Software Requirements Specification

(SRS)

Intelligent Academic Advisory System

(IAAS)

Team Members:

Feras Alkahtani

Abdulaziz Albaz

Mohammad Alhajri

Supervisor:

Professor Anis Koubaa

Date:

October 18, 2025

1. Proposed System (Overview)

1.1 Purpose

(IAAS) is a prototype web based application that provides Software Engineering students with automated academic advisory by using an AI-powered chatbot and a visual degree progress dashboard.
The primary purpose is to assist students in planning their selection of courses, offering personalized recommendations using a large language model based chatbot. The system addresses the need for more scalable and accessible advising by simulating the guidance an academic advisor would provide.
This helps students make informed decisions about upcoming semesters and understand their progress without the need of constant physical advisor meetings

1.2 Scope

The scope focuses on core functionalities. The system includes three main components: The first component is an (LLM) driven chatbot that allows students to ask and receive recommendations in natural language.The second component is a degree progress visualization interface that displays the student's completed and remaining degree requirements in a clear, understood graphical format. The third component is Synthetic Data Management features to load and process a pre-generated dataset of student records and course information.

1.3 Motivation

The motivation stems from critical challenges in current academic advising systems. Engineering programs face a significant advisor-to-student ratio problem, with advisors handling 150-200 students each, limiting personalized guidance. Students frequently select courses that violate prerequisites or delay graduation due to poor planning. Manual advising cannot scale to meet increasing demand, and students often cannot get timely appointments during registration periods. IAAS addresses these challenges by providing 24/7 accessible, personalized academic guidance that helps students make informed course selections. The system targets undergraduate Software Engineering students as primary users, with academic advisors as secondary beneficiaries who can use the tool to help their advising process. By automating routine advising sessions, the system frees advisors to focus on complex cases requiring human judgment.

1.4 Ethical Considerations and Disclaimer

IAAS is designed as an advisory tool to help, not replace, official academic advising. The system provides recommendations that are advisory only and must be verified with academic advisors before course registration. Users should be aware that the system may contain inherent biases from training data or model limitations and does not account for all individual circumstances or special cases. The system cannot consider personal factors such as work schedules, family obligations, or learning preferences that might affect course selection. All recommendations should be validated against official degree requirements and discussed with academic advisors. The user interface displays a persistent banner stating: "IAAS recommendations are advisory only. Please consult your academic advisor before finalizing course registration."

2. Functional Requirements

Table 1: Functional Requirements

Requirement ID	**Description**	**Priority (H/M/L)**	**Rationale**	**Acceptance Criteria**
FR1	The system shall load and process pre-generated datasets of 500 students and 200 courses with their information	High	Enables system functionality using synthetic data	Given valid CSV/JSON files, the system inserts all records into database tables within 30 seconds with 100% schema compliance and produces a validation report
FR2	The system shall validate dataset integrity before storage	High	Prevents inconsistent data from affecting recommendations	Validation reports show 0 schema errors, 100% referential integrity for all course references

FR3	The system shall display interactive visualization of student degree progress with prerequisite chains	High	Helps users understand completed and pending requirements	For 10 test profiles, the system correctly calculates completed/remaining credits with 100% accuracy with a response time of ≤2s
FR4	The system shall allow users to search and view course information and prerequisites	High	Supports academic planning with accurate course data	For 20 catalog courses, returned title, credits, and prerequisites exactly match the catalog within 1s
FR5	The system shall provide LLM-based chatbot using OpenAI GPT-4o-mini API	High	Offers cost-effective, accurate academic advising through natural language	For 50 test queries, chatbot returns recommendations with 0% prerequisite violations and a response time of ≤2s
FR6	The system shall provide authenticated administrator interface to manage and refresh datasets	Medium	Allows dataset updates without developer intervention	Upload of the standard dataset completes ≤60s. And 0 invalid rows are inserted and a validation report file is produced with error counts

FR7	The system shall log all chatbot interactions and recommendation requests with timestamps	Medium	Enables system debugging and quality evaluation	100% of chatbot interactions are logged with timestamp,
FR8	The system shall generate PDF summaries of course plans and progress reports	Low	Provides offline documentation for students	Exported PDF matches on-screen data for the same student with 0 discrepancies across 5 samples
FR9	The system shall automatically switch to deterministic rule-based recommendations when OpenAI API monthly quota ($20) reaches 90% consumption	High	Ensures continuous operation within budget constraints	System monitors token usage in real-time; at 90% quota, it will activate rule-based fallback within 3 seconds and returns valid recommendations with 100% prerequisite compliance
FR10	The system shall validate all chatbot recommendations against prerequisite rules before presentation	High	Prevents invalid course suggestions	100% of recommendations pass prerequisite validation; invalid suggestions filtered with explanation

