End-of-Semester Progress Report

Intelligent Academic Advisory System (IAAS)

Team Members:

Feras Alkahtani ID 230313
Abdulaziz Albaz ID 23
Mohammed Alhajri ID 23

Supervisor:

Prof. Anis Koubaa

SE 495 - Capstone I

Fall 2025

30/11/2025

1. Introduction and Background

1.1 Problem Statement

Academic Advising plays a very important role in a student's career, but still most universities still rely on physical and manual methods that cant keep up with the growing number of enrollments, as a result, many students cant get access to vital and important academic advising that can shift a students studying career

Students sometimes get mixed advice or even face delays in graduating. Usually, this happens because there just aren't enough advisors to go around, especially during busy registration times. Sometimes students have to wait a long time, and the advice they get can depend a lot on which advisor they talk to. In our experience, existing online systems don't really solve these problems. They mostly help with signing up for classes, but they don't offer smart academic advice or help with planning and checking if a student's choices actually make sense.

Without something smarter in place, we noticed students often feel lost when it comes to planning their courses, and advisors get bogged down doing the same simple tasks over and over, tasks that could probably be automated.

There is a need for an automated system that can check prerequisites, monitor academic progress, help students create personalized study plans, and provide accessible guidance.This will help students decide more confidently and ease the advisors of repetitive tasks

1.2 Motivation and Rationale

IAAS was created because academic programs are getting more complicated, and students expect clear, efficient, and personalized advising. Students can get easily confused about their decisions with the lack of resources and time

At first, traditional advising was thought to be enough, but with the increase of students and courses, it became clear that short fast meetings, different opinions, and conflict of time can slow students down. This issue is clear and seen by many universities, but automated tools have not been really implemented. Digital advising can give students quick and accurate answers, but most implemented tools don't check prerequisites, so even with a semi automated system, students would still need to figure it out on their own

IAAS addresses these gaps by integrating intelligent automation into its advisory services. Features such as automated prerequisite validation, degree progress visualization, semester planning algorithms, and an AI-powered Chat Advisor provide students with accurate, consistent

guidance at any time, and with a structured rule-based logic it will ensure precise interpretation of academic policies, reducing errors and promoting fairness.

IAAS automates repetitive tasks,such as checking course eligibility, validating combinations and getting rid of conflicts, and interpreting program requirements, and reducing administrative tasks for advisors. This enables greater focus on individualized support, improved operational efficiency, and enhanced student experience.

1.3 Objectives

IAAS is designed to give students a smart, easy-to-use platform that makes planning and academic advising simpler and more reliable. The main goals cover what the system should do, how it should work technically, and how it helps the university overall.

1. Core Functional Objectives

Build a chat-based AI advisor that understands students' questions about things like degree requirements, prerequisites, course options, and academic rules, and gives them quick, helpful answers.

Create an automated semester planner that suggests study plans for students, taking into account the courses they've finished, their current status, and all the requirements for their program.

Add a dashboard that shows which courses are done, which are ongoing, and which are left, so students can see exactly where they stand on their path to graduation.

Set up a rule-based checker that checks if students meet all prerequisites before signing up for classes, which will help them avoid scheduling errors.

Design a recommendation tool that suggests the best courses each semester, based on whats available for students, how much they want to take on, and their overall degree plan.

Provide an admin portal where academic staff can easily upload, edit, and manage course data and program details.

2. Technical and Engineering Objectives

Build the system with a modular, layered design, so that everything is separate. This makes it easier to keep things running smoothly as the system grows.

Include a structured rule checker that can handle prerequisite rules, and course pathways, using reliable data.

Make sure the system is easy to use, with a simple interface so students can find what they need without much training.

Add secure login features so students and admins have different access levels, keeping all academic data safe.

3. User Experience and Institutional Objectives

To reduce reliance on physical advisors by providing students with accurate, real-time academic support.

To Improve advising efficiency by automating repetitive tasks.

To Support the university's digital goals by offering a scalable advising tool.

To Enhance student satisfaction and success by empowering students to make informed academic decisions.

1.4 Scope and Constraints

The scope of IAAS defines the boundaries of the Capstone I project, specifying deliverables and limitations. IAAS is intended as a platform that helps students understand degree requirements, plan semesters, track progress, and receive intelligent guidance. In Capstone I, the focus is on requirements analysis.

Project Scope:

The IAAS project includes the following major components:

1. Student-Facing Features

Interactive Chat Advisor:
A text-based intelligent agent designed to answer academic questions related to prerequisites, program requirements, registration rules, degree progress, and course information.

Degree Progress Dashboard:
A visual Diagram of the student's completed, in-progress, and remaining courses, based on their degree plan.

Automated Semester Planner:
A planning tool that suggests possible semester schedules by analyzing prerequisites, remaining requirements, and course requisites.

Course Recommendation Suggestions:
A guided recommendation system that highlights possible courses for upcoming terms based on the student's academic history and eligibility.

2. Admin and Backend Features

Dataset Management Interface:
Tools that allow administrators to upload course data, curriculum structures, prerequisite rules, and program requirements.

Rule-Based Prerequisite Engine:
A backend module responsible for validating course eligibility and program logic.

User Authentication System:
Support for secure login, distinguishing student users from administrative users.

3. Engineering and Development Scope

Creation of the system’s requirements specification, including functional and non-functional requirements.

Development of UML diagrams, including use case diagrams, ERD, sequence diagrams, activity diagrams, and component diagrams.

Establishment of a modular layered architecture, separating frontend, backend, business logic, and data layers.
Constraints

The IAAS project is subject to several constraints that influence development progress during Capstone I:

1. No Direct Integration With University Systems

The system does not connect to Alfaisal University’s Banner or internal databases. All data must be manually collected, cleaned, and uploaded, limiting real-time academic interactions.

2. Limited Dataset Availability

Course catalogs, prerequisite maps, and program structures must be manually prepared. Incomplete or inconsistent datasets may limit the system's accuracy during the early stages.

3. Time Constraints

Capstone I focuses primarily on design, analysis.. Full system development is planned for Capstone II.

4. Specification-Level Functionality

Due to time and resource limitations, only core system components such as the planner logic, chatbot prototype, and prerequisite engine will be planned and implemented during Capstone II.

5. No Machine Learning in Capstone I

IAAS primarily relies on rule-based logic due to time and data constraints.

6. Team Size and Division of Responsibilities

With a small development team, multiple tasks such as dataset creation, UI design, backend modeling, and documentation must be distributed carefully.

1.5 Project Overview

(IAAS) is a platform designed to automate a students course enrollment process. It brings together automated reasoning, guidance tools, and data to help students understand their degrees, plan their semesters, and avoid common errors that can be overlooked by physical advisors, IAAS functions as an intelligent system, that brings together different modules to support students across all years.

IAAS is built as a layered architecture that separates the system into several components, Presentation Layer, which contains the interfaces both students and admins see, the Application Layer, Which handles requests between frontend and backend to ensure smoothness and efficient execution. Logic Layer, which contains prerequisite validation, rule interpretation, and planning algorithms, The data layer, stores the datasets that contain course info, prerequisite structures, and degree plans.

