🌟 Performance Innovative Task Report: Full-Stack To-Do List App with FastAPI & React

🚀 Overview

This project demonstrates the integration of a **FastAPI** backend and a **React** frontend to build a fully functional **To-Do List Web Application**. It implements full **CRUD functionality**, **task filtering**, and a **dark/light mode toggle**. The application is cleanly structured, well-documented, and deployed with live links for demonstration.

📚 Instructions Overview

1. Set Up Your Development Environment

- Installed **Python**, **Node.js**, and created a **virtual environment** for the backend.
- Structured project into two main folders:
    - `/frontend` — React application
    - `/backend` — FastAPI application

2. FastAPI Backend Development

- Created FastAPI project with the following components:
    - `main.py` - App entry point
    - `models.py` - SQLAlchemy models
    - `schemas.py` - Pydantic models
    - `crud.py` - CRUD logic
    - `database.py` - PostgreSQL connection

- Defined To-Do item model:

```
class Todo(Base):
    id = Column(Integer, primary_key=True, index=True)
    title = Column(String, index=True)
    completed = Column(Boolean, default=False)
```

- Implemented endpoints:
    - `GET /todos/` - Retrieve all tasks
    - `POST /todos/` - Create a new task
    - `GET /todos/{id}` - Get a single task
    - `PUT /todos/{id}` - Update a task
    - `DELETE /todos/{id}` - Delete a task
    - `GET /todos?completed=true/false` - Filter tasks

3. React Front-End Development

- Built using **Vite + React + TailwindCSS**
- Functionalities:
    - Add, Edit, Delete tasks
    - Filter by: All | Completed | Pending
    - Dark/Light mode toggle
- Used `fetch()` API to connect with FastAPI
- Modular component structure:
    - `TodoInput`, `TodoList`, `TodoItem`, `TodoFilters`, `DarkModeToggle`

4. Integration & Testing

- Verified all CRUD and filter operations through:

  - FastAPI Swagger Docs
  - HTTPie / Postman
  - UI interaction
- Backend deployed on **Render**
- Frontend deployed on **Netlify**

📁 Expected Outputs (Delivered)

- ✅ Fully working To-Do List app
- ✅ CRUD operations
- ✅ Filtering by status
- ✅ Dark/Light toggle
- ✅ Live integration (React + FastAPI)
- ✅ Deployed frontend and backend

📊 DRF vs FastAPI Comparison

Feature	**Django REST Framework (DRF)**	**FastAPI**
Performance	Slower due to Django's stack	Very fast (Starlette + async support)
Learning Curve	Higher (Django's complex structure)	Easier for beginners
Type Checking	Limited	Full Python type hint support with Pydantic
Async Support	Limited (requires workarounds)	Built-in
Docs	Manual or DRF Browsable API	Auto-generated Swagger & ReDoc
Use Case	Enterprise-level, complex logic	Lightweight, modern APIs

Summary:

- **DRF** is best for complex, monolithic apps.
- **FastAPI** is ideal for fast, async, modern APIs.

🚫 Challenges & Solutions

❌ Deployment (Backend)

- **Problem:** Gunicorn failed with `TypeError: FastAPI.__call__() missing 1 required positional argument: 'send'`
- **Solution:** Switched to correct start command:

```
gunicorn app.main:app --worker-class uvicorn.workers.UvicornWorker
```

❌ ORM Migration

- **Problem:** Pydantic `orm_mode` deprecated
- **Solution:** Used `from_attributes = True` (Pydantic v2 fix)

❌ CORS Issues

- **Problem:** Frontend fetch failed due to CORS
- **Solution:** Enabled CORS in `main.py`:

```
from fastapi.middleware.cors import CORSMiddleware
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

❌ State Sync (Frontend)

- **Problem:** Tasks did not update instantly on actions
- **Solution:** Updated local state using React hooks (`useState`, `useEffect`)

📝 Submission Checklist

- GitHub Repo: https://github.com/RavenXconner/FastAPI-Todo-Fullstack
- Backend Live: https://todo-backend-i7yq.onrender.com/api/todos/
- Frontend Live: https://todo-frontend-netlify-clone.netlify.app
- Full project files (React + FastAPI)
- Complete README.md with instructions and endpoints
- PDF report (this doc)

✨ Conclusion

This full-stack To-Do List app demonstrates practical integration of FastAPI and React, deploying a modern, responsive, and interactive task manager with robust backend support.

It highlights key real-world development skills, including:

- API design
- Frontend-backend communication
- Deployment
- Debugging and async development with FastAPI

🌟 **Project Complete!**