Name: Shaira Joy B. Nerio **Section:** BSIT – 3R5

Todo List Application Documentation

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

This full-stack To-Do List Application was built using FastAPI for the backend and React (Vite) for the frontend. The system allows users 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 using a PostgreSQL database via a RESTful API

The backend is designed with FastAPI, using SQLAlchemy for database operations and PostgreSQL as the primary relational database. The frontend connects to the API using fetch or Axios, delivering a responsive user interface with dynamic features and visual modes.

FEATURE COMPARISON: FastAPI vs. Django Rest Framework (DRF)

Feature	FastAPI	Django Rest Framework (DRF)
Framework Type	Web framework focused on APIs.	API toolkit on top of Django.
Performance	High-speed, supports asynchronous operations.	Synchronous, generally slower than FastAPI.
Async Support	Full async support out of the box.	Limited, mostly synchronous.
Ease of Use	Lightweight, easy to set up with auto docs.	Simple, but requires Django familiarity.
Validation	Built-in via Pydantic	Requires serializers for validation.

Learning Curve	Shorter, good for quick API development.	Steeper due to Django structure and setup.

TECHNOLOGIES USED

Frontend

- React (Vite)

Backend

- FastAPI with SQLAlchemy ORM

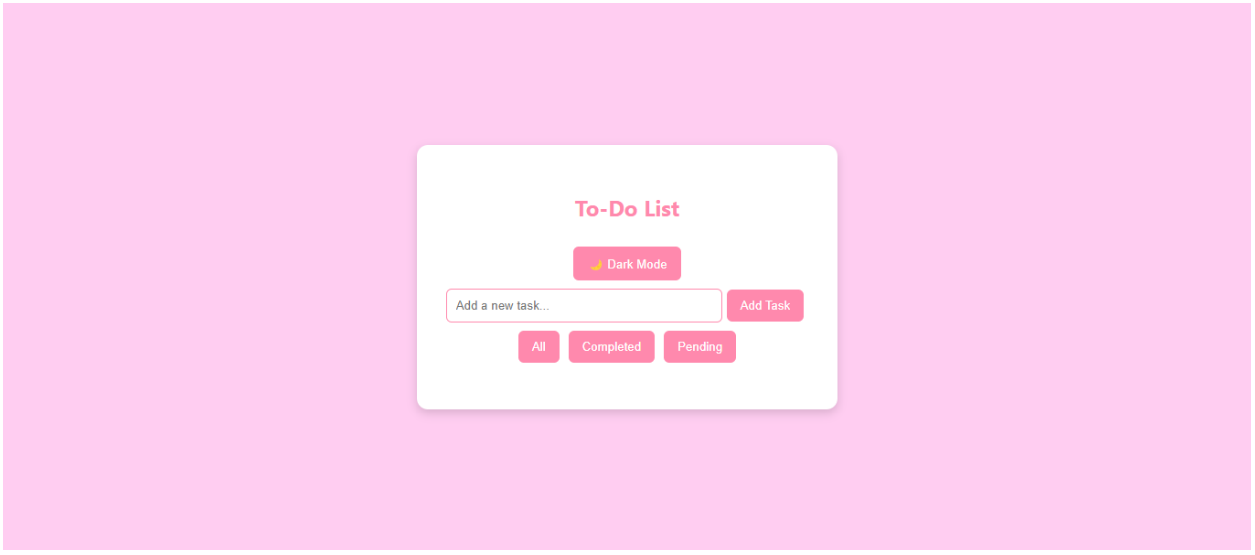
Database

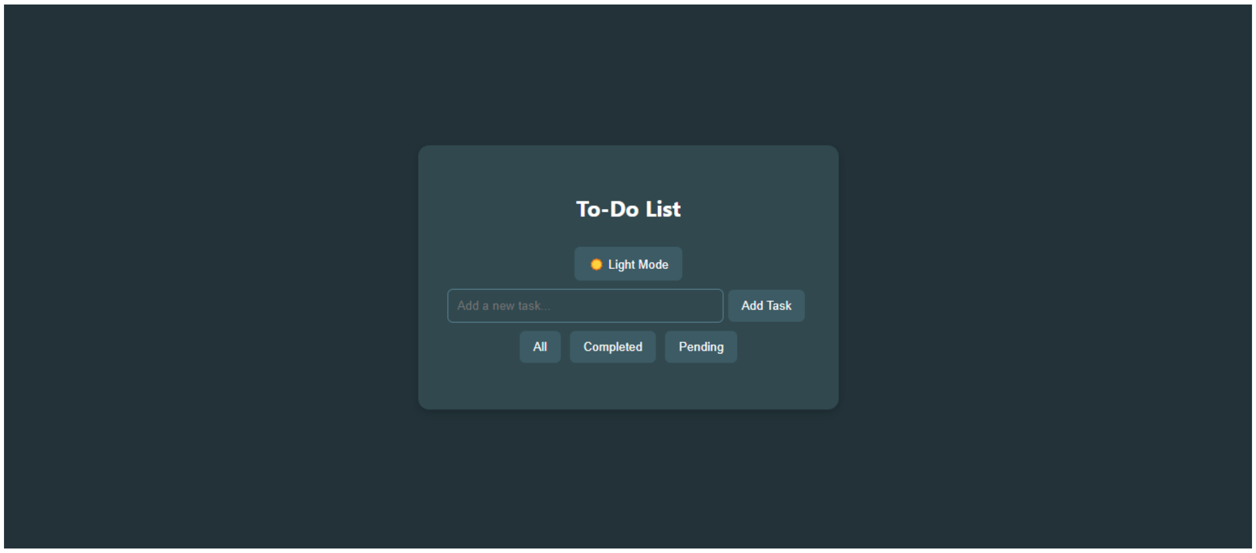
- PostgreSQL (deployed via Render)

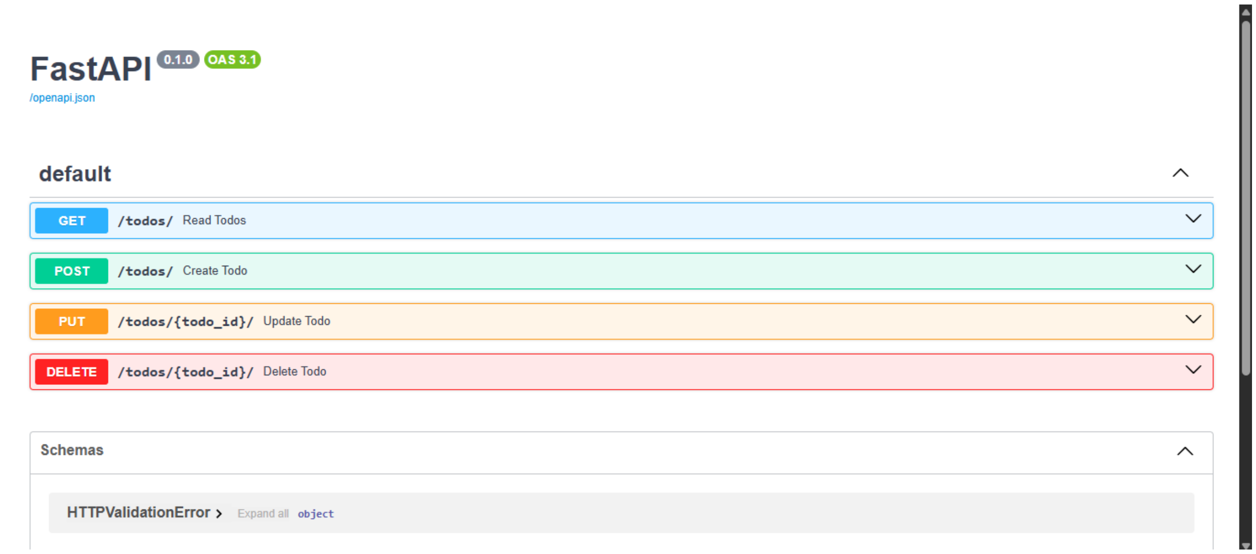
Deployment

- Backend on https://pit4-appdev.onrender.com/docs
- Frontend on https://shairanerio.github.io/reactjs-vite-django-webApp/

SCREENSHOTS






FastAPI 0.1.0 OAS 3.1

/openapi.json

default

GET /todos/ Read Todos

POST /todos/ Create Todo

PUT /todos/{todo_id}/ Update Todo

DELETE /todos/{todo_id}/ Delete Todo

Schemas

HTTPValidationError > Expand all object

CHALLENGES FACED

Django REST Framework (DRF):

One of the challenges with DRF was dealing with configuration and deployment. Django projects often require extra setup, especially for databases, CORS settings, and static files during deployment. But thankfully, we were taught by our Integrative Programming and Technologies about how to handle this kind of challenge in DRF.

FastAPI:

Using FastAPI was smoother. The automatic documentation (Swagger) helped a lot in testing endpoints. PostgreSQL integration through SQLAlchemy required correct URI formatting and CORS settings. Although I faced a challenge in deploying to Render because I am only allowed to manage one PostgreSQL instance per account on the free tier, so I have to disregard some of my past deployments to accommodate this project.