One of the central features of IAAS is the AI-powered Chat Advisor, which acts as an interactive interface, enabling students to ask questions about courses, program requirements, graduation timelines, and general academic concerns. Through rule-based logic and structured datasets, the Chat Advisor provides instant, consistent, and accurate responses, which eliminates the need for students to rely on advisor availability and provides 24/7 academic support.

Another core component is the Degree Progress Dashboard, which represents a student's academic status. The dashboard allows students to view completed, ongoing, and remaining courses in their degree plan. This visual clarity helps students understand how far they have come and what requirements remain. It also serves as a foundation for generating personalized academic recommendations.

The Automated Semester Planner is a tool that helps students build semester schedules. The planner checks prerequisite conditions, remaining degree requirements, course corequisites, and typical academic sequencing patterns to generate course combinations. This ensures that students select courses in an order that aligns with academic rules based on the semester and long-term success.

Additionally, IAAS includes a Rule-Based Prerequisite Checker that checks the program's academic structure and verifies whether students meet the required conditions before enrolling in any course. This feature directly addresses one of the most common student mistakes, enrolling in courses without the proper background, which leads to administrative withdrawals, academic delays, or unnecessary difficulties.

The Admin Portal allows authorized academic staff to upload, update, and manage curriculum data, prerequisite rules, and degree structures. This ensures the system remains accurate and up to date with program changes. The portal provides tools for maintaining academic rules without requiring programming knowledge.

During Capstone I, the IAAS project focused entirely on the foundational planning and design stages. Our work during this phase involved identifying system requirements, defining the architecture, outlining the user interface layout, and detailing how the core components should function. No actual implementation or coding of these components has been completed at this stage. Instead, Capstone I served to establish a clear blueprint for how these modules will be built in Capstone II.

Overall, the IAAS project aims to combine automation, intelligent academic reasoning, and user-centered design to create a system that supports student success, reduces advisor workloads, and contributes to the university's digital transformation goals. By integrating multiple tools into a single unified platform, IAAS offers a structured, accurate, and efficient approach to academic planning and advising—ultimately helping students progress through their degrees with clarity, confidence, and reduced risk of academic setbacks.

2. Related Work

Academic advising systems have evolved in recent years as universities seek to improve student support, reduce administrative challenges, and make degree planning easier. Several digital degree-auditing and planning tools are available on the market, such as Ellucian

DegreeWorks, Stellic, uAchieve, and various advising platforms. These tools provide structured support for tracking academic progress, but they differ in automation capabilities, user interactivity, flexibility, and reliance on human advisors. This section examines existing solutions, identifies their strengths and limitations, and highlights the gaps that the Intelligent Academic Advisory System (IAAS) aims to address.

2. Related Work

2.1 Existing Solutions

2.1.1 Ellucian DegreeWorks

DegreeWorks is one of the most popular degree-audit systems used by universities worldwide. It provides automated degree audits, prerequisite tracking, and structured dashboards that show student progress. DegreeWorks integrates tightly with university student information systems such as Banner.

Strengths:

Strong institutional integration
Clear visualization of degree progress
Automated “degree audit” reports
Standardized curriculum data management

Limitations:

Not interactive
No personalized semester planning
No AI-based recommendations
Dependent on full institutional integration

2.1.2 Stellic

Stellic is a modern cloud-based academic planning platform focused on personalization and user experience. It allows students to drag-and-drop courses into future semesters, explore degree pathways, and visualize their graduation timeline.

Strengths:

Excellent user interface and interaction design

Strong semester planning features
Predictive pathways for graduation timelines

Limitations:

No AI-driven chat interface
No flexible rule engine, meaning prerequisite structures rely on strict datasets.
Expensive
Limited transparency in advising logic

2.1.3 uAchieve

uAchieve is a well-established degree audit and student planning suite used in many universities. It focuses on detailed degree requirements, progress mapping, and audit reporting.

Strengths:

High audit accuracy
Supports transfer credit evaluation
Program maps are detailed and customizable.

Limitations:
Not student friendly
Limited intelligent advising capabilities
No AI integration

2.1.4 University-Developed Chatbots

Some universities have implemented basic advising chatbots, examples include IT support bots, admissions inquiry bots, or basic FAQ bots.

Strengths:

Provide quick responses to common questions.
Reduce advisor workload for routine inquiries.

Limitations:

Not integrated with degree audits or prerequisite logic
Responses are often generic and not personalized.

Unable to interpret individual student progress
Cannot generate or validate semester plans

2.2 Comparative Analysis

Feature	DegreeWorks	Stellic	uAchieve	IAAS
Degree Progress Tracking	Included	Included	Included	Included
Semester Planning	Limited	Strong	Limited	Automated Planner
AI Chat Advisor	None	None	None	Included
Prerequisite Checking	Included	Included	Included	Included
Course Recommendations	None	Basic	None	Included
User Friendliness	Medium	High	Low	High
Primary Focus	Degree Auditing	Planning	Backend auditing	Intelligent Advising

2.3 Summary of Gaps in Existing Systems

Across all reviewed solutions, three major gaps consistently appear:

1. Lack of Intelligent Reasoning

Most systems check degree requirements, but they do not provide real-time academic advising or answer complex student queries. IAAS fills this gap with a chat-based advising interface, powered by structured rules.

2. Limited Personalization

Existing tools rely heavily on static curriculum structures. IAAS introduces dynamic planning, adjusting course suggestions based on, prerequisites, current progress, program variations and student load preferences

3. High Integration Dependence

Major commercial systems require deep integration with SIS (student information system). IAAS is designed to operate independently, allowing, rapid prototyping, controlled data testing, minimal deployment overhead

2.4 Contribution of IAAS Compared to Prior Work

IAAS differentiates itself through the following:

1. Unified Advising

IAAS integrates degree progress tracking, planning, and a chat-based advising assistant into a single platform.

2. Rule-Based Academic Intelligence

Unlike traditional advising tools, IAAS interprets academic rules, enabling prerequisite checks, course eligibility checks, and validation of student choices

3. Academic Support

IAAS includes an interactive advising chatbot capable of real-time responses, which existing commercial tools do not provide.

4. Modular Architecture

IAAS is built using a layered architecture, which is easy to expand, update, and migrate into a full-scale deployment.

2.5 Conclusion of Related Work

It turns out that existing advising systems are effective in certain cases, but often lack intelligent reasoning, personalization, and flexibility. IAAS is designed to fill these gaps by offering an integrated, intelligent, academic advising tool with advanced planning capabilities.

3. System Requirements Specification (SRS Summary)

3.1 Overview of Proposed System

IAAS was made with the idea in mind to support students and help them understand the requirements they need to meet and to make decisions that are right throughout their career. The system joins several tools to help ease the academic life of a student, those tools would track their progress, check if they are eligible for a course, explore any other remaining requirements the student has not yet met, and plan early on for upcoming semesters

IAAS serves two main roles: students and administrators. Students interact with the system as they see fit and needed, either they view their progress, or check the eligibility for a course, or chat with the built in chat advisor. Administrators on the other hand, they are responsible for managing the datasets through an interface deigned for them, they can update courses, prerequisites and program requirements to make sure the system reflects current academic information.

