A Large Language Model-Based Academic Advising Assistant for Engineering Technology Students

Dr. Murat Kuzlu, Old Dominion University

Murat Kuzlu (Senior Member, IEEE) is an Associate Professor in the Department of Engineering Technology at Old Dominion University in Norfolk, VA, USA. He earned his B.Sc., M.Sc., and Ph.D. degrees in Electronics and Communication Engineering in 2001, 2004, and 2010, respectively. Before joining ODU, he served as a Senior Researcher at the Scientific and Technological Research Council of Turkey (TUBITAK) and as a Research Assistant Professor at the Advanced Research Institute of Virginia Tech. His research interests include smart systems, trustworthy AI, explainable AI, agentic AI, and next-generation networks.

Dr. Vukica M. Jovanovic, Old Dominion University

Dr. Vukica Jovanovic is the Chair of the Engineering Technology Department and a Full Professor of Engineering Technology. She holds a Ph.D. in Mechanical Engineering Technology from Purdue University, with a focus on Digital Manufacturing. In addition, she earned a Magister degree (equivalent to Ph.D. candidacy in the United States) in Industrial Engineering and Engineering Management, specializing in Production Systems Design, as well as a Dipl.-Ing. (Master's) degree in Industrial Engineering and Engineering Management with a concentration in Mechatronics, Robotics, and Automation. Dr. Jovanovic brings extensive industry experience to academia. She worked as an Industrial Engineer in the manufacturing of aircraft wings and nacelles for commercial aircraft models ERJ-145 and ERJ-147 (Embraer) at Gamesa Aeronáutica in Vitoria/Bilbao, Spain. Prior to her university studies, she was a certified master electrician, completing apprenticeships in the power industry and an aluminum mill. Her professional foundation was further shaped through hands-on work in her family-owned manufacturing company, which specialized in sheet metal forming machines and metalworking services. Before pursuing an academic career, she served on the faculty at the University of Novi Sad in Serbia.

Dr. Katherine Smith, Old Dominion University

Katherine Smith received Ph.D. in Modeling and Simulation, M.S. in Applied and Computational Mathematics, and B.S. degrees in applied mathematics and mechanical engineering from Old Dominion University. Dr. Smith is an Assistant Professor in the Engineering Technology Department at Old Dominion University. Her research interests include data science, machine learning, and artificial intelligence, systems modeling and network analysis with applications in supply chain and cyber physical systems, data-driven implementation of industry 4.0 technologies, and serious game development.

Dr. Otilia Popescu, Old Dominion University

Dr. Otilia Popescu received the Engineering Diploma and M.S. degree from the Polytechnic Institute of Bucharest, Romania, and the PhD degree from Rutgers University, all in Electrical and Computer Engineering. Her research interests are in the general areas of communication systems, control theory, signal processing and engineering education. She is currently a Professor in the Department of Engineering Technology, at Old Dominion University in Norfolk, Virginia, and had served for seven years as the Program Director for the Electrical Engineering Technology Program. In the past she has worked for the University of Texas at Dallas, University of Texas at San Antonio, Rutgers University, and Politehnica University of Bucharest. She is a senior member of the IEEE.

Adel El-Shahat, Old Dominion University

Dr. Adel El-Shahat is currently an Assistant Professor & Director of Electric Machinery and Power Systems Lab, Department of Engineering Technology, Frank Batten College of Engineering & Technology, at Old Dominion University, USA. Before that, He was an Assistant Professor - Energy Technology, School of Engineering Technology at Purdue University, USA. He is the Founder and Director of Advanced Power Units and Renewable Distributed Energy Lab (A_PURDUE). He has more than 20 years of experience

