AI Tutor

Technical Report

Spencer Thompson

Table of Contents

1. Introduction

This project, the AI Tutor, is a project that has now been under development for quite some time. We have seen some exciting success regarding utilizing artificial intelligence in real world application.

This report outlines Why, How, and What this project is all about, with a focus on the technical design, implementation and solution.

1.1. Background

As previously mentioned, the AI Tutor project started back in early 2024. In discussions with the excellent faculty in Tech Management Department at Utah Valley University, we hypothesized that using the new and exciting technology of large language models, we could provide excellent, *personalized* tutoring to students 24/7.

So we set out to develop a simple application to accomplish this goal. The original AI Tutor was indeed a success. Quickly the project gained attention among multiple departments and more than a handful of students. Although, given the rather breakneck pace of development, there was a rather rapid accumulation of technical debt. Some of the core features that we wanted had become quite difficult.

After the original scope of the project had been completed, we still had a desire to have a system that was better suited for both students and professors. As the lead developer, I had the feeling that a fresh start might be better than attempting to pay down a significant amount of technical debt. Therefore, this report is particularly focused on:

- The evolution of the project as a whole.
- The features that gives the AI Tutor an advantage over **every other alternative**.
- The design and implementation of the tutor, as well as the challenges faced along the way.

1.1.1. Context

A rather interesting piece of context to keep in mind while reading this report is that: during the duration of this project, every single piece of technology has changed **drastically**. Oftentimes the APIs or services that we were utilizing evolved or changed overnight. This could possibly be attributed to the incredible amount of development and hype around the use of generative AI.

The point being, many other competitors both at our own university and others, were quickly building and iterating on similar ideas. Our team built and deployed **two** complete iterations, while other teams have yet to deploy their projects.

1.1.2. Purpose

From the beginning our project was focused on providing a rather niche ideal. We all had this idea of a tutor or assistant that had intimate knowledge of a student's courses.

1.2. Goals

Primarily we wanted:

- 24/7 access to students to assist with coursework.
- Personalized responses to student questions that were unique to *each* student.

- Answer questions about the syllabus, upcoming assignments, grades and more.

The hope and idea being that, this could be an incredibly valuable resource for students, faculty and the university as a whole.

1.3. Limitations

During the duration of this project, there were really two main issues.

1. Incredibly Busy Semester.

2. Canvas API Key Permissions.

The first problem is simply that the developer team and I had incredibly difficult and loaded semesters. In addition, we had several other job opportunities open up. Given that graduation is just over the horizon, it is absolutely true that this somewhat hindered our development.

Regardless though, this first problem pales in comparison to the other. Our university had a competing team that was also working on an AI assistant. We diligently attempted to work with this other team, over the course of a year, but they were obstinate, they insisted that they wanted to hire outside talent.

Part of this problem was that they were unwilling to give us an API Key. This could also have been university permissions issue.

Not having a proper API Key was a truly difficult challenge. Especially given that our team had already deployed to active courses, while they had not.

2. Design

The overall design of the project can be a bit confusing at first. Really there are **five** major distinct pieces.

- **Browser Extension**
  - ‣ For user authentication and communication with the Canvas API.

- **Frontend**
  - ‣ The chat interface for using the AI Tutor.

- **Backend**
  - ‣ Where all the data processing and external API calls happen.

- **Analytics**
  - ‣ The service that monitors and displays the user Telemetry data.

- **Database**
  - ‣ Where we store everything we need.

2.1. Data

The order in which each piece will be explained is the rough order that they function. As a whole, the system is essentially two data pipelines with a unified user interface. Essentially these two pipelines are:

- Analytics
  - ‣ Telemetry and usage data gathered from students.

- Canvas
  - ‣ Student data regarding assignments, courses, submissions, etc.

2.1.1. Pipeline

While the analytics data is interesting and deserves its own time in the spotlight, it is not the focus of the project.