The system focuses on delivering accurate and accessible advising support by organizing academic data in a structured way and presenting it through tools designed to help students navigate their degree requirements. This overview outlines the core purpose, users, and expected capabilities of IAAS at a high level, serving as a foundation for the functional and non-functional requirements described in the rest of this section.

3.2 Functional Requirements Summary

ID	Description	Priority	Status
FR1	The system shall load and process pre-generated datasets of 500 students and 200 courses with their information	High	Not implemented
FR2	The system shall validate dataset	High	Not implemented

	integrity before storage		
FR3	The system shall display interactive visualization of student degree progress with prerequisite chains	High	Not implemented
FR4	The system shall allow users to search and view course information and prerequisites	High	Not implemented
FR5	The system shall provide LLM-based chatbot using OpenAI GPT-4o-mini API	High	Not implemented
FR6	The system shall provide authenticated administrator interface to manage and refresh datasets	Medium	Not implemented
FR7	The system shall log all chatbot interactions and recommendation requests with timestamps	Medium	Not implemented
FR8	The system shall generate PDF summaries of course plans and progress reports	Low	Not implemented
FR9	The system shall automatically switch to deterministic rule-based recommendations when OpenAI API monthly quota ($20) reaches 90% consumption	High	Not implemented
FR10	The system shall	High	Not implemented

	validate all chatbot recommendations against prerequisite rules before presentation		

3.3 Non-Functional Requirements Summary

The non-functional requirements for the Intelligent Academic Advisory System (IAAS) describe the quality expectations that guide how the system should operate and how users should experience it, they play an important role in shaping the system's performance, reliability, ease of use, and long-term sustainability. Identifying these attributes this early helps ensure that the final implementation will meet both technical standards and user expectations.

One of the most important non-functional requirements for IAAS is performance. The system is expected to respond to common advising tasks, such as checking course eligibility, retrieving degree progress, or answering questions within a short time frame, ideally under two seconds. Students will rely on IAAS for quick academic decisions, so delays or slow responses could negatively affect the usability of the platform. In addition to response speed, the system should be able to support many users accessing it at the same time, especially during peak advising or registration periods.

Reliability is another major requirement. IAAS needs to consistently return accurate information about prerequisites, degree requirements, and course availability. Because mistakes in advising can heavily influence academic progress, the system must handle rule evaluations and data retrieval with high accuracy. Any errors in the dataset or rule definitions should be caught and flagged for administrators before they affect students. The platform should also be resilient enough to recover from unexpected failures or invalid data inputs without crashing or producing misleading results.

The system must also take care of security and privacy, since it may eventually work with sensitive academic information such as student records, completed courses, and program progress. While the Capstone I prototype uses test data, the full system should enforce user authentication and role-based access control to ensure that only authorized administrators can modify academic rules or datasets. Student accounts should be protected with secure password storage, and communication between the frontend and backend should follow secure protocols.

Another key requirement is usability. IAAS is intended to assist students who may not be familiar with academic terminology or complex degree structures. For this reason, the interface must be simple, readable, and easy to navigate. The Chat Advisor should provide clear explanations, and the degree progress dashboard should present information in a way that helps students quickly understand where they stand. The Admin Panel should also be designed with ease of

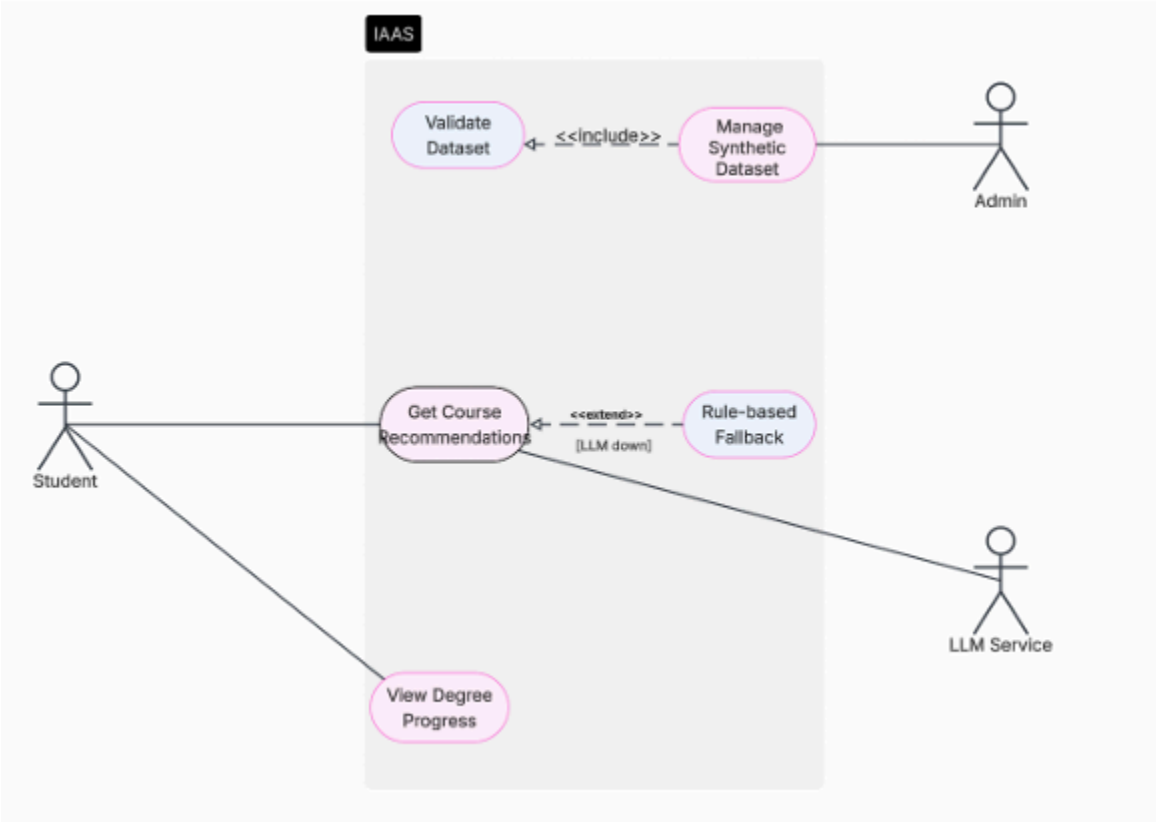
use in mind so that advisors and staff can update academic data without needing technical expertise.

Scalability is also an important consideration. As the university offers new degree programs or updates existing ones, the system should be able to grow without needing major architectural changes. The rule engine and data structures should be flexible enough to support additional courses, prerequisites, and program requirements. Future enhancements, such as machine-learning-based recommendations or real-time integration with the university information system, should fit naturally into the architecture.