in academia and industry. He has good experience in funding grant proposals, and He got some awards and recognitions due to his research work. He has good experience directing research for both graduate and undergraduate students for funded projects. He served as principal investigator and co-investigator for multiple externally and internally funded proposals. He holds full-time academic positions at Purdue University, Georgia Southern University, the University of Illinois at Chicago, Ohio State University, USA, and Suez University, Egypt. He received a B.Sc. in Electrical Engineering from Zagazig University, Egypt, in 1999. The M.Sc. in Electrical Engineering (Power and Machines) from Zagazig University, Egypt, in 2004, and the Ph.D. degree (Joint Supervision) from Zagazig University, Egypt, and The Ohio State University (OSU), Columbus, OH, USA, in 2011. His research focuses on Modeling, Design, Optimization, Simulation, Analysis, and Control of various aspects such as, Renewable Energy Systems; Smart Nano & Micro- Grids; Electric Mobility & Transportation Electrification; Wireless Charging of Electric Vehicles; Climate Change; Electric Vehicles; Special Purposes Electric Machines; Deep Learning Techniques; Distributed Generation Systems; Thermoelectric Generation; Special Power Electronics Converters; Power Systems; Energy Storage & Conservation; and Engineering Education. So far, He has 12 books, 6 book chapters, 1 patent, 85 journal papers, 78 conference proceedings papers, and 114 other publications with his collaborators and students related to his research interests. He has more than 20 years of experience in academia and industry. He has experience in funding grant proposals, and He got some awards and recognitions due to his research work. He has several funded external and internal grant proposals for more than USD 1,500,000.00. He has good experience directing research for both graduate and undergraduate students for funded projects. He holds full-time academic positions at Purdue University, Georgia Southern University, the University of Illinois at Chicago, Ohio State University, USA, and Suez University, Egypt, along with some full-time and part-time positions in Egyptian companies as an electrical engineer and consultant as a professional engineer. Additionally, He has distinguished professional training, and He is a Senior Member in the IEEE and IRED institutions, along with 21 professional memberships in other societies. Finally, He served as an editor for 5 textbooks, a reviewer for another 9 books, a journal reviewer/ member of editorial boards, and conference reviewer/ member of technical committees, and/ or session chair for 77 international venues. He is a guest editor and editor-in-chief for two international journals.

Ryan Cotton, Old Dominion University
Kayla Marie Seegers, Old Dominion University
William Austin Henderson, Old Dominion University

A Large Language Model-Based Academic Advising Assistant for Engineering Technology Students

Abstract

In recent years, Artificial Intelligence (AI)-based solutions, particularly Large Language Models (LLMs), have been applied to a variety of domains, such as energy, finance, transportation, healthcare, and education. Among these domains, education has become increasingly popular due to strong interest among educators and students. This study proposes an academic advising assistant system that uses LLMs to help Engineering Technology (ET) students plan their course load based on their educational history, departmental course offerings, and personal constraints, such as their preferred semester course load. The proposed LLM-based academic advising assistant system maintains a database of students' course histories and upcoming course availability to provide personalized recommendations for which courses to take based on user input. It also explains course dependencies and adapts its suggestions according to user constraints, such as course difficulty (inferred from grade distributions) and maximum credit limits per semester.

Keywords

Artificial Intelligence, Large Language Models, Academic Advising, Higher Education

Introduction

In recent years, academic advising has become a critical task in higher education due to the increasing number of degree programs and students, leading advising offices to struggle to meet the demand for advising appointments. Shortages of advisor availability and time per student have left many undergraduates without clear or timely information about degree requirements, prerequisite ordering, and course-by-course plans for each semester.

As a result, many students take unnecessary courses or delay necessary ones because they lack the opportunity to clarify their course plans. National enrollment and completion statistics also indicate that unclear degree pathways and advising bottlenecks lead to longer time-to-degree for a majority of college students[1,2]. In a modern university environment with increasingly rigorous degree programs, inadequate advising can lead to significant, lasting academic and financial consequences for students. Fortunately, Large Language Models (LLMs) have improved dramatically in recent years and can now follow complex instructions, retain context in long-form

interactions, and generate highly coherent responses. State-of-the-art language models such as GPT, Claude, and LLaMA have demonstrated remarkable success in advanced reasoning, multi-turn dialogue, and natural language explanations for different types of context[3,4,5]. While traditional academic advising systems and tools are largely limited to static degree audits, static menus, and hard-coded decision trees, LLMs' reasoning and explainability capabilities set them apart. Academic advising is, by nature, a rule-intensive task, and the structured, logical thought processes for prerequisites, degree requirements, and other planning details have not been easy to translate into a language system. However, recent successes by LLMs in explaining complex structured logic in natural language for academic degrees show that the gap is closing.

