BeyondChats – Technical Assignment

Overview

This repository contains my submission for the **BeyondChats Technical Assignment**. The goal of this assignment was to design and implement a small end-to-end system involving backend APIs, article processing, and a frontend UI under time constraints.

The project is divided into three logical parts:

- Backend APIs for managing articles
- A NodeJS-based article processing workflow
- A ReactJS-based frontend to display original and updated articles

Partial completion is intentional and aligned with the assignment instructions.

- **Backend APIs:** Python (FastAPI)
- **Database:** SQLite / PostgreSQL
- **Article Processing:** NodeJS
- **Frontend:** ReactJS
- **Version Control:** Git & GitHub

Backend Implementation Note

The assignment suggested using **Laravel** for backend APIs.
However, due to persistent local environment setup issues with PHP/Laravel within the limited time window, I made a conscious decision to implement the backend using **Python (FastAPI)** instead.

This decision allowed me to:

- Deliver fully functional CRUD APIs
- Maintain clean and readable API contracts
- Focus on system design, data flow, and integration

Project Structure

beyondchats-assignment/

```
│
├── backend/            # FastAPI backend (CRUD APIs)
│
├── node-processor/     # NodeJS article processing logic
│
```

```
├── frontend/            # ReactJS frontend
│
└── README.md            # Project documentation
```

Application Features

Phase 1 – Backend APIs (Completed)

- Scraped and stored blog articles in the database
- CRUD APIs for managing articles
- API to fetch latest articles
- Support for original and processed articles

Phase 2 – Article Processing (Partially Implemented)

- NodeJS project structure created
- Logic to fetch latest article from backend APIs
- Placeholder logic for:
    - Google search
    - Content scraping
    - LLM-based content transformation

Due to time constraints, external scraping and LLM calls are demonstrated through structure and documented logic rather than full production implementations.

Phase 3 – Frontend (Completed)

- ReactJS frontend to display articles
- Shows original and updated articles
- Responsive and clean UI

Data Flow / Architecture

BeyondChats Blog

↓

Backend APIs (FastAPI)

↓

Database (Articles)

↓

NodeJS Processor

↓

Updated Article Stored via API

↓

React Frontend UI

Local Setup Instructions

Backend (FastAPI)

cd backend

python -m venv venv

source venv/bin/activate # Windows: venv\Scripts\activate

pip install -r requirements.txt

uvicorn main:app –reload

Backend runs at:

http://localhost:8000

NodeJS Processor

cd node-processor

npm install

node index.js

Frontend (ReactJS)

cd frontend

npm install

npm start

Frontend runs at:

http://localhost:5173

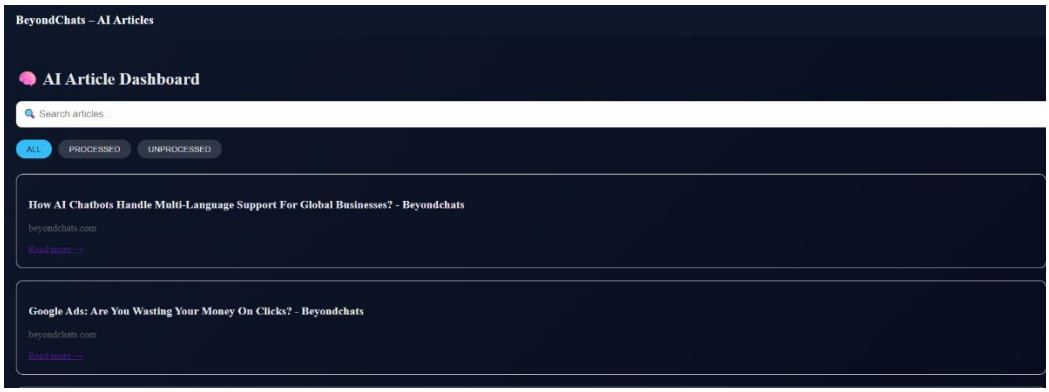
Github Link - https://github.com/Aditya-9753/BeyondChats

Frontend Live Link –

https://beyond-chats-nu.vercel.app/

The frontend is live; however, data is not displayed because it connects to a locally running backend API.

With the backend running locally, the application works end-to-end as expected. Screenshots have been added for reference.



Backend Live Link –

https://beyondchats-backend-cgyv.onrender.com/docs#/Articles/ingest_article_api_articles__post

Author

Aditya Tiwari

Email- adityatiwari2044@gmail.com
(Technical Assignment Submission)