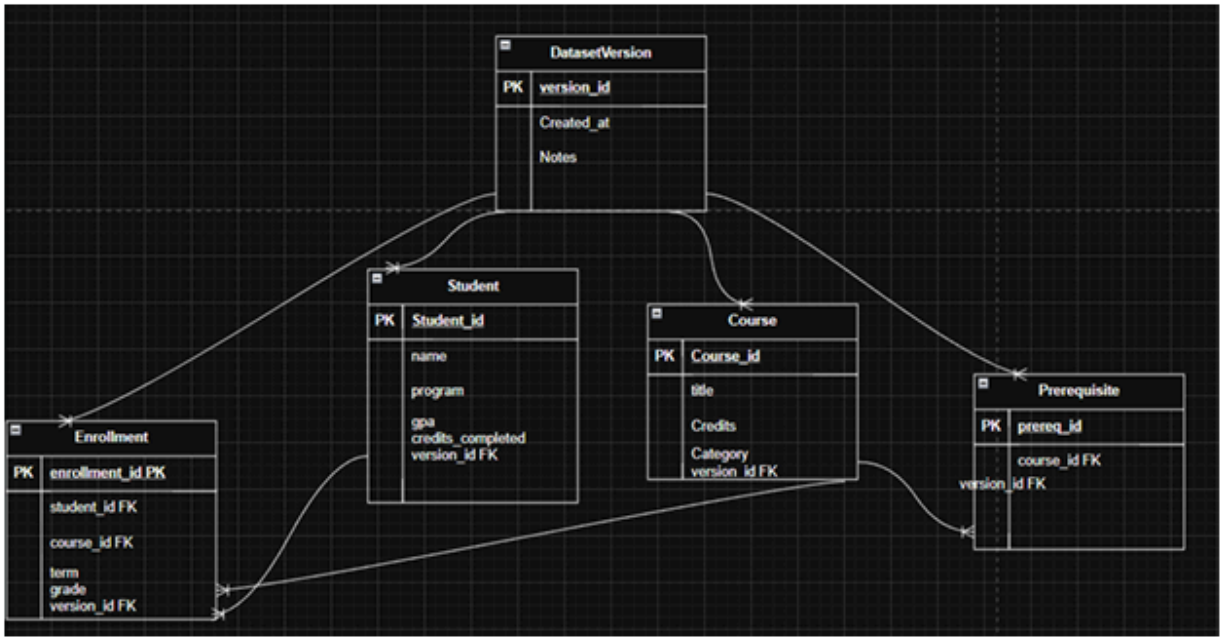
Finally, maintainability is a key non-functional requirement. Because academic rules and program structures change over time, the system must be easy to update. Good documentation, clean module design, and separation of concerns will make it easier for future developers to update system components without risking other parts of the system.

Together, these non-functional requirements ensure that IAAS is not only functional but also high-quality, dependable, secure, and user-friendly. They provide a framework that guides the development of the system in Capstone II and help guarantee that IAAS will be a reliable tool for students and advisors once it is fully implemented.

3.4 Use Case Diagrams



3.5 Entity Relationship Diagram



3.6 Requirements Traceability Matrix

Objective ID	Objective Description	Related Requirements	Requirement Description
OBJ1	Accurate and actionable academic advising	FR5, FR10, NFR1, NFR2	FR5: LLM-based chatbot provides personalized recommendations FR10: Prerequisite validation ensures accuracy NFR1: Fast response times for usability NFR2: 90% accuracy in recommendations
OBJ2	Transparent progress tracking and usability	FR3, FR4, NFR5	FR3: Interactive degree progress visualization FR4: Searchable course information NFR5: Cross-browser compatibility
OBJ3	Simple, safe dataset management	FR1, FR2, FR6, NFR4	FR1: Load pre-generated synthetic datasets FR2: Comprehensive data validation FR6: Admin interface for dataset management NFR4: Dataset versioning and immutability

OBJ4	System reliability and recoverability	FR7, FR9, NFR3	FR7: Comprehensive logging for debugging FR9: Rule-based fallback system
OBJ5	Maintainability and auditability	FR7, FR8, NFR4	FR7: Audit trail of all interactions FR8: PDF export for documentation NFR4: Reproducible testing through versioning

4. System Design and Architecture

4.1 the Architectural Style

The Intelligent Academic Advisory System (IAAS) is based on a Layered Architecture Style, this architecture organizes the system into separate layers, each responsible for a specific set of functions.

The layered architecture of IAAS is composed of four primary layers:

- Presentation Layer
- Application Layer
- Domain Logic Layer
- Data Layer

1. Presentation Layer

The Presentation Layer is responsible for all interactions between the system and its users, It includes the web-based dashboards, forms, planning tools, chatbot interface, and admin management pages. This layer focuses on user experience design, accessibility, and the presentation of information in an easy-to-navigate format.

Responsibilities include:

- Displaying degree progress dashboards
- Providing interfaces for semester planning
- Managing chat interactions through the Chat Advisor
- Allowing admins to upload and edit academic datasets

The Presentation Layer communicates exclusively with the Application Layer, ensuring that all business logic remains separated from UI components.

2. Application Layer

The Application Layer acts as the intermediary between the user interface and the internal system logic. It manages data flow, controls system processes, and handles incoming requests from the Presentation Layer. This layer contains controllers, request handlers, and service modules that executes the operations across the system.

Responsibilities include:

- Handling student and admin requests
- Routing queries to the appropriate domain logic components
- Managing authentication and session control
- Serving RESTful API endpoints to the frontend
- Coordinating interactions between modules

By centralizing these operations, the Application Layer ensures a consistent, controlled interaction with lower system layers.

3. Domain Logic Layer

The Domain Logic Layer contains the core of IAAS and performs all rule-based and computational operations. It is the most critical component of the architecture, as it structures how academic data is interpreted and how decisions are made.

Responsibilities include:

- Executing prerequisite validation logic
- Interpreting academic rules and program structures
- Generating automated semester plans
- Running course recommendation logic

- Producing Chat Advisor responses based on academic rules
- Handling degree progress calculations

This layer ensures that all academic reasoning within IAAS is centralized, consistent, and independent of both the user interface and data storage mechanisms.