3. Non-Functional Requirements

Table 2: Non-Functional Requirements

NFR ID	Description	Rationale	Measurement/Test
NFR1	The system shall respond to user requests within 2 seconds under normal load	Improves usability and user satisfaction	Perform load testing with 20 simultaneous users, then verify the median response time ≤ 2 seconds
NFR2	The chatbot shall maintain ≥90% accuracy in mapping queries to valid course recommendations	Ensures reliability of AI-driven advising	Evaluate against 50 benchmark queries and measure precision of ≥90%

NFR3	The system shall maintain 95% functional uptime during testing periods	Ensures system reliability and Up time availability	Continuous uptime monitoring over a 7 day test cycle
NFR4	The system shall preserve dataset versioning and immutability	Guarantees experimental repeatability	Reloading the same dataset version produces identical outputs and validation reports are stored alongside the version
NFR5	The system shall function correctly on Chrome/Edge v100+ with consistent UI rendering	Ensures cross-platfor m accessibility	Cross-browser testing on 5 browser versions and then verify identical functionality and layout

4. System Characteristics

4.1 User Interface

The user interface is designed to be simple and focused on two main features: the chatbot and the degree progress view. The student dashboard has a clean simple layout with a navigation menu or tabs for "chat advisor" and "Degree Progress". In the chat advisor section the user sees a chat window, a text input box, and a scrollable conversation, this interface feels familiar to many users and resembles common messaging apps. The chatbot interface will use a clear font that differentiates between the users question and the bots answer and perhaps to ensure readability. In the degree progress section, The Interface presents a visual diagram or a structured list of courses. Perhaps a flowchart or a roadmap of courses organized by semester or by prerequisite structure. Completed courses could appear in green while unfinished or pending courses could appear in red or gray colour. By hovering over a said course the user can see the details about the course, title, credit hours, section, pre-requisites. The interface is responsive on various screen sizes but optimized for desktop viewing

4.2 Documentation

Comprehensive documentation supports all user types. The Student User Manual provides a guide with annotated screenshots explaining chatbot usage, sample queries for common scenarios, interpretation of degree progress visualizations, and troubleshooting common issues. The Administrator Guide offers a manual covering dataset preparation requirements, upload procedures with validation rules, interpretation of validation reports, system configuration options, and error resolution procedures. Technical documentation includes API reference with all REST endpoints (admin endpoints protected), data schemas in JSON format, database entity relationships, and integration guidelines. All documentation is available through the "Help" menu.

4.3 Hardware Considerations

A standard development laptop or a small server can host the backend. Minimum recommended server specs include a modern CPU, 8 GB of RAM, and about 10 GB of free storage space. This is sufficient to run the web server and SQLite database. The LLM inference is cloud-hosted via the OpenAI API. Students or admins need a device capable of running a modern web browser, there is no need for specialized hardware on the client side, any typical desktop or laptop from the last 5-7 years with an internet connection suffices.

4.4 Error Handling & Extreme Conditions

The system implements comprehensive error handling. When the OpenAI API quota ($20 monthly limit) reaches 90% consumption, the system automatically switches to rule-based recommendations within 3 seconds, displaying a notification: "Approaching monthly AI budget limit, switching to simplified recommendations. Full service resumes next billing cycle." This ensures continuous service. API failures or timeouts also trigger the rule-based fallback with user notification.

4.5 System Interface

The backend works as a web service built with a framework like FastAPI.The browser talks to the backend using HTTP requests. For example, when a user types a question in the chatbot, the browser sends it to an internal link, and the backend sends back the answer. These APIs are only for the system's own use, not public. IAAS uses a simple SQLite database to store synthetic student, course, and enrollment data. The backend connects to this database through either ORM code or direct SQL queries. When the admin uploads new data, the backend automatically runs logic to insert all records into the database in bulk. The system connects to an LLM service that powers the chatbot. It sends the student's data and question to the AI model and receives a reply. The project uses an online model, it calls an external API using an access key and internet connection.

4.6 System Modifications

The system connects to an LLM service that powers the chatbot. It sends the student's data and question to the AI model and receives a reply. The current version includes only core features. Future versions can add GPA forecasting, risk analysis, or real student data. The UI is simple but expandable. New pages or tools can be added easily since the navigation and design are consistent. Many settings come from files, not hard-coded values. If the curriculum changes, the admin can upload a new dataset instead of editing the code.

