Course Advisor System Using AI

Adam Noor, Ellora Devulapally, Esther Cheng, Jack Munson, Huy Nguyen

ABOUT

Our project aims to enhance the course advising system at Duke by introducing a chatbot designed to provide 24/7 support to students. The chatbot employs semantic search to identify relevant courses, resources, and requirements to empower students to make the most of their Duke experience.

Team Members

Adam Noor · Jack Munson · Huy Nguyen · Esther Cheng · Ellora Devulapally

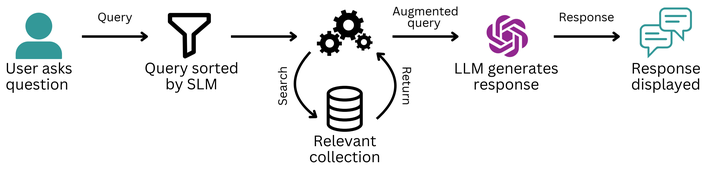
Team Leads

George Bowen · Drew Stinnett

AI INTEGRATION

Retrieval-Augmented Generation (RAG)

In our project, RAG boosts accuracy by supplying the model with applicable course information for each query.



Models Used

OpenAI LLMs

We generated context-aware responses for user queries through OpenAI's LLMs (3.5-turbo and 4o).

Phi-3-Mini

We routed queries to relevant databases using Microsoft's smaller language model, Phi-3-Mini.

Prompt Engineering & Fine Tuning

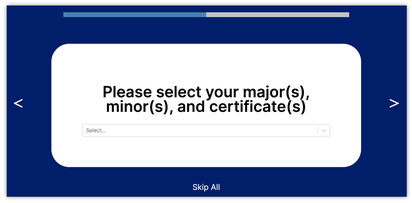
Using models out-of-the-box can lead to inconsistent and generic responses. By altering the prompt language and formatting and providing example responses, our model improved greatly.

Category averages across 102 responses for untuned vs. fine-tuned models

FRONTEND

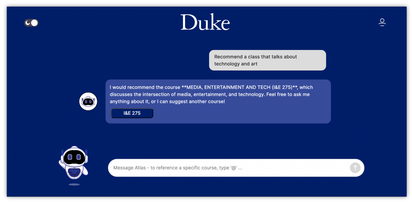
User Survey

We implemented an optional survey that collects student information, allowing chatbot responses to be personalized for each user.



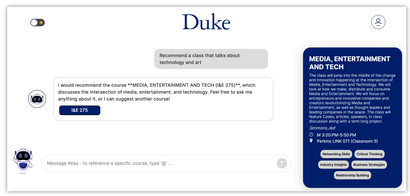
UI Updates

We overhauled the original UI to align more with Duke's branding and incorporate Atlas' logos and icons.



Additional Features

We introduced a quick-view popout and skill tags for each recommended course, along with the option to switch between light and dark modes for accessibility.



BACKEND

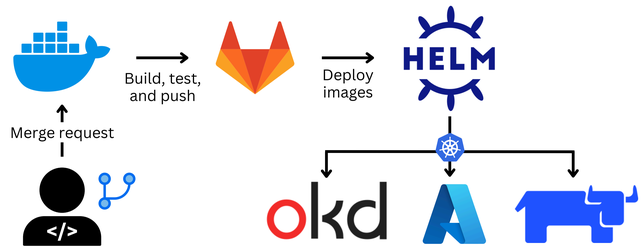
Authentication

We implemented the OpenID Connect (OIDC) authentication protocol, built on top of OAuth 2.0, delegating authentication logic to the server. Our process uses refresh and access tokens to ensure uninterrupted app usage for users.

Database and API Connections

Our backend utilizes MongoDB to store data as vector embeddings, with each collection maintaining a search index to minimize latency; Redis acts as a caching layer for contextual data to reduce the frequency of database queries. To manage seamless communications between our frontend, backend, and database, we designed and developed over ten API endpoints to handle CRUD operations for user and message management, incorporating integrated hashing mechanisms to ensure robust security of user data.

DEPLOYMENT



Set up a CI/CD pipeline that automatically builds and pushes images to the GitLab Container Registry, which are deployed using Helm onto various Kubernetes clusters.

TECH STACK

TS · mongoDB · Flask · React · redis · Python · Azure

LOOKING FORWARD

New Features

Given the variation in program structure, converting to a graph database would enable more flexible querying of prerequisite relationships. Additionally, we would like to implement a related-course view when Atlas recommends a class to enhance students' ability to discover new courses.

Enhanced Deployment

We plan to further refine our deployment process to improve our app's functionality and, through our ongoing connection with NCShare, we aim to make our tool available to more universities across North Carolina.

Acknowledgements

We would like to thank NCShare and the National Science Foundation for supporting our project via the following grants: 2201525 (NCShare Science DMZ) and 2201105 (NCShare Compute as a Service). Additionally, we would like to thank our stakeholders: Jon Reifschneider, Frank Blalark, Chris Derickson, Charley Kneifel, Ean High, and OIT.