4. Data Layer (Database and Storage Layer)

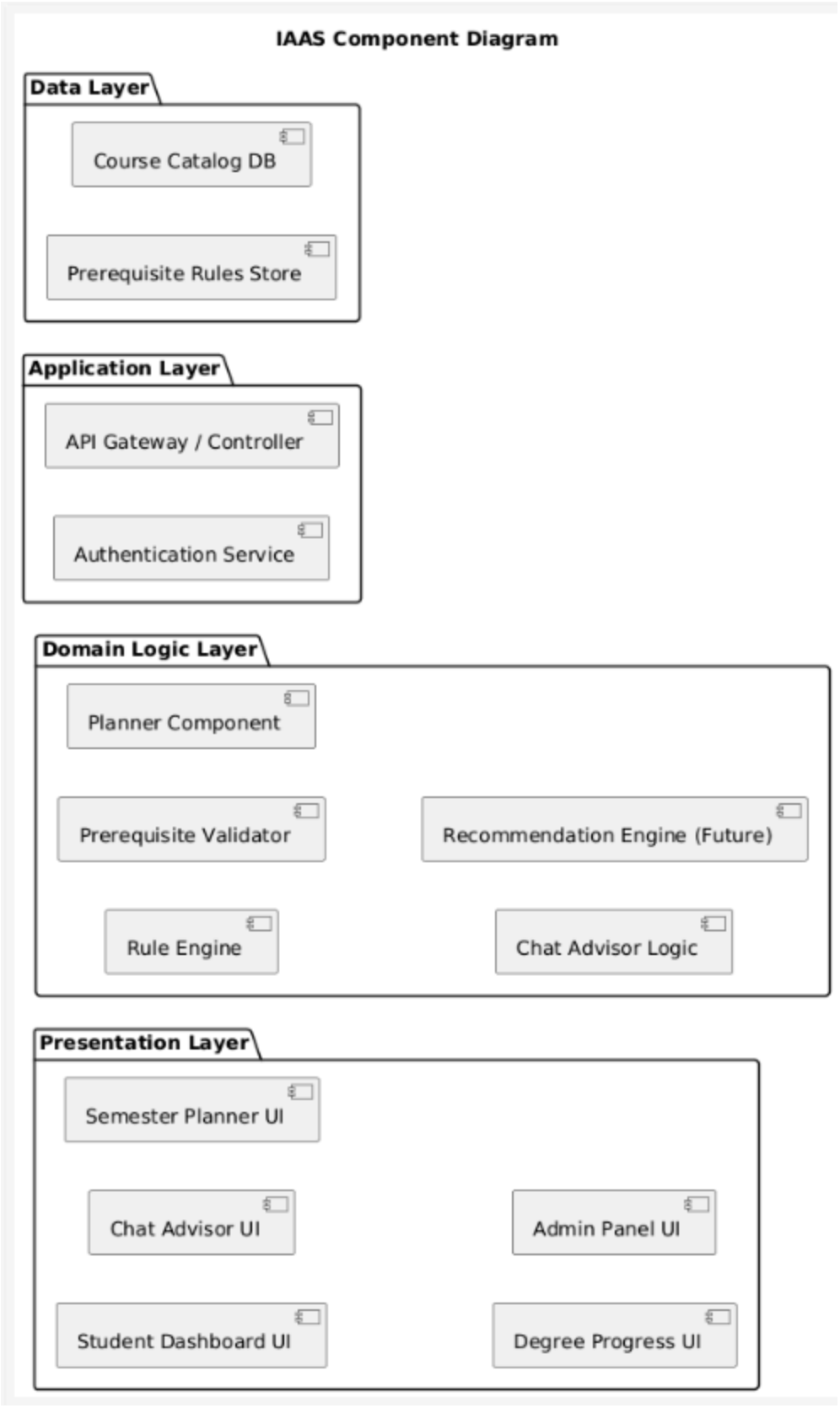
The Data Layer is responsible for all long-term storage and retrieval of academic data. It includes the system's databases, structured datasets, files, and data access interfaces.

Responsibilities include:

- Storing course catalogs
- Maintaining prerequisite maps
- Managing degree requirements and program structures
- Holding student academic profiles
- Ensuring data integrity and consistency
- Providing secure access to stored information

Data in this layer is accessed exclusively through the Domain Logic Layer or repository interfaces, ensuring controlled and consistent data usage throughout the system.

4.2 UML Component Diagram


IAAS Component Diagram
Data Layer
Course Catalog DB
Prerequisite Rules Store
Application Layer
API Gateway / Controller
Authentication Service
Domain Logic Layer
Planner Component
Prerequisite Validator
Recommendation Engine (Future)
Rule Engine
Chat Advisor Logic
Presentation Layer
Semester Planner UI
Chat Advisor UI
Admin Panel UI
Student Dashboard UI
Degree Progress UI

4.3 Key Design Decisions and Rationale

One of the main key decisions was implementing the layered architecture, this was chosen to isolate each major component so that everything looks neat and can be changed or updated without configuring the whole system

Another key decision was to implement a rule based engine to validate prerequisites, academic programs must adhere to to strict policies that change semester to semester and from each doctor, if we hardcode this into the system then updating it per semester would be time costly and repetitive, this gives admins the free will to update the system regularly with the change of policies or course timings or program prerequisites.

Another key decision was to design each core advisory function as independent components. Treating these modules as standalone subsystems makes the IAAS platform more modular and scalable. Each component can be expanded or replaced in the future without disrupting the rest of the system.

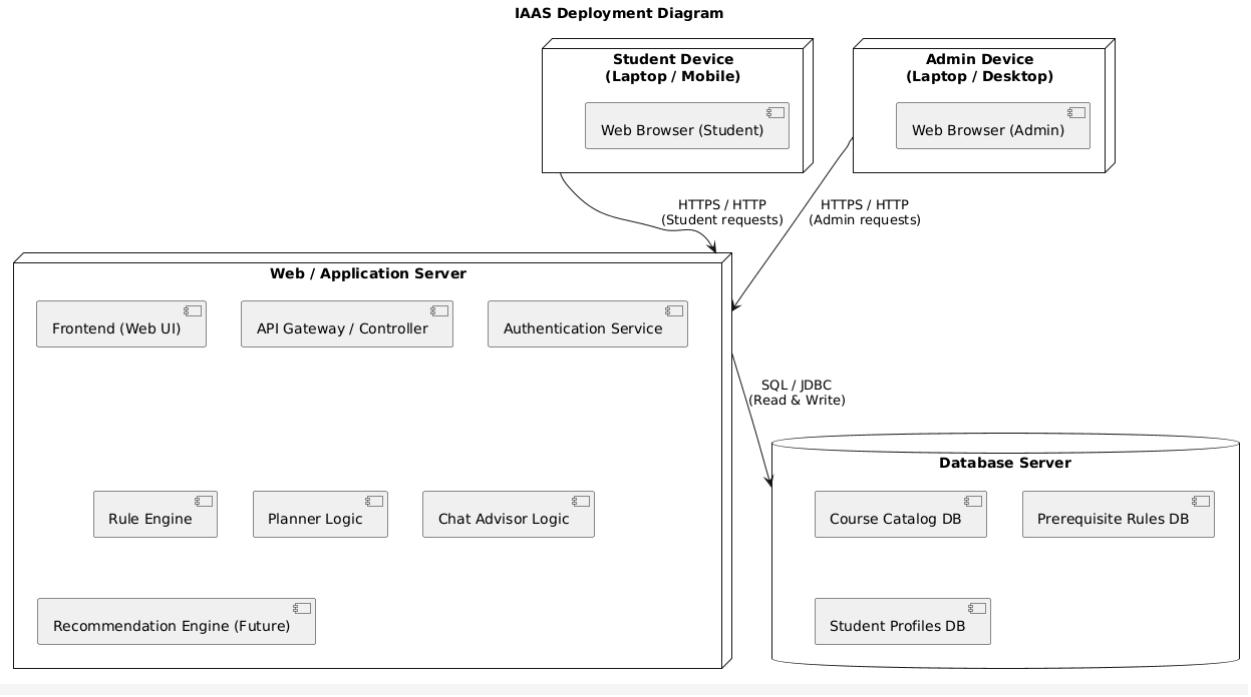
From a user experience standpoint, the team decided to implement IAAS as a web-based system accessible from standard browsers. This decision was chosen because of the diverse needs of students and administrators, who require immediate access to academic tools without installing specific applications. A web interface simplifies deployment and testing and ensures that the system remains accessible across devices and platforms

Data organization was another key design consideration. Academic Data such as course information, prerequisites, program structures, and student records, needed to be stored in structured and easily maintainable formats. The team chose to design the data layer to support both relational databases and structured files,ensuring flexibility during early development and scalability in future phases.