4.7 Constraints

The project runs for two semesters (about 8–9 months). Only core features are built and tested. Three student developers, OpenAI monthly budget. Built with open-source tools. It must run on standard lab computers. Only synthetic data is used for privacy. The dataset models one program and skips complex real-world cases. Used only in demos or evaluations.

4.8 Performance Characteristics

Chatbot and progress pages should load in under 3 seconds. Tests show the AI replies within 2 seconds. It handles up to 20 users at once without slowdown about 150 chatbot queries per minute in total.The database lookups are indexed, and algorithms are lightweight. Even large synthetic datasets for example 10k students, 500 courses can run easily. The system efficiently uses little memory under 200 MB on average.

4.9 Quality Issues

Quality assurance for IAAS identifies potential risks and issues that could affect the quality of the system, along with strategies to mitigate them. Course recommendations are tested against degree rules to prevent wrong course advice. Synthetic data is reviewed and validated to prevent logical errors. Peer testing will improve UI clarity and labeling. Error cases like no recommendations and finished degree are handled gracefully to ensure user friendliness. Clean, commented code. Tested with synthetic data only. Real-world testing is left for future phases.

4.10 Physical Environment

IAAS will operate in a typical academic computing environment. IAAS is made to run on normal university computers or laptops. There will be no special hardware needed, and the users can access it through a browser, often in a lab or classroom. It is also designed to display clearly on projectors and standard screens, and will need a stable network

4.11 Security Issues

The admins will log in with a username and password, and the admin pages will be protected. Even synthetic data is handled carefully. Users will only be able to see relevant records. All communications use HTTPS where possible, and the admin passwords are securely hashed. The admin sessions will expire automatically after certain time of inactivity. The chatbot is limited to the course data and blocked from giving unrelated or unsafe answers.

4.12 Resource Issues

The complete software stack includes Python 3.9+ for backend logic and API development, FastAPI framework for REST endpoint implementation, SQLite database for data persistence, SQLAlchemy ORM for database abstraction, OpenAI GPT-4o-mini API for natural language processing (with $20/month budget allocation), HTML5/CSS3/JavaScript for frontend development, D3.js library for interactive visualizations, bcrypt for secure password hashing, and Git for version control. The system includes server-side usage logs and a simple quota meter showing current/monthly spend and percentage of cap. Logs can be exported for review. Development tools include Visual Studio Code or PyCharm IDE, Postman for API testing, and Docker for containerization.

5. System Models

5.1 Scenarios/User Stories

Story 1: Course Recommendation Request

As a student, I want personalized course recommendations for next semester, so that I can register for appropriate courses that align with my degree progress and academic standing.

Context: Alia is a third-year Software Engineering student who has completed 60% of her degree requirements. She struggles with mathematics courses and prefers project-based learning. She needs to select 4-5 courses for the upcoming semester.

Preconditions: Student record exists in the system; dataset loaded. Course catalog for the upcoming semester is loaded. LLM service is operational.

Trigger: Alia opens IAAS and accesses the Chat Advisor interface.

Main Flow: (1) Alia types "What courses should I take next semester?" (2) System loads Alia's academic profile from dataset. (3) LLM analyzes profile: completed courses and remaining requirements. (4) System generates initial recommendations based on degree requirements. (5)

Prerequisite validation confirms all suggested courses are eligible. (6) System returns 4-5 courses with explanations. (7) Alia asks follow-up questions as needed.

Alternative Flows: If no eligible required courses remain, system suggests electives. If LLM service fails, rule-based system activates within 3 seconds. If prerequisites not met, system explains requirements needed first.

Acceptance Criteria: Recommendations include only courses with satisfied prerequisites. Response median ≤2s, p95 ≤3.5s under normal load. Each recommendation includes justification referencing degree requirements.

Outcome: Student receives an actionable plan and can request “why” or alternatives.

Postconditions: Interaction logged, and the progress dashboard remains consistent.

Story 2: Administrator Dataset Update

As an administrator, I want to upload an updated course catalog with new prerequisites, so that the system reflects current curriculum changes.

Context: The Software Engineering program has revised prerequisites for several courses and added two new electives for the upcoming academic year.

Preconditions: Admin authenticated and an active dataset version is set.

Trigger: Open Admin Data and click “Upload dataset”.

