Design Document

Objective

The objective of this document is to define a detailed plan for developing an Undergraduate Advising Assistant. This system aims to streamline and enhance the academic advising experience for users by leveraging generative AI technology. While users could include students, academic advisors and system administrators, the student role would be considered as as the main persona for our software implementation.

Requirements

User Input:

- Checklist Submission: Students are able to submit their checklist populated with their registered classes for the given semester.
- User prompt: Users including students are able to interact with a chatbot to receive course recommendations and personalized guidance based on the student academic performance

Database Query:

The system will store and retrieve:

- User login information for authorization and authentication purposes
- Checklists populated with classes/courses registered for the given semester
- Old personalized recommendations and suggestions from generative AI model.

Gen AI Calls:

- Personalized Advising: Provide recommendations and suggestions while implementing proactive interventions to ensure that students remain on course for timely graduation.
    - Inputs: User prompt tailored toward personalized academic advising
    - Outputs: Recommendations and suggestion prompt based on the input prompt from the user.

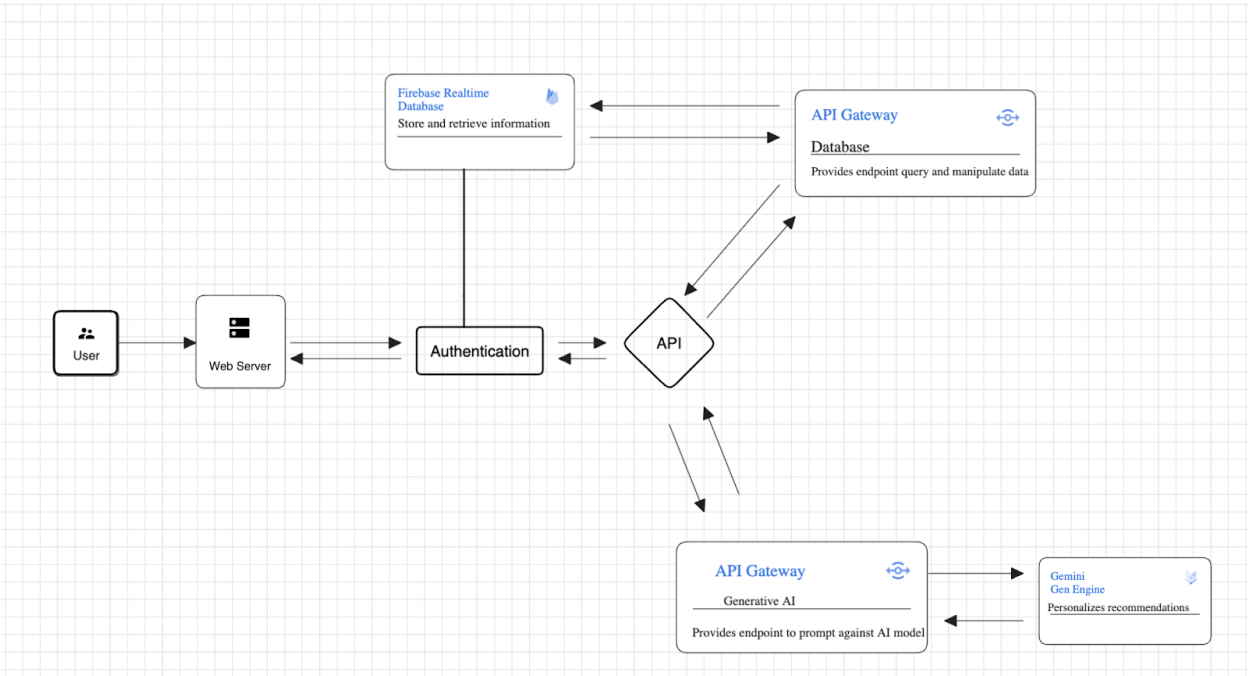
Background

At my home university, the academic advising for undergraduate students encounters difficulties in delivering personalized guidance and assistance efficiently due to the large student population and time constraints. The conventional advising approach is often time-consuming and lacks the use of new technological advancements to students' experience in terms of accessibility and quality of advisory services, hence the need to incorporate cutting-edge technologies to provide more effective support to undergraduate students

Design Ideas

The following design ideas would be considered for the software design specification and implementation:

- Streamlit would be used to design a user-friendly web application interface using widgets including inputs, buttons, dropdowns, and tables.
- A suitable database from Google Cloud Platform (GCP) like Firestore for instance would be used to store and retrieve relevant information as described in the requirements section.
- Database API would be used to make calls to query and manipulate data stored in the database using CRUD operations as appropriate.
- The functionality of the generative AI model would be exposed through the use of an API. API would provide endpoints to receiving information from streamlit, processing information using suitable generartive AI models and returning the generated output.
- API for generative AI engine would be integrated with the backend of the system. When students submit a query or requests using a chatbot for instance, the backend can make calls to the gen AI API to generate personalized recommendations based on inputs and responses stored in the database for future use and analysis.
- Application is then deployed using Streamlit deployment option or Google Cloud platform through a containerization technique.



Alternative Considered

- Flasks and Django where other web framework for building web applications in Python. Streamlit was however preferred over them since Streamlit allows you to build data-driven applications with minimal boilerplate code providing a simple and interactive API for creating interactive web apps using python scripts.
- User credentials for user authentication and authorization could have been loaded into Streamlit authenticator package from a pickle file for user authentication and authorization stored. While this technique is feasible, it's not the efficient since there is poor mechanisms for data integrity checks, limited security control, and low scalability and maintainability.
- Besides Google Cloud Platform(GCP), other popular alternatives include Amazon Web Services(AWS), Microsoft Azure, and Oracle Cloud Infrastructure(OCI), to name a few. Not only does GCP offer several unique features and capabilities with efficient use cases but also provide a cost-free service for a free credits through the google exchange program.

Prompts

- What is Streamlit and what is it used for?
- In the context of making api calls at the backend to fetch data in the database, what conponents would be the api and what could be the example?

- Why is loading Information including user credentials for user authentication and authorization for the use in Streamlit authenticator package from a pickle file for user authentication and authorizationstored a bad practice
- How do I integrate a gen AI engine into our system?
- What other popular web application tools could have been used instead of streamlit?
- How does streamlit compare with Flask and Django?