We chose to use RESTful APIs to manage communication between the user interface and backend logic. It ensures that interactions between system layers remain consistent and organized, allowing the frontend, backend, and data layers to evolve independently.

Security considerations also played a vital role in the system's design. IAAS handles sensitive academic information, so implementing clear role separation between students and administrators was essential. The system was designed with access control mechanisms that prevent unauthorized modifications to datasets or program rules. Although full security specification and implementation will occur in Capstone II, establishing these design principles early ensures that the system remains aligned with university policies and data protection expectations.

4.4 Deployment View



4.5 Boundary Conditions

IAAS must establish all of the core components during initialization. The first step is to load the academic data stored to the memory, for example course lists and prerequisites, then the system prepares the main modules like the rule engine and the planner so they are ready to handle requests, and finally the system establishes a connection with the database and start the service required for users to log in, when all this is completed then the system is ready to use

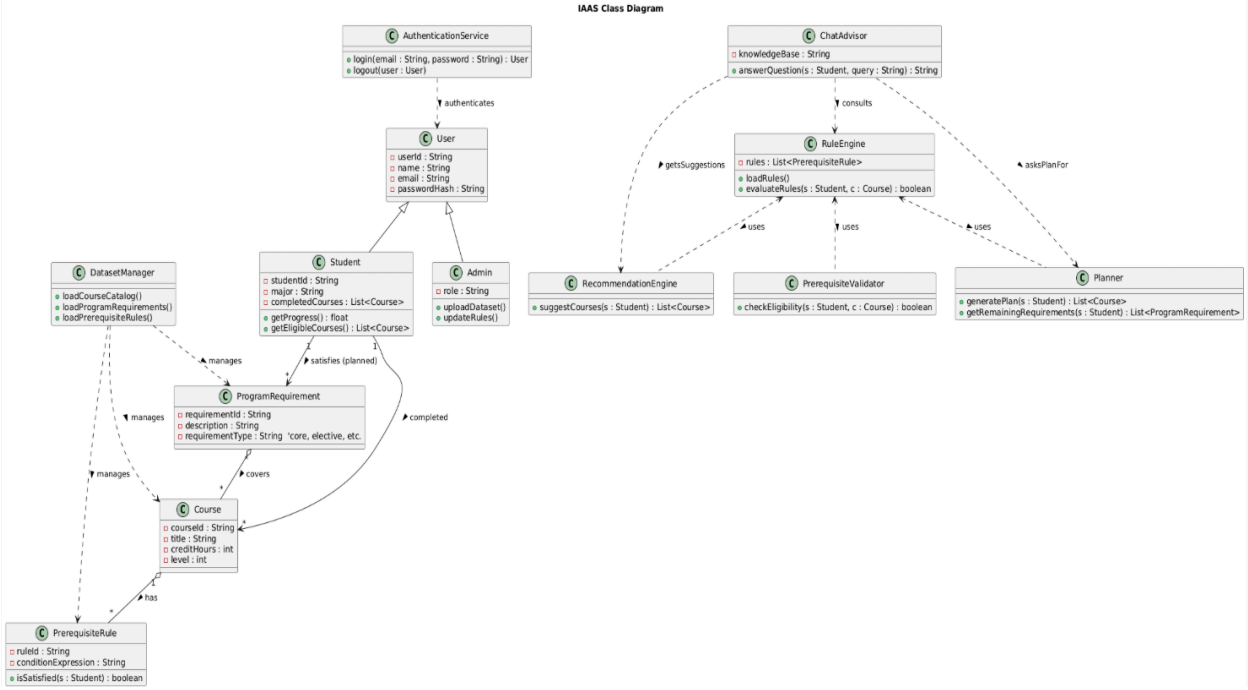
In the termination phase, the system will close all active sessions and disconnect safely from the database to prevent data loss, and it saves logs or records to help with future debugging. After all this the system shuts down fully

Failure handling ensures that the system will respond in a a safe and predictable way, for example if a dataset is missing. The system should notify the admin instead of trying to give incorrect answers or results, or if a student enters an unsupported request, the system should return a simple message rather than overdo it and crash the whole system.

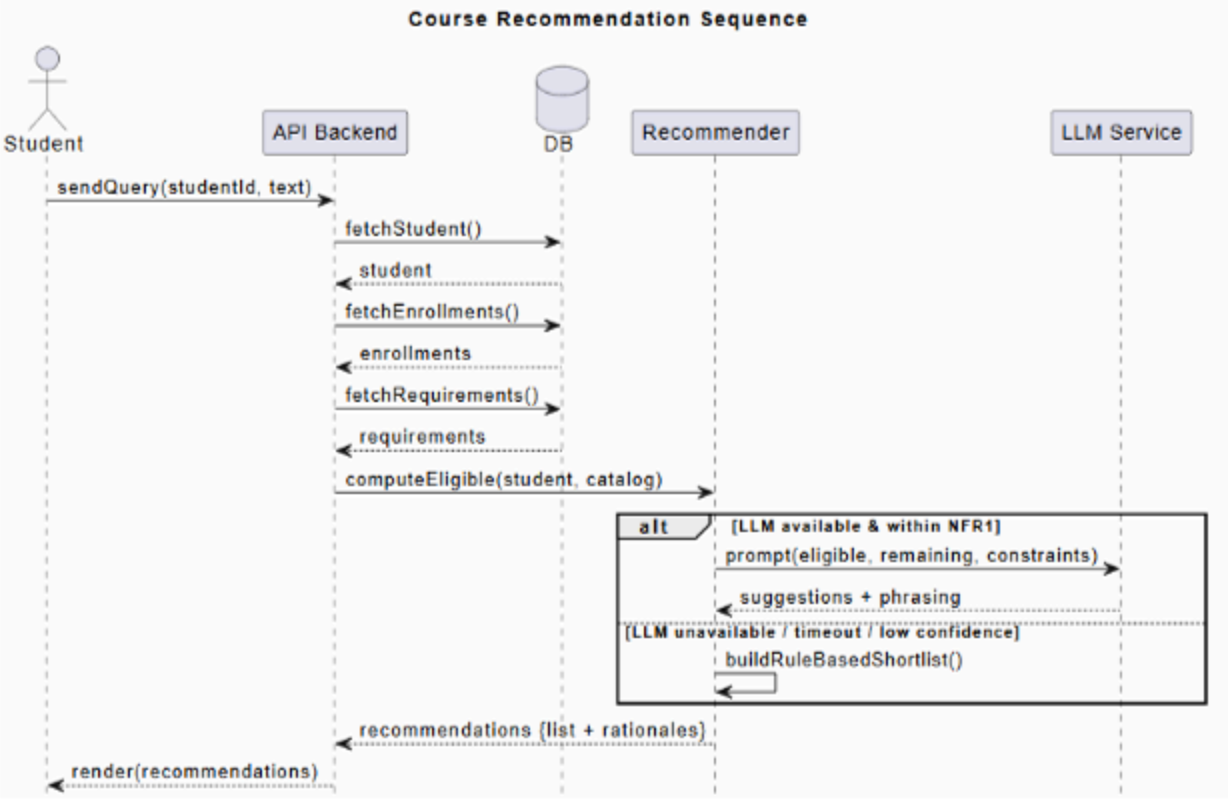
In summary, the boundary conditions of IAAS ensure that the system behaves predictably and safely during startup, shutdown, and unexpected failures. By defining clear initialization routines, controlled termination procedures, and robust failure-handling strategies, Our System maintains reliability and protects academic integrity.