Main Flow: (1) Admin logs into the administrative portal with credentials. (2) Admin selects "Upload Dataset" and chooses prepared CSV files. (3) System validates file format and schema compliance. (4) System checks referential integrity. (5) System generates validation report. (6) Admin reviews and confirms upload. (7) System activates new dataset version. (8) Confirmation message displays with timestamp and version number.

Alternatives flows: Validation failures then it will reject the upload and provide a report. Partial logical issues then it will skip bad rows and warn. LLM unavailable then it will show error message.

Acceptance Criteria: Upload completes within 60 seconds for standard dataset. Validation report identifies all issues before activation. Previous version remains accessible for rollback.

Outcome: System runs with the new catalog immediately.

Postconditions: Action logged, and prior version remains restorable.

5.2 Use Case Model

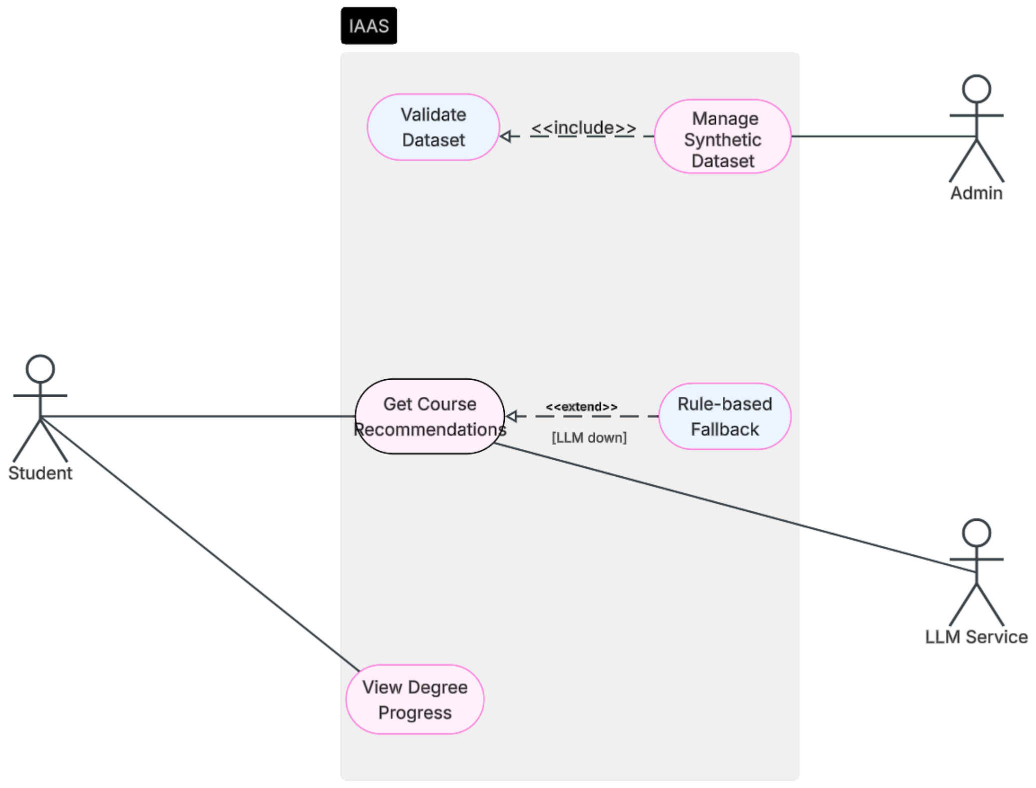
Use Case 1: Get Course Recommendations

- Actors: Student (Primary), System
- Preconditions: Student record exists; dataset loaded
- Main Flow:
    1. Student requests recommendations.
    2. System loads records and rules (prereqs, caps, conflicts).
    3. System calls LLM Service with structured context
    4. System validates LLM output against hard rules.
    5. System shows ranked recommendations with rationale.

Alternatives:

- LLM unavailable/over-budget/invalid → UC-3 (Rule-based Fallback).

- After validation nothing remains → UC-3.