Current academic advising systems offer limited personalization, whereas LLMs have demonstrated a range of powerful capabilities. Advisors commonly rely on degree audit systems to indicate which requirements have been satisfied and which remain outstanding. There are limited planning features for what-if analyses, such as "what if I take that class later," "I am transferring these credits, what do I still need," or "are there different ways to order the remaining classes?" Advising systems also typically do not explain why requirements must be satisfied or which credits to take. If an advisor needs to convey this information, it is typically provided in verbose written explanations, and the student receives no immediate feedback. As a student progresses, their courses taken, courses in progress, graduation dates, and goals change. Students may repeat a course, switch to part-time or full-time status, retake classes, or change their goals. These natural changes to students' educational trajectories are only partially supported by existing advising tools. According to various academic research and industry reports, students value procedural guidance and explanations over raw lists of requirements[6]. The study[7] shows that LLMs are useful tools for free-form reasoning, open-ended logic, and natural language dialogue. However, LLMs are not yet sufficiently accurate to be used safely and reliably on their own for high-stakes planning or summarization tasks in a rule-free environment. The model in the study generated false information, made incorrect policy assumptions, or provided overly high-level or under-specified guidance when used for advisory tasks without sufficient structure or filtering. Due to these results, research by the same team has shown that a multi-stage pipeline that tightly couples an LLM with rule-based logic, verified data, and retrieval systems can reliably address the limitations and safety risks of an LLM-only system[8,9,10]. In this system architecture, the LLM layer serves as an explainable, reasoning wrapper around deterministic logic, handling state tracking and requirement enforcement in an advisory dialogue. This closely mirrors the desired user-facing properties of an advising system.

This study proposes an academic advising assistant system that uses LLMs to help Engineering Technology (ET) students plan their course load based on their educational history, departmental course offerings, and personal constraints. The system stores students' academic histories and upcoming course offerings and generates personalized course recommendations based on their input. Students at universities today expect instant, on-demand access to knowledge that the current manual advising process cannot provide. Students can also save time by using an automated answering tool for repetitive questions that are currently answered manually. Universities are key stakeholders in the adoption and deployment of advising tools and require a scalable, cost-effective solution. Considering the needs of students, advisors, and universities, the goal of this study was to develop an LLM-based advising assistant that can quickly and accurately assist students and allow them to explore degree requirements with greater confidence.

Academic Advising Process

The academic advising process plays a critical role in students' academic journey and success. Most major universities require advising to be conducted in one-on-one meetings between the student and the student's advisor before the start of early registration. Students meet with their academic advisor at least twice a year, during the pre-registration period in the Fall and Spring semesters. During these sessions, the student's academic records are updated in an Excel spreadsheet to align with the University's digital records; grades are monitored; students are advised of potential GPA and course-plan issues; and the student's career goals are reviewed. During the pre-registration and registration periods, advisors review student records to ensure that all courses are taken in the proper sequence and that all prerequisites are met. The advisor verifies curriculum completion and guides future scheduling. Upon completion of the meeting, the advisor block is removed from the student's account, and the student can then register. The student can then also be encouraged to visit their advisor for guidance on academic matters and career opportunities.

Student classifications and academic statuses may require additional requirements. All first-year students will meet with a professional advisor during orientation and throughout their first year, in addition to the two regular visits. Freshman instructors are required to submit midterm grade reports to students so students can discuss any academic progress issues with the instructors if needed. After freshman year, non-freshmen's academic performance is assessed by advisors during academic advising appointments. The program director of their corresponding major advises all on-campus non-first-year students. The director assists students in selecting courses and advises them on graduation requirements, senior technical options, advanced studies, and professional opportunities. Faculty advisors also strictly enforce prerequisites, which is adequate for most students making satisfactory progress. Lastly, if the student is graduating in the coming semester, the program director reviews the student's entire record and verifies that all degree requirements are met by the end of the coming semester. The program director then verifies that DegreeWorks correctly reflects the student's academic record. Upon graduation, the Registrar's Office reviews DegreeWorks to verify that students have met degree requirements and then awards their degree.