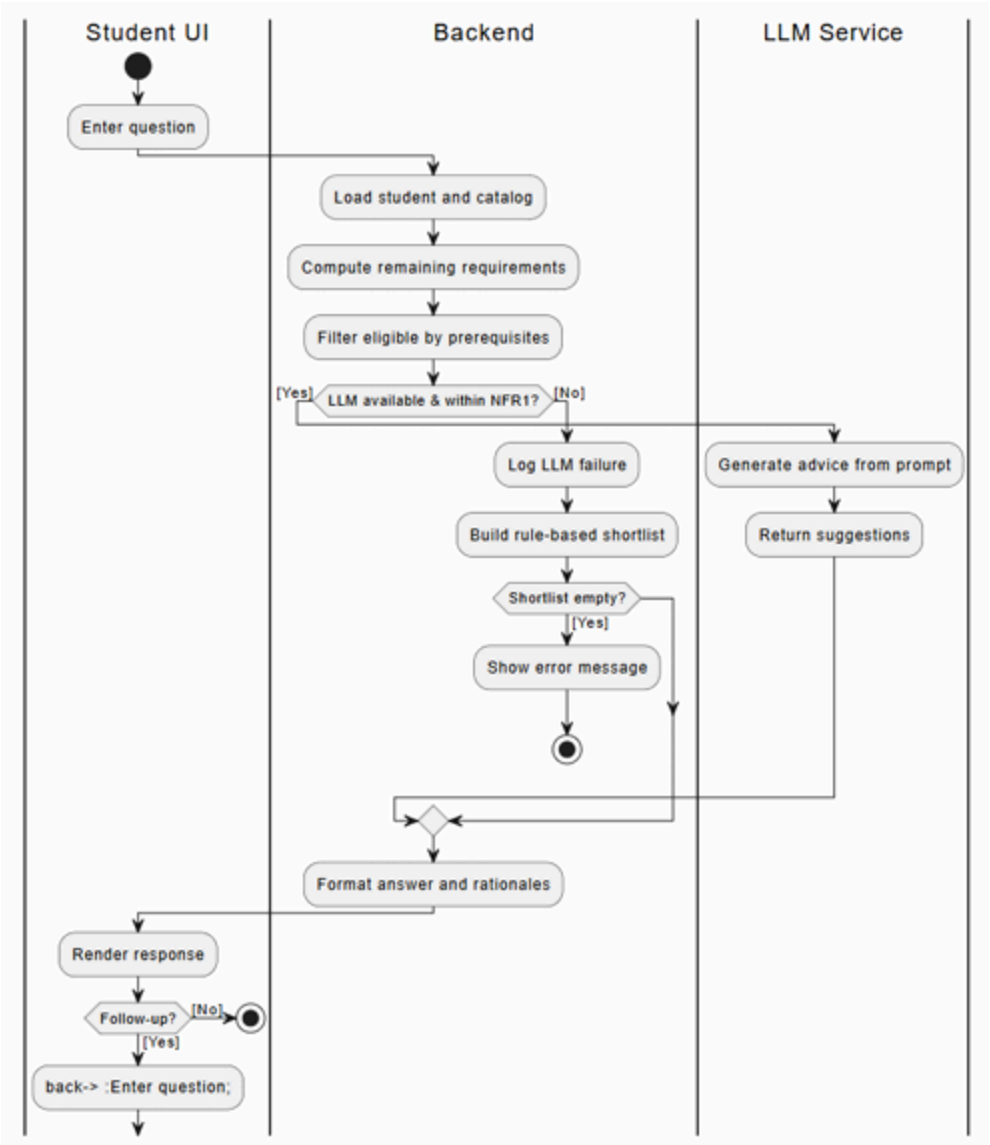
5. Object-Oriented Design

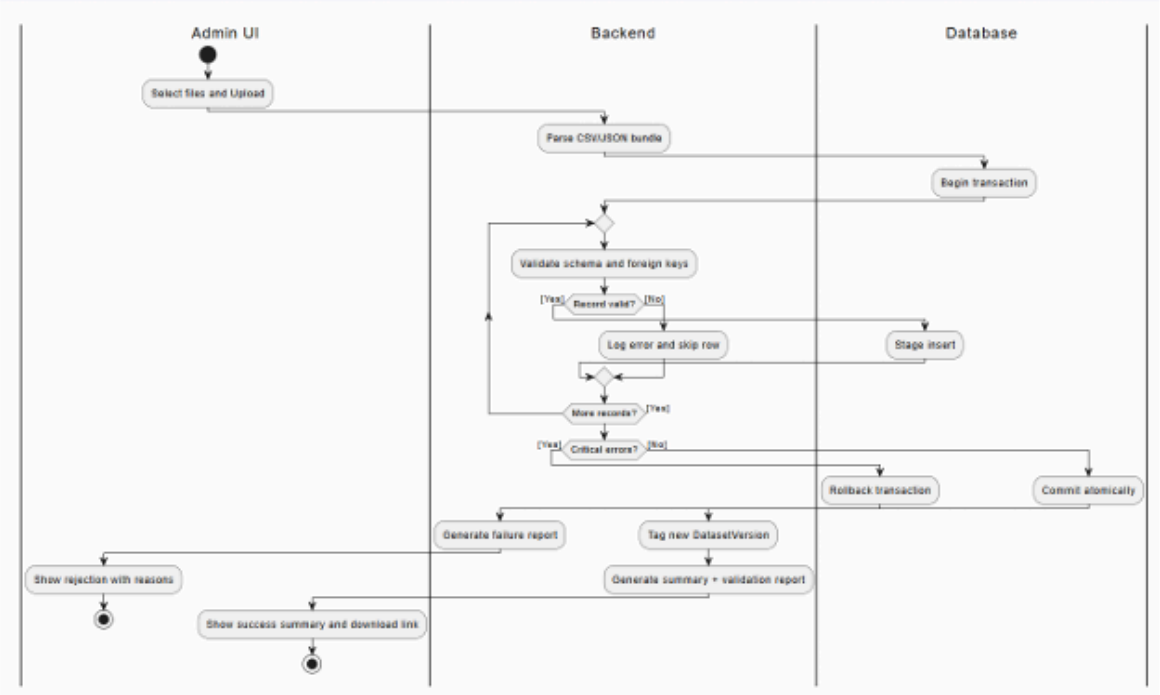
5.1 Class Diagram



5.2 Sequence and Activity Diagrams


Course Recommendation Sequence
Student
API Backend
DB
Recommender
LLM Service
sendQuery(studentId, text)
fetchStudent()
student
fetchEnrollments()
enrollments
fetchRequirements()
requirements
computeEligible(student, catalog)
alt
[LLM available & within NFR1]
prompt(eligible, remaining, constraints)
suggestions + phrasing
[LLM unavailable / timeout / low confidence]
buildRuleBasedShortlist()
recommendations (list + rationales)
render(recommendations)