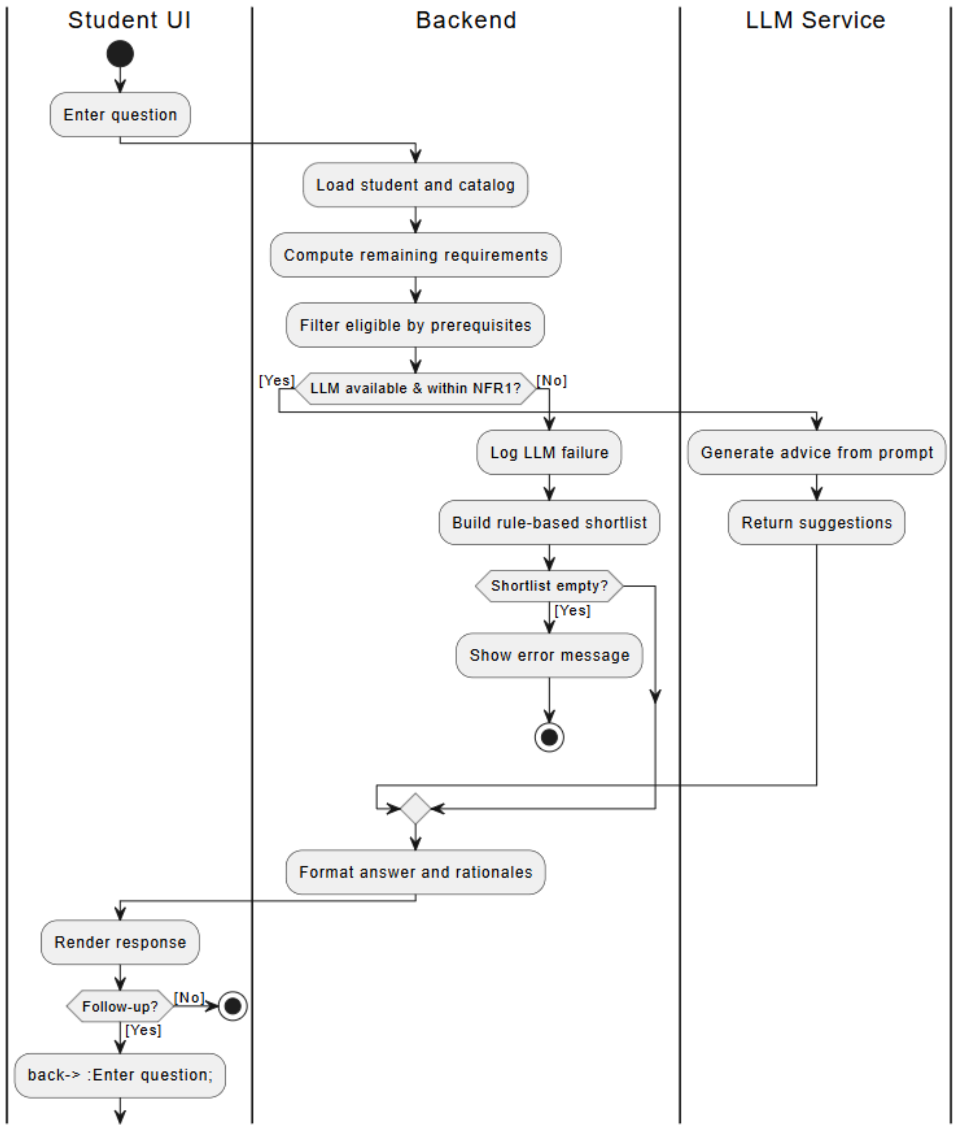
Use Case 2: View Degree Progress

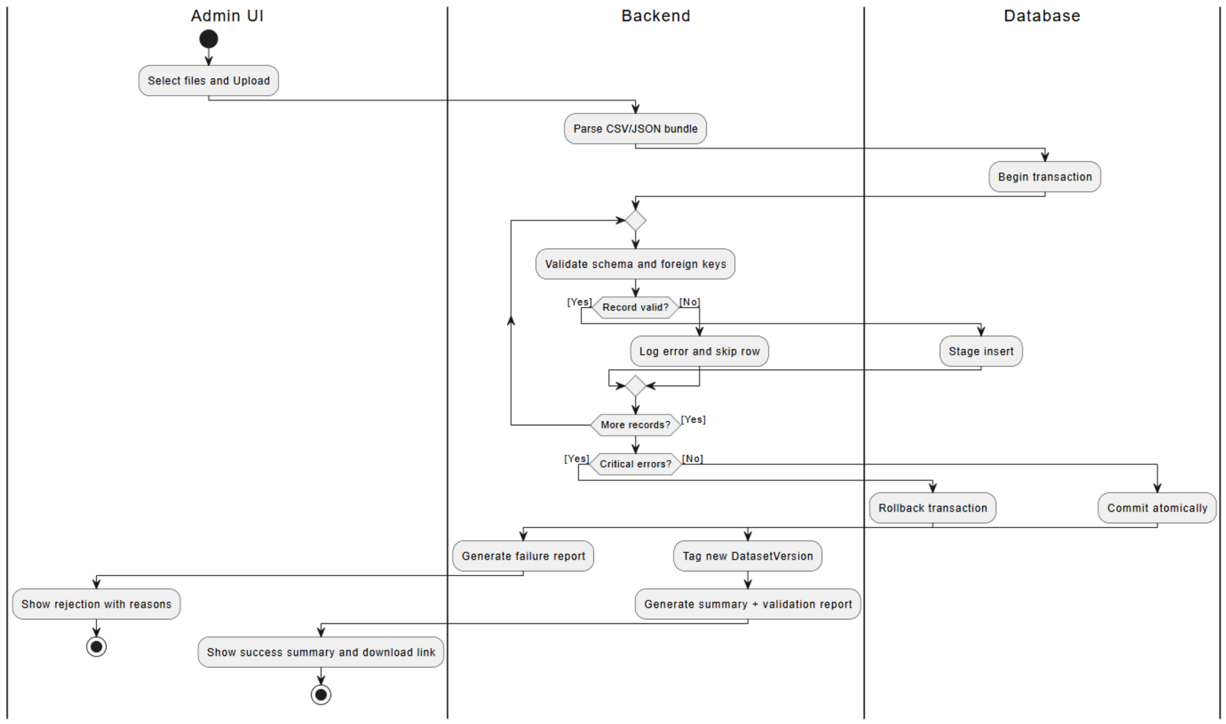
- Actors: Student (Primary), System
- Preconditions: Student record exists; dataset loaded
- Main Flow:
    1. Student requests progress view