In addition to the different statuses that can affect advising, whether the student is on campus or in a distance-learning program will also affect the process. Distance-learning students are advised by professional advisors at their respective distance-learning sites. If the student is at a location without a professional advisor, they must contact the on-campus distance-learning office. The program director is consulted when questions arise during the advising process. During the academic year, all students' academic records are kept electronically in the University's Banner system. Faculty advisors have access to the Banner system, and students can review their electronic academic records online. All course prerequisites are published in the University's Undergraduate Catalog and on the website. Additionally, the system automatically checks prerequisite completion during web registration, and students who do not meet any prerequisites will be blocked from registering for the course.

Proposed LLM-Based Academic Advising Workflow and Main Components

LLM-Based Academic Advising Workflow

Figure 1 illustrates the proposed LLM-based academic advising workflow, consisting of three main components.

- Student Inputs Data and Goals,
- A System Process Request,
- An AI-generated Plan.

In this workflow, first, students enter their information, including completed courses, preferred workload (number of courses or credits), and constraints (e.g., graduation timeline, availability). Then, the LLM-based system processes the request along with the course database and degree requirements. Finally, the system generates a personalized plan for each student that includes a multi-term course plan, course selections, graduation goals, and constraints.

Main Components

Figure 2 illustrates the three main parts of the system, i.e., Database, OpenAI Large Language Model (LLM), and Backend Application Logic, in addition to the advisor and student ones. Each component is explained briefly as follows:

- Database: It is the primary memory bank of the system, stored in a PostgreSQL database. The database holds all the information regarding users, educational programs run by the institution, course sections, and the classes needed to fulfill these sections. The database obtains all user information from the student User Interface (UI), where students can apply schedule constraints and mark all completed courses. The course structures themselves are added to the database by the advisor UI. The Advisor UI can also add educational programs to the database for the student to choose from, along with all available classes and times.
- OpenAI LLM: This component serves as the core reasoning engine of the system, utilizing the gpt-4o-mini. It uses the database to generate the requested schedules. It quickly processes the classes the user needs to complete their program, the class availability, and any schedule constraints requested by the student to create an all-inclusive schedule for the remaining semesters. This portion of the system performs the main processing for the program.
- Backend Application Logic: The backend application logic is the coding logic that connects the Student and Advisor UIs, OpenAI, and the Database, and makes the processes work. It

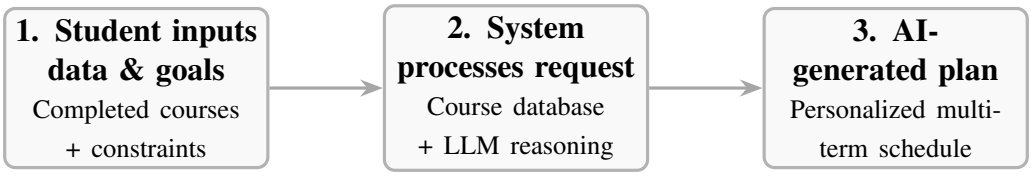

Figure 1: Overview of the LLM-driven academic advising workflow

serves as the interface between the system's interactive components. It is currently divided into two parts: the authorization and session manager, and the request orchestrator. The authorization and session manager are components of the code that allow the student to log in by verifying the saved username and password combinations. If the information entered matches the database, the student will be able to log in to the student UI. The request orchestrator is the portion of the code that interfaces between the database and OpenAI, ensuring that the AI has all the information it needs to create a schedule. It also sends the completed schedule to the Student UI for review. A third portion of this code is planned, but is currently not operational.