The Canvas data is really where things get interesting.

- I knew that we needed to grab data from the Canvas LMS, which is [Open Source](#). After thoroughly reading the documentation and looking into the source code, I knew I was on the right track.
- We also needed to process that data and cross reference it with our own users.
- Lastly, we needed a way to securely store and retrieve our mix of user data and course data.

$$\text{Canvas} \;\rightarrow\; \text{Our Code} \;\rightarrow\; \text{Storage}$$

2.1.2. Analytics

From the beginning, we also knew that we wanted to get some data to attempt to answer the question:

Does AI Tutoring help students?

In addition to the data pipeline, we also needed a smaller pipeline to get user telemetry data to dive into answering this question.

2.2. AI Behavior

From the first iteration of the AI Tutor, we knew that the behavior of the AI Tutor itself was particularly important. It was difficult just to get the tutor not to send super long messages, let alone get it to answer questions with a high accuracy. W

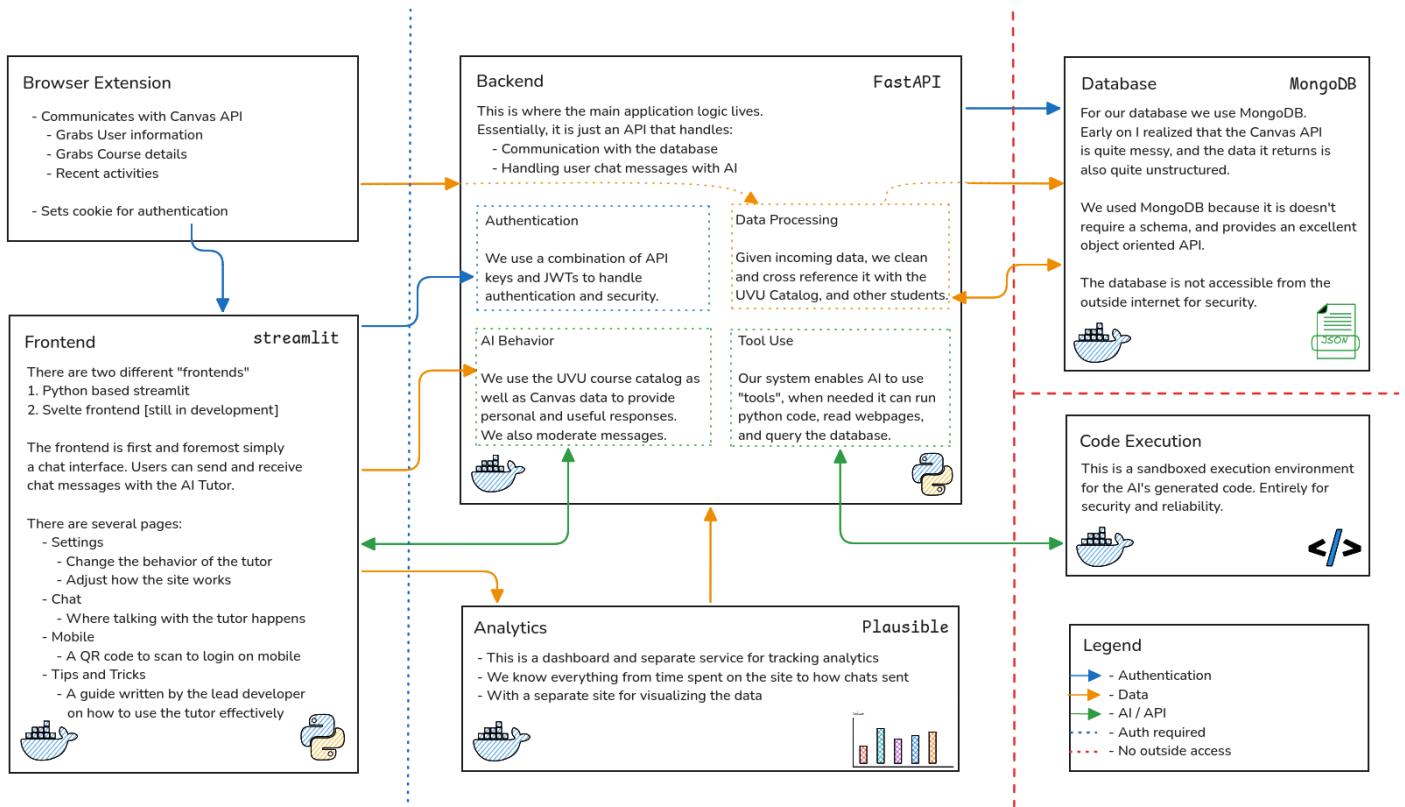# 3. Implementation

- Lessons Learned from first iteration

Browser Extension

- Communicates with Canvas API
  - Grabs User information
  - Grabs Course details
  - Recent activities

- Sets cookie for authentication

Backend   FastAPI

This is where the main application logic lives.
Essentially, it is just an API that handles:
  - Communication with the database
  - Handling user chat messages with AI

Authentication

We use a combination of API keys and JWTs to handle authentication and security.

Data Processing

Given incoming data, we clean and cross reference it with the UVU Catalog, and other students.

AI Behavior

We use the UVU course catalog as well as Canvas data to provide personal and useful responses. We also moderate messages.

Tool Use

Our system enables AI to use "tools", when needed it can run python code, read webpages, and query the database.

Database   MongoDB

For our database we use MongoDB. Early on I realized that the Canvas API is quite messy, and the data it returns is also quite unstructured.