    2. System calculates completion status
    3. System renders interactive visualization
    4. Student interacts with visualization elements

Alternatives:

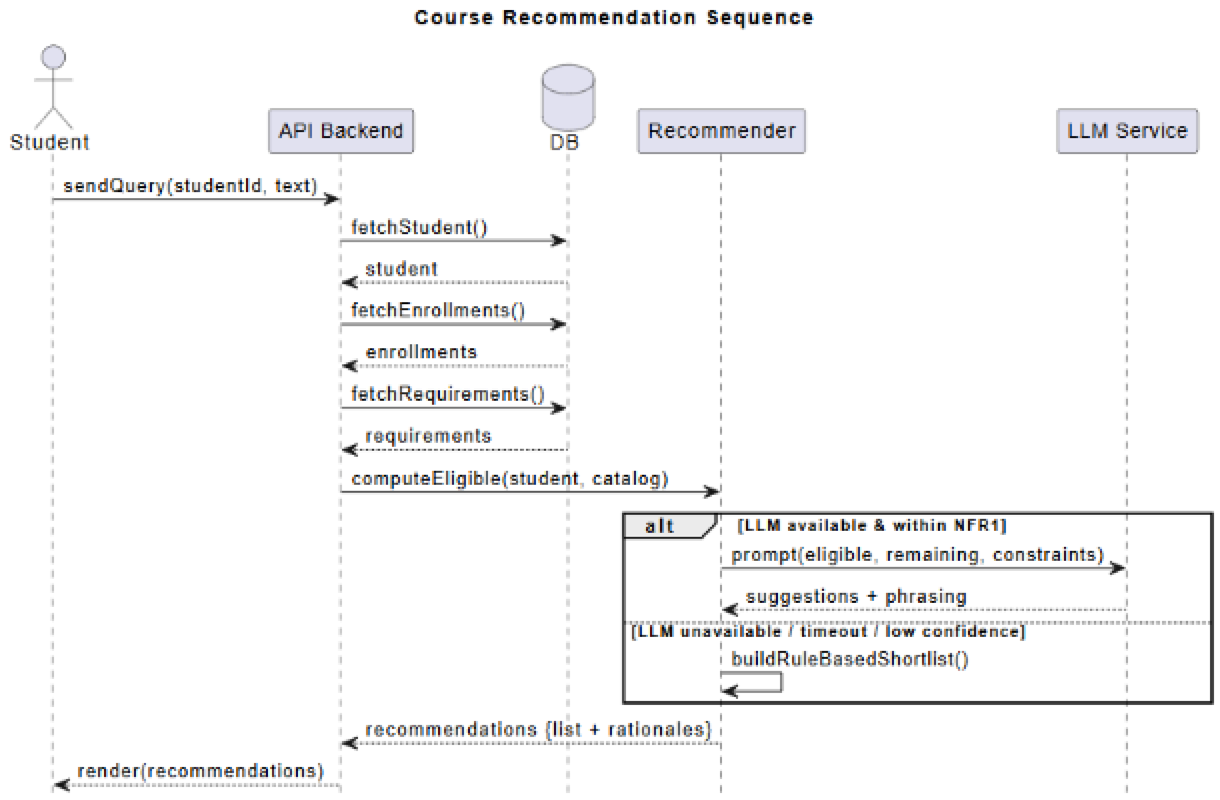
- Missing/ambiguous data → system flags gaps and advises contacting Admin.

