Lotus Valley International School Gurugram

Valtam: a bot to solve student troubles

Jayan Sunil[a], **Rachit Rustagi**[b] **and Aarav Khandpur**[c]

[a]*IX — Everest, +91 9910856655*
[b]*IX — Nilgiris, +91 9211450648*
[c]*IX — Nilgiris, +91 9211612568*

Abstract—This project, Valtam, is a chatbot that is designed to solve the most frustrating that students face in their daily lives. From struggling to find assignments to needing help with their cumbersome homeworks, this chatbot can do it all — all while being reliable and easy to use.

Keywords—*shiv-nadar.vercel.app*

Contents

1. Introduction

Welcome to Valtam, the chatbot to solve it all. This bot is designed to fulfil your every need as a student: it can find your homework, and it can help you solve it. Utilizing modern frameworks, the bot entails a clean, functional UI, making the power of Python accessible to the layperson. This document is a guide to the inner working of Valtam.

2. Technologies Used

1. NextJS (frontend)
   - Shadcn/ui (UI)
   - Better Auth (User Authentication)
2. Python (backend)
   - API calls — FastAPI
   - LLM — PyTorch and Transformers (HuggingFace)
3. PostgreSQL (database)
   - ORM — Drizzle
   - DB provider — Vercel Postgres (Neon)
4. Telemery and Analytics
   - Telemetry & Error Reporting — Sentry
   - Analytics — Posthog

2.1. Frontend

For a fast and responsive frontend, we used NextJS. Its vast ecosystem, and reliability were major perks. It allowed us to create many complex functions effectively. To make NextJS even better, we used Shadcn, a popular UI library that provides accesible base components. For authentication, we used BetterAuth, an open-source authentication library that gives us support for many social logins like Google and Github.

2.2. Backend & Database

backend has two parts: the LLM and the API. The LLM is powered by PyTorch and Transformers, where we have trained a base model to cater to a student's needs. There are two trained models: one that is only for the NCERT and one that also has the assignments. To allow the frontend to interact with the LLM, we have a FastAPI wrapper around the model. The API has two routes: `/generate/{ncert,assignments}`, which take a request body of

```
{
  prompt: string;
  temperature: number (defaults to 0.7);
  max_length: number (defaults to 128);
}
```

Code 1. Request Body for /generate/*

2.3. Model

To ensure that our chatbot is relevant to the academic context of Indian school students, we fine-tuned the LLM using a dataset of NCERT textbook explanations. We used two publicly available NCERT-based datasets for Science and Social Science with approximately 8k rows on 3 epochs.

We chose the Qwen 0.6B model for this project.

The training pipeline used the NCERT Database combined with a database which has queries from our school portal

- **Tokenizer:** AutoTokenizer from Qwen model
- **Max Length:** 128 tokens
- **Epochs:** 3
- **Precision:** FP16 (mixed-precision training)

We used PyTorch for low-level control and trained the model locally on our GPU. After training, the model showed improved performance on curriculum-specific queries, demonstrating a solid understanding of academic content.

2.4. More Information

The bot's main use case is supposed to be in QA format(Question-Answering), as it is supposed to answer queries and provide help to students Therefore, the data was organised into a Question-Answer format for optimal training. The Qwen 0.6B (752M params) model offers speed, balance and accuracy. To training this model, we used an RTX 4070 GPU on a machine with the following specs.

- CPU: i5-13500k
- RAM: 16GB
- GPU: RTX 4070
- Storage: 1TB NVME 4.0 SSD

For the model with only the NCERT, the training loss metric is shown in Table 1.

Table 1. Training Loss for model 1:

Step	Training Loss
500	1.312500
1000	0.921400
1500	0.623400
2000	0.291200
2500	0.202200
3000	0.155800
3500	0.109400
4000	0.069900
4500	0.059500

The database is a PostgreSQL database, which is used to store user information and their chat history. We use Vercel Postgres, with

Neon, for a fast and scalable experience. As the ORM, we use Drizzle, which is type-safe, lightweight, and has a SQL-centric approach to interacting with databases, making it ideal for modern TypeScript.