We used MongoDB because it is doesn't require a schema, and provides an excellent object oriented API.

The database is not accessible from the outside internet for security.

Code Execution

This is a sandboxed execution environment for the AI's generated code. Entirely for security and reliability.

Frontend   streamlit

There are two different "frontends"
1. Python based streamlit
2. Svelte frontend [still in development]

The frontend is first and foremost simply a chat interface. Users can send and receive chat messages with the AI Tutor.

There are several pages:
  - Settings
    - Change the behavior of the tutor
    - Adjust how the site works
  - Chat
    - Where talking with the tutor happens
  - Mobile
    - A QR code to scan to login on mobile
  - Tips and Tricks
    - A guide written by the lead developer on how to use the tutor effectively

Analytics   Plausible

- This is a dashboard and separate service for tracking analytics
- We know everything from time spent on the site to how chats sent
- With a separate site for visualizing the data

Legend
- → - Authentication
- → - Data
- → - AI / API
- ···· - Auth required
- ---- - No outside access

Figure 1: AI Tutor Infrastructure Diagram

3.1. Infrastructure

- Hetzner VPS
- Cost reduction
- Docker & Compose

3.2. Security

- JWT
- Keys
- SSH
- 2FA?

3.3. Frontend

- User Interface
- Prioritize good User Experience

3.3.1. Browser Extension

- Side panel
- Cookies

3.3.2. Chat Interface

- Streamlit

3.4. Backend

- FastAPI

3.4.1. AI

- OpenAI API

3.4.2. Tools

- Catalog
- Upcoming Assignments
- Grades

3.4.3. Telemetry

- Plausible Analytics

3.5. Database

- MongoDB
- Document Based

4. Challenges

4.1. Permissions

- UVU
- Competing Team

4.2. API Inconsistency

4.2.1. Canvas LMS

- Name vs Course Code

4.2.2. Plausible Analytics

- Weirdest API I have ever seen
- Cartesian products galore

4.3. Messy Data

- Canvas
- My own Design

4.3.1. Live Service

- Developing / Adding features for an application in use is difficult

5. Solutions

5.1. Workarounds

- Browser Extension

5.2. Clever Tricks

5.2.1. Authentication
- JWT / Cookie

6. Conclusion

6.1. Findings

6.2. Implications