5.3 Activity Diagrams


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;



5.4 Sequence 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)

5.5 Navigational Paths

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:

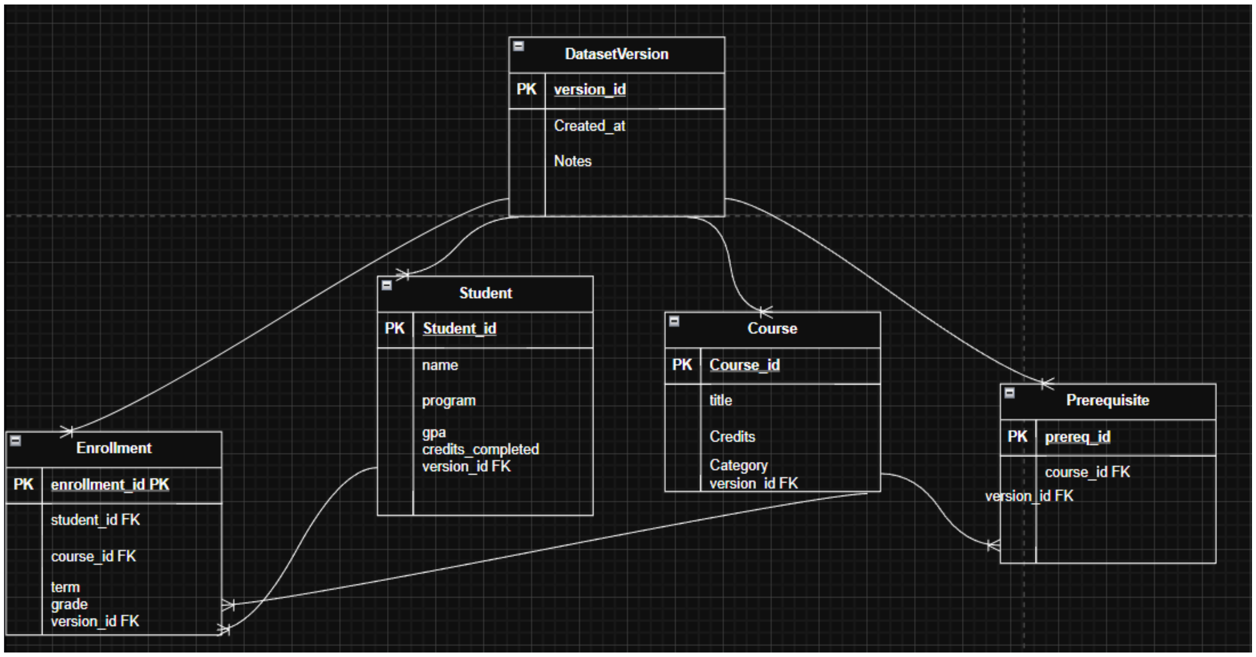
- Entry: Admin Login → Session Creation → Admin Dashboard
- Data Management: Upload Dataset → Validation → Review Report → Activate/Reject
- Monitoring: View Logs → Filter by Date/Type → Export Reports

5.6 Wireframes

```
[Top Bar]  IAAS | Advisor Chat | Progress | (Data Upload*)      [Dataset vX.Y]


┌────────────────────────────────┬────────────────────────────────┐
│ Chat History (scroll)          │ Side Panel                     │
│ - Bot: SE230 - satisfies R3    │ Student: [ Ahmed ▼ ]           │
│ - You: Why SE230?              │ Quick Prompts:                 │
│ - Bot: Unlocks SE340 next…     │ [What next?] [Why this?]       │
│                                │ [Alternatives]                 │
│                                │                                │
│                                │ Notice/Banner area             │
├────────────────────────────────┴────────────────────────────────┤
│ [ Type your question…                        ] [ Send ]         │
└─────────────────────────────────────────────────────────────────┘
```

5.7 Entity Relationship Model



6. Traceability to Objectives

The Requirements Traceability Matrix ensures all project objectives are addressed by specific requirements:

Table 3: Requirements Traceability Matrix (RTM)

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

			NFR3: 95% uptime requirement
OBJ5	Maintainability and auditability	FR7, FR8, NFR4	FR7: Audit trail of all interactions FR8: PDF export for documentation NFR4: Reproducible testing through versioning

This traceability matrix demonstrates complete coverage of all project objectives through functional and non-functional requirements. Each objective is supported by multiple requirements ensuring robust implementation.