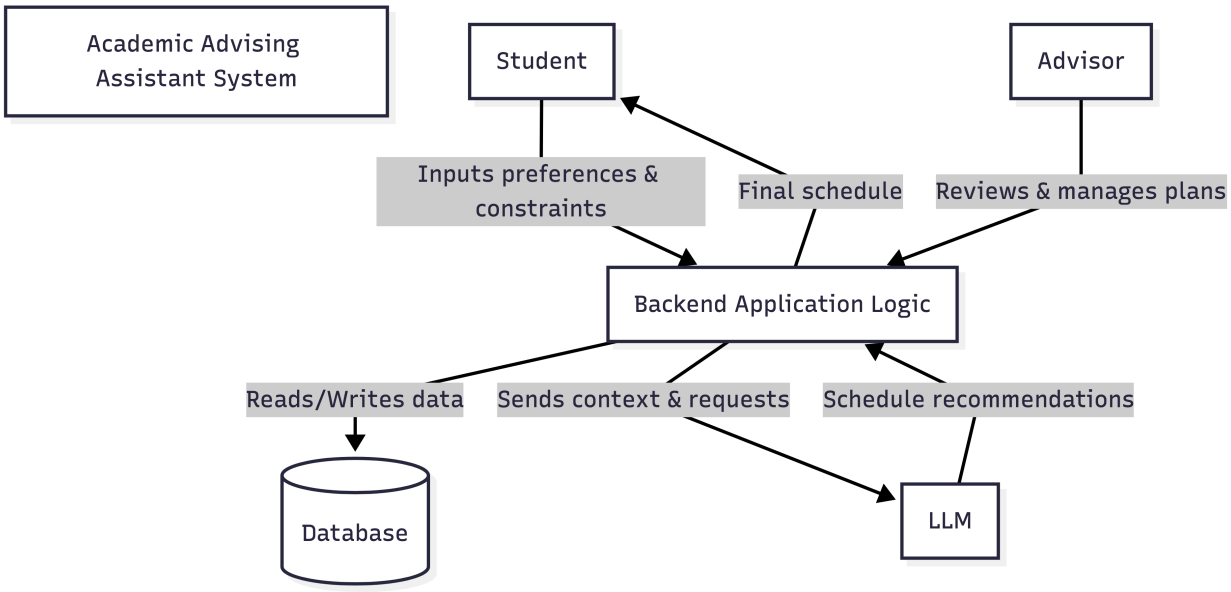

Figure 2: Interaction flow of the main components of the LLM-based academic advising assistant.

System Implementation

This section describes the implementation of the LLM-based academic advising assistant system, with the overall system architecture shown in Figure 3. The architecture encompasses a variety of modular components. Each component is briefly explained below.

User Interface Layer

The system consists of two primary interfaces, i.e., Student and Advisor User Interfaces (UIs), implemented with Streamlit, a Python-based web framework.

- The Student UI provides:
  - login fields for username and password,
  - forms to enter degree-planning inputs and constraints (e.g., "no more than four courses per term"),

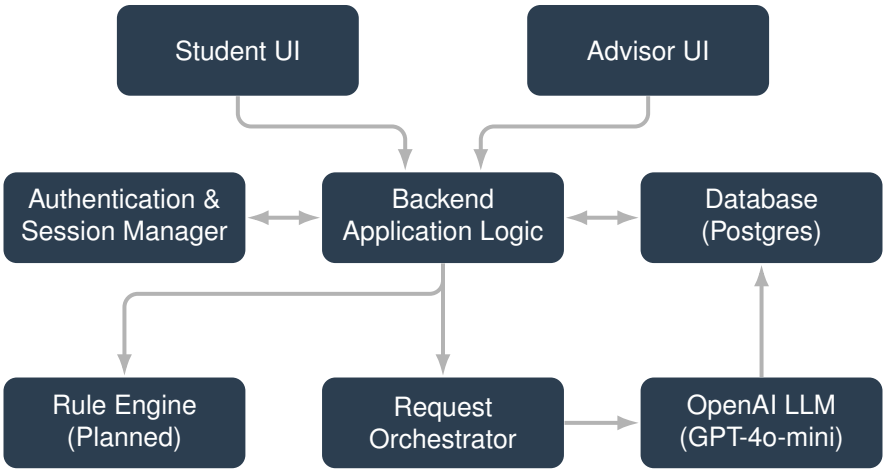

Figure 3: Multi-stage pipeline for LLM-based academic advising

  - views for reviewing AI-generated multi-term schedules and associated rationale.

- The Advisor UI provides:
  - read access to student plans and constraints,
  - access to basic student data stored in the database,
  - interfaces intended for future course catalog adding, editing, and program configuration.

Navigation between the pages, i.e., login, the advisor, and the student pages, is handled via Streamlit's page routing, based on the user's role stored in the session state. Figure 4, 5, and 6 illustrate the example UIs for students and advisors, respectively.

