Northeastern University Student Assistance Platform (NEU-SA)

Team Members:

Dharun Karthick Ramaraj - ramaraj.d@northeastern.edu
Linata Rajendra Deshmukh - deshmukh.li@northeastern.edu
Nikhil Godalla - godalla.n@northeastern.edu

Project Proposal Video :
https://drive.google.com/file/d/1L5a91dWODh3c-679K3531ZQin7WYmVCi/view?usp=sharing
Github project (Timelines) : https://github.com/orgs/BigDataIA-Fall2024-Team-5/projects/12
Addresed feedback : NEU-Assistance Project propsal (Page 17 in codelabs)

Introduction

In simple words :

A multi-agent chatbot for a Northeastern University graduate student with which a student can chat on planning courses for upcoming semester and asking general information about the college.

Background :

Students have trouble finding the right course for them. We have program restrictions(prerequisites), course availability(if it is being offered for the semester), seat allocation(total seats), class timings, campus location(classes from other location)
Our project aims to address these difficulties by suggesting the course for the student which aligns their interest

Objective :

To develop a user friendly application which answers general queries and suggest courses based on the enrolled program and completed courses along with CRN, Professor, Timing Details.

Project Overview

Scope:

Application Focus : The platform will currently be tailored for Graduate students enrolled in Northeastern University

Data Sources : Northeastern University websites (check methodologies for links)

Technologies:

- Pipelines (Airflow)
- Web Scraping (Selenium and BeautifulSoap)
- Embedding generation (Nvidia embedding models)
- Storage (Snowflake & Pinecone)
- LLM (Openai Models)
- Transcript pdf processing (Amazon Textraxt)
- Multi Agents (Langgraph)
- Backed (Fast Api)
- Frontend (Streamlit)
- Deployment (Docker, GCP VM Instances)

Deliverables:

- Multi user application with JWT authentication
- Frontend and Backend deployed in GCP
- General information and Course suggestion chatbot

Stakeholders:

- Primary users: Graduate students in Northeastern University
- Secondary users: University staff and faculty seeking a platform for student assistance.

Problem Statement

Current Challenges :

- Information scattered on different websites, making it hard for students to find relevant information
- No current system provides course suggestion and let you know about the eligible courses for student

Opportunities

- Improve student experience in choosing their courses
- Upcoming semesters course information available
- A scalable solution that can be extended to other institutions

Architecture Diagrams

Agent Architecture Diagram - NEU-Assistance Project propsal

User Flow Diagram - NEU-Assistance Project propsal