7. Assumptions, Dependencies & Glossary

7.1 Assumptions

- Users have stable internet access with minimum 1Mbps bandwidth and modern browsers (released after 2020)
- The university provides server infrastructure with at least 8GB RAM for hosting the application
- Pre-generated synthetic data files adequately represent real student patterns
- OpenAI API maintains current pricing model throughout project timeline
- Course prerequisites follow a directed acyclic graph structure without circular dependencies
- Students are familiar with basic web navigation and chat interfaces
- The Software Engineering curriculum remains relatively stable during development

7.2 Dependencies

- OpenAI API access with valid API key and $20 monthly budget allocation
- Internet connectivity for API calls to OpenAI services
- Python 3.9+ runtime environment with pip package manager
- SQLite 3.35+ database system
- Modern web browser supporting ES6 JavaScript
- Network connectivity for web access and API communications
- Git for version control and collaboration

- Environment variable management for secure API key storage

7.3 Glossary

- **API (Application Programming Interface):** Set of protocols for building software applications
- **API Quota:** Usage limit imposed to control costs, set at $20/month for OpenAI services
- **bcrypt:** Cryptographic hashing function designed for password storage
- **CSV (Comma-Separated Values):** File format for tabular data storage
- **D3.js:** JavaScript library for producing dynamic data visualizations
- **FastAPI:** Modern Python web framework for building APIs
- **GPT-4o-mini:** OpenAI's cost-efficient language model, 60% cheaper than GPT-3.5 with superior accuracy
- **JSON (JavaScript Object Notation):** Lightweight data interchange format
- **LLM (Large Language Model):** AI model trained on vast text data for natural language understanding
- **OpenAI API:** Cloud-based service providing access to GPT models for natural language processing
- **ORM (Object-Relational Mapping):** Technique for converting data between incompatible type systems
- **Prerequisite Chain:** Sequence of courses that must be completed in specific order
- **Quota Monitoring:** Server-side counters tracking API tokens vs. monthly cap, with alerts at 50%, 80%, 90%
- **REST (Representational State Transfer):** Architectural style for distributed hypermedia systems
- **Rule-based Fallback:** Deterministic algorithm using if-then rules activated when API quota reaches 90%
- **SQLAlchemy:** Python SQL toolkit and Object-Relational Mapping library
- **Synthetic Data:** Artificially generated data preserving statistical properties without real information
- **Token:** Basic unit of text processed by LLMs; GPT-4o-mini costs $0.15/1M input, $0.60/1M output tokens
- **3NF (Third Normal Form):** Database normalization form eliminating transitive dependencies

References

[1] R. Agrawal and H. Patel, "Automated Degree Audit and Prerequisite Checking Systems in Higher Education," *IEEE Transactions on Education*, vol. 64, no. 3, pp. 234-245, Aug. 2021, doi: 10.1109/TE.2021.3049571.

[2] L. Zhang, Y. Chen, and M. Kumar, "Graph-Based Prerequisite Chain Modeling for Curriculum Planning," in *2020 IEEE International Conference on Data Mining (ICDM)*, Sorrento, Italy, 2020, pp. 812-821, doi: 10.1109/ICDM50108.2020.00089.

[3] S. Johnson and K. Williams, "Privacy-Preserving Synthetic Data Generation for Educational Analytics," *IEEE Security & Privacy*, vol. 19, no. 4, pp. 45-53, July/Aug. 2021, doi: 10.1109/MSEC.2021.3074259.

[4] T. Martinez and A. Davis, "Constraint-Based Course Recommendation Systems Using Integer Programming," *IEEE Access*, vol. 8, pp. 112456-112467, 2020, doi: 10.1109/ACCESS.2020.3002819.

[5] J. Park, N. Ahmed, and R. Singh, "Reproducible Dataset Versioning for Academic Research Systems," in *2022 IEEE Conference on Big Data*, Osaka, Japan, 2022, pp. 1892-1901, doi: 10.1109/BigData55660.2022.10020571.