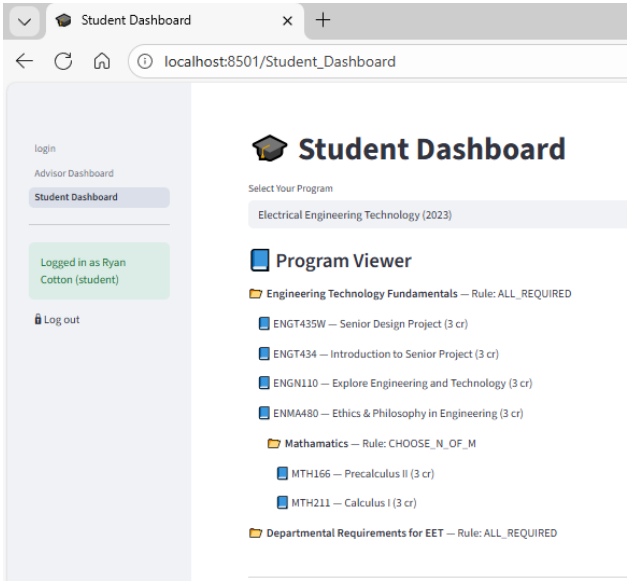

Figure 4: View of student user interface: Program Viewer

Figure 5: View of student user interface: Completed Courses

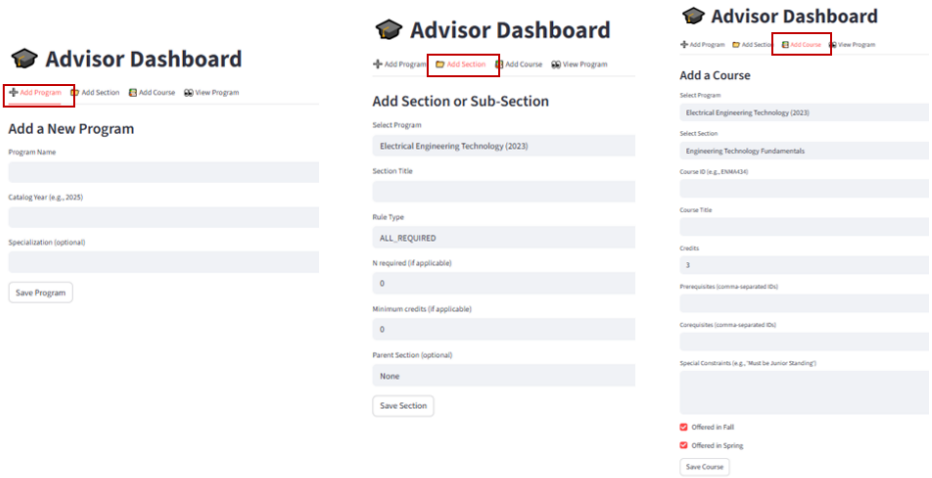

Figure 6: View of advisor user interface

Authentication and Session Management

The authentication process in the system is performed through a database-backed, i.e., username-password check. The current system stores passwords in plain text; however, the final platform will support secure hashing, and session management is handled entirely by Streamlit. When a user intends to log into the platform, the system executes the following steps:

- opens a connection to the Supabase-hosted PostgreSQL instance using psycopg2,
- queries the users table for a record,
- if a match is found, creates an in-memory user object containing the username, full name, and role (i.e., student or advisor),
- stores this object, which is then used to gate access to pages.

Data Storage layer

Persistent data is stored in a PostgreSQL database hosted by Supabase, which appears in the diagram as the central database component. The schema currently includes, at a minimum:

- users: usernames, full names, passwords, and roles,
- programs: academic programs and catalog years,
- sections: hierarchical program sections (e.g., core, electives, concentration blocks),
- courses: course IDs, titles, credits, term offerings, prerequisites, and special constraints text,
- student_courses_completed: mapping between students and completed courses,
- student_constraints: normalized constraint "chips" derived from student free-text input,
- (planned) recommended_schedules: storage for generated schedules for later review.

The data storage layer interacts with all system components. The first step is authentication, which verifies credentials. Then, both the advisor and student user interfaces receive data related to the student's academic history and program requirements, while the backend logic reads and writes constraints and schedules from and to the database.