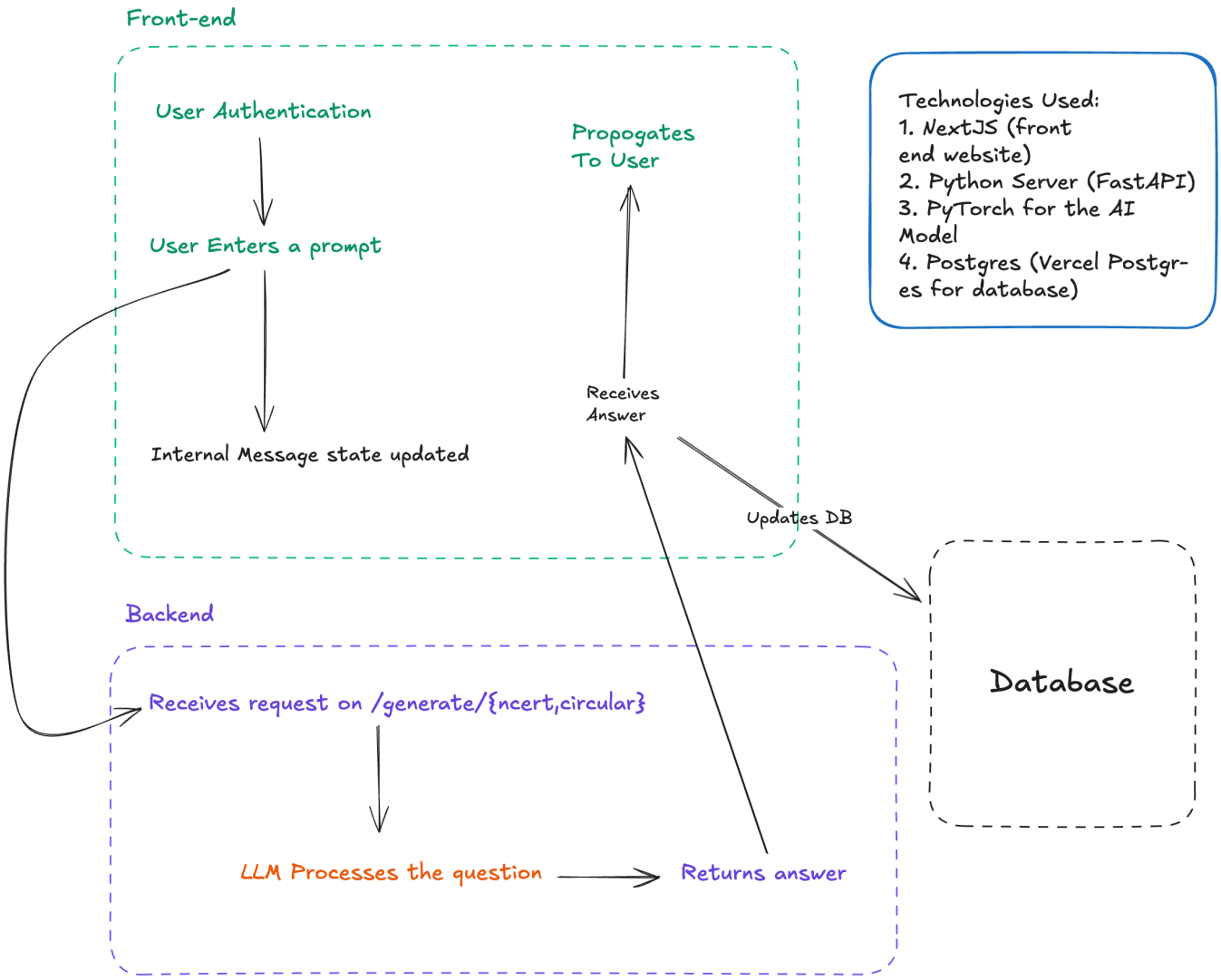

Figure 1. A flow chart of the application lifecycle.