Student UI
Backend
LLM Service
Enter question
Load student and catalog
Compute remaining requirements
Filter eligible by prerequisites
[Yes]
LLM available & within NFR1?
[No]
Log LLM failure
Generate advice from prompt
Build rule-based shortlist
Return suggestions
Shortlist empty?
[Yes]
Show error message
Format answer and rationales
Render response
Follow-up?
[No]
[Yes]
back-> :Enter question;



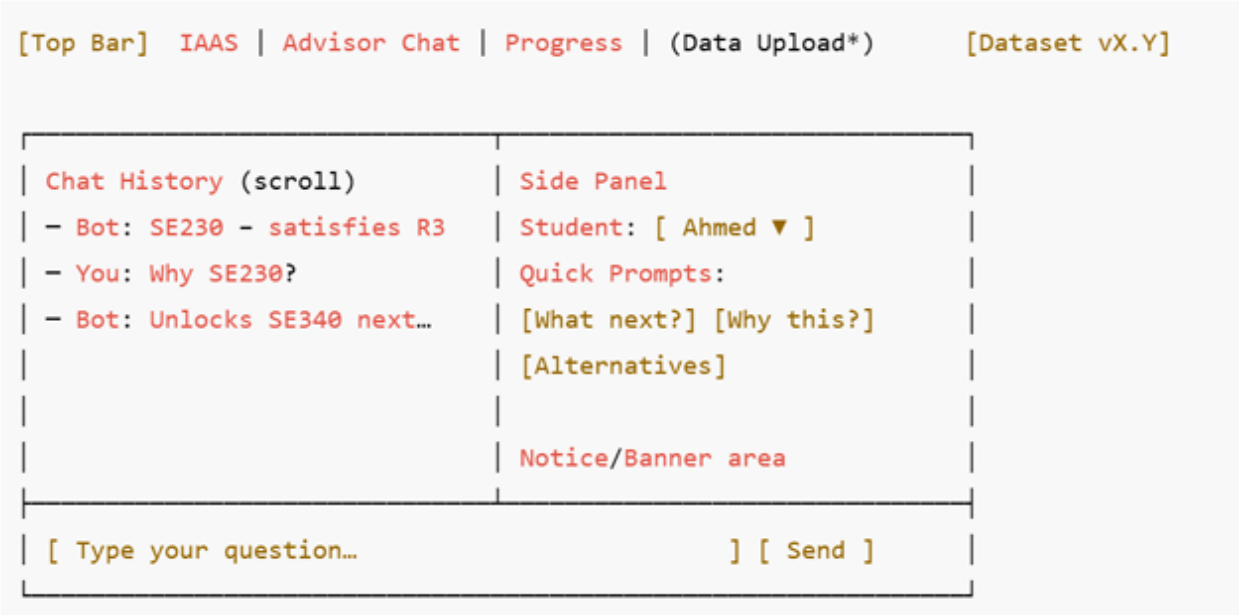
5.3 User Interface Structure

Student Navigation Flow:

- Entry → Advisor Chat (default).
- Chat → stays on page; may open Progress via top nav.
- Chat error (LLM down) → inline banner; remains on Chat.
- Top nav → Progress.
- Progress → click course node → Details panel (in-page).
- Progress warning (inconsistent data) → banner; stays on Progress.
- Progress → top nav → Advisor Chat.

Administrator Navigation Flow:

- Entry: Admin Login → Session Creation → Admin Dashboard
- Data Management: Upload Dataset → Validation → Review Report → Activate/Reject
- Monitoring: View Logs → Filter by Date/Type → Export Reports



6. Project Management Overview

6.1 Work Breakdown and Roles

Feras Alkahtani focused on the conceptual planning of the backend and the structure of the system. His work included helping outline how the core advising components should function, such as the rule engine and the overall academic logic that the system will eventually use. He also contributed heavily to writing and refining different parts of the documentation, especially the requirements and design sections. His role centered on shaping how these components should behave once development begins in Capstone II.

Abdulaziz Albaz worked on defining the direction of the user interface and the general user experience. He sketched early versions of the main screens, and was heavily involved in all the diagram drawings. He also participated in writing and reviewing several parts of the report and helped ensure the UI concepts matched the system's goals.

Mohammed Alhajri concentrated on organizing the academic data structure and supporting the development of the requirements. He worked on outlining how course information, prerequisites, and degree requirements should be arranged for future implementation. He also helped prepare

the functional and non-functional requirements, contributed to the use case descriptions, and participated in documenting the overall system behavior.

Because our team is small, many tasks were shared between all members, and a lot of the work was done together. We often worked together on the requirements, the diagrams, and the system design discussions, making decisions as a group and helping each other refine ideas. Capstone I was mainly focused on planning and documentation rather than implementation, so most of the work involved conceptual development and preparing a clear foundation for the next phase.

6.2 Ethical and Professional Considerations

The team approached the design of the system with a strong emphasis on correctness and accountability, because any automated advising system has the potential to influence a student's academic trajectory, making reliability and accuracy ethical obligations rather than merely technical requirements.

Another critical ethical consideration involves data privacy and confidentiality. IAAS uses academic datasets, student progress information, and program structure files that must be handled with strict confidentiality. Even though the Capstone I conceptual model uses test or manually prepared data rather than live student records, the design decisions anticipate real-world deployment conditions. Access control mechanisms were considered from an early stage to ensure that student data remains private and that only authorized users can modify curriculum files or prerequisite rules.

Through past experiences we realized that traditional advising can sometimes be affected by inconsistencies or differences in interpretation among advisors. IAAS aims to provide unbiased advising that treats all students equally, but the team keeps in mind that automated systems may overlook exceptional or personal cases, that's why IAAS is a supporting tool to the advisors not something to take over traditional advising fully. Its purpose is to reduce the workload and provide students with immediate support while still allowing academic staff to review personal or unusual cases. This model ensures fairness without removing the personalized human element needed in certain academic situations.

Finally, professional responsibility is must, it is a students right to know why the system dictates whether course A is a right or wrong course, this is why explanations would be needed to help students understand the systems point of view, and that at the end of the day it is a system designed to give the most optimal answer and has a lack of emotion, but at the end of the day the system must be rational about its advising decisions.

7. Conclusion and Final Work

During Capstone I, our goal was to understand what IAAS should do and plan how it should work once we begin. In this semester we focused on gathering requirements, studying the advising process, and designing the main parts of the system. Although no functional components have been built, we were able to create a clear picture of how the system should behave and what users will interact with.

This project taught us the importance of planning before coding, and of the dangers of not minimizing the scope. We learnt how to break down the system into several smaller components and begin working on it from scratch rather than seeing the whole picture, we also learned the value of working as a team and with a small team we did not take our numbers for granted.

In Capstone II, our focus will be on building what we specified in Capstone I, we also intend to uphold the design decisions and planning established in Capstone I and shift them into a working system during Capstone II. We will implement the backend, develop the UI, connect datasets and make the advising system functional, we also plan to test each feature to make sure it corresponds with the level of quality we seek. By the end of Capstone II, we will have a working version of the IAAS that users can use for real advising tasks.