Backend Application Logic

The backend application logic is implemented with the following responsibilities:

- Data Aggregation:
  - fetches course catalog data (programs, sections, courses, offerings, prerequisites, special constraints) from the database,
  - pulls the student's academic history, along with constraints.
- Constraint Handling:
  - accepts constraints, e.g., no more than 5 courses per semester,

    - passes them through an LLM call to convert them into structured data.
- Prompt Construction:
    - combines all information, including catalog data, completed courses, and constraints, into a structured prompt for the LLM.
    - enforces catalog consistency by including only known course IDs, credits, and term offerings in the prompt.
- Result Integration:
    - parses the LLM's response,
    - provides course IDs with titles and credits against the catalog,
    - displays the schedule,
    - writes the LLM-generated plan back to the database.

Request Orchestrator

The request orchestrator is a subcomponent of the backend that focuses on LLM interaction. Its primary responsibilities are:

- receiving the context (catalog, constraints, student history) from the backend,
- building a prompt defining course IDs, maximum credits per semester, and constraints,
- calling the OpenAI API using GPT-4o-mini and capturing the raw response,
- performing minimal validation, i.e., e.g., confirming the response is valid JSON,
- providing the cleaned result to the backend logic.

OpenAI LLM (GPT-4o-mini)

OpenAI's GPT-4o-mini is used as the AI reasoning layer, accessed via the OpenAI Python client. At this stage of the study, the LLM performs nearly all scheduling intelligence, including:

- sequencing courses based on prerequisite descriptions,
- honoring student constraints where possible (e.g., maximum course load, avoiding specific combinations),
- choosing term offerings based on encoded Fall/Spring/Summer availability,
- balancing workload across terms while attempting to minimize time to completion,
- explaining the rationale for the suggested plan.

Rule Engine (Planned Component)

The architecture includes a rule engine box connected to both the backend logic and the orchestrator. This component is not yet implemented in the current system.

The planned responsibilities for the rule engine are:

- encoding prerequisites and co-requisites in a machine-readable form, for example, boolean expressions over course IDs,
- enforcing maximum and minimum credits per term,
- performing hard checks on term availability,
- rejecting or flagging LLM-generated schedules if they violate s institutional policies.

In the current phase, these responsibilities are handled by the LLM through prompt engineering. The rule engine is intended to gradually take over all non-negotiable institutional rules, allowing the LLM to focus on softer preferences or constraints that are not hard-coded into the rule engine validation.

Demonstration

The system is developed on the Streamlit Community Cloud to allow students and advisors to access the application via the dashboard without installing any software. The deployment pipeline is as follows:

1. Source code and configuration are stored in a GitHub repository.
2. A requirements.txt file specifies runtime dependencies, including streamlit, psycopg2-binary, supabase, and openai.
3. A runtime.txt file pins the Python version to match dependencies known to build successfully on Streamlit Cloud.
4. Sensitive configuration values, such as the OpenAI API key, Supabase URL, and PostgreSQL connection string, are stored in Streamlit's secrets management (st.secrets) and are not committed to source control.
5. Streamlit Cloud automatically creates an isolated environment and launches the front-end login page.

If the submitted credentials match the database records, the student is redirected to the Student UI. It will display two interactive sections to the student. The first is the program viewer, which shows the program the student is taking and all of the possible classes that can fill each part of the program. This program can be chosen from the list available to the student. This list includes all available programs and the year each program started. The chosen year ensures that the database refers to the correct catalog year, so any changes made to the program after that year will not affect the student. The second section is "Mark Completed Courses." This section allows students to mark which classes they have already taken, so that they can be removed from any future class schedules.

As the student scrolls down, they will see the AI Planner and the starting term for which the schedule should be planned, see Figure 7. The AI planner allows the student to interact directly with the AI and request specific constraints for their schedule to better meet their lifestyle needs. These constraints can range from classes the student would like to avoid to the number of credit hours the student would like to take per semester.

Once ”Plan my Schedule” is selected, the AI creates the multi-term plan. An example of this is found in Figure 8. This will show the recommended classes for every remaining semester and the rationale the AI used to produce this schedule. Future versions of this area will also list any classes that could not be included due to constraints, but are needed to complete the program.

