - Frontend hosted on GitHub Page

Screenshots:






Curl

```
curl -X 'GET' \
  'https://mariabackendfastapi.onrender.com/mariatodo/' \
  -H 'accept: application/json'
```

Request URL

```
https://mariabackendfastapi.onrender.com/mariatodo/
```

Server response

Code Details

200 Response body

```
    "title": "maria gwapa",
    "completed": false,
    "id": 2
  },
  {
    "title": "string",
    "completed": false,
    "id": 3
  },
  {
    "title": "okay payn",
    "completed": false,
    "id": 4
  },
  {
    "title": "string",
    "completed": false,
    "id": 5
  },
  {
    "title": "try try lang",
    "completed": false,
    "id": 6
  },
  {
    "title": "string",
    "completed": false,
```

Live Links:

Backend: https://mariabackendfastapi.onrender.com/docs

Frontend: https://desireeshortiee.github.io/mariafront/

Challenges Faced:
Django REST Framework (DRF):
While working with DRF, one of the main challenges I encountered was related to deployment on Render. Initially, I faced some difficulties pushing the project to GitHub, which delayed the deployment process significantly. This issue took up more time than anticipated, and I had to troubleshoot several aspects of the project before I could successfully get it pushed to the repository. Once that hurdle was overcome, the rest of the deployment process went smoothly, but the initial setback did affect my timeline.

FastAPI:
On the other hand, when I transitioned to FastAPI, I found the experience to be much more straightforward. Thanks to my prior knowledge of DRF, I was able to adapt quickly and implement the necessary features with ease. FastAPI's simplicity and clear documentation were instrumental in allowing me to set up and deploy the backend without encountering any significant obstacles. The framework's design is user-friendly, and since I was already familiar with similar tools, I was able to navigate through the setup and deployment process swiftly and efficiently.