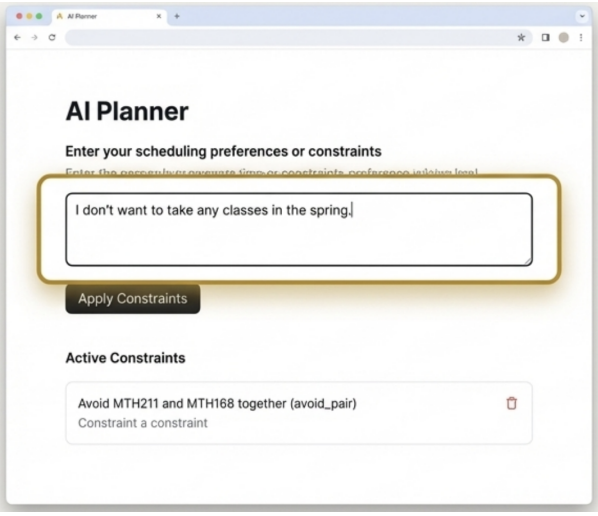

Figure 7: LLM-based AI planner

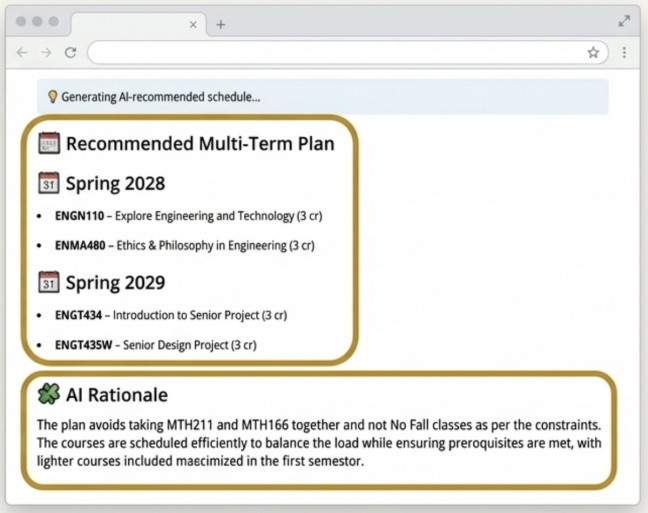

Figure 8: AI-generated multi-term plan

Conclusion

Many higher education institutions have observed an increasing demand for academic advising services in degree planning. However, many existing advising tools struggle to support complex, multi-step reasoning. As degree requirements become more prescriptive and student populations grow, students often have to wade through dense, opaque degree audits to understand their progress. This can lead students to take unnecessary courses, miss prerequisite sequences, or take longer to complete their degree.

This study focused on developing an LLM-based advising assistant to support human academic advisors while ensuring that requirements are met. Emerging large language models (LLMs) are increasingly demonstrating mastery of contextual understanding, explainability, and multi-step reasoning, presenting an opportunity to improve academic advising systems. This study leverages the more general capabilities of LLMs, such as understanding informal student language, while constraining outputs with deterministic policies to address known weaknesses. The proposed LLM-based advising assistant system is intended to support the current advising process at universities, not replace human academic advisors. The proposed system can significantly improve the student experience and the academic decision-making process.

References

[1] National Student Clearinghouse Research Center, "Current term enrollment estimates," https://nscresearchcenter.org, 2023, accessed: 2025-01-01.

[2] U.S. Department of Education, "College completion and student success," https://www.ed.gov, 2022, accessed: 2025-01-01.

[3] OpenAI, "Gpt-4 technical report," OpenAI, Tech. Rep., 2023. [Online]. Available: https://arxiv.org/abs/2303.08774

[4] Anthropic, "Claude 3 model card," https://www.anthropic.com, 2024, accessed: 2025-01-01.

[5] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[6] EDUCAUSE, "The future of academic advising: Technology-enabled student support," https://www.educause.edu, 2022, accessed: 2025-01-01.

[7] J. Sun, C. Zheng, E. Xie, Z. Liu, R. Chu, J. Qiu, J. Xu, M. Ding, H. Li, M. Geng *et al.*, "A survey of reasoning with foundation models: Concepts, methodologies, and outlook," *ACM Computing Surveys*, vol. 57, no. 11, pp. 1–43, 2025.

[8] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022.

[9] S. Yao *et al.*, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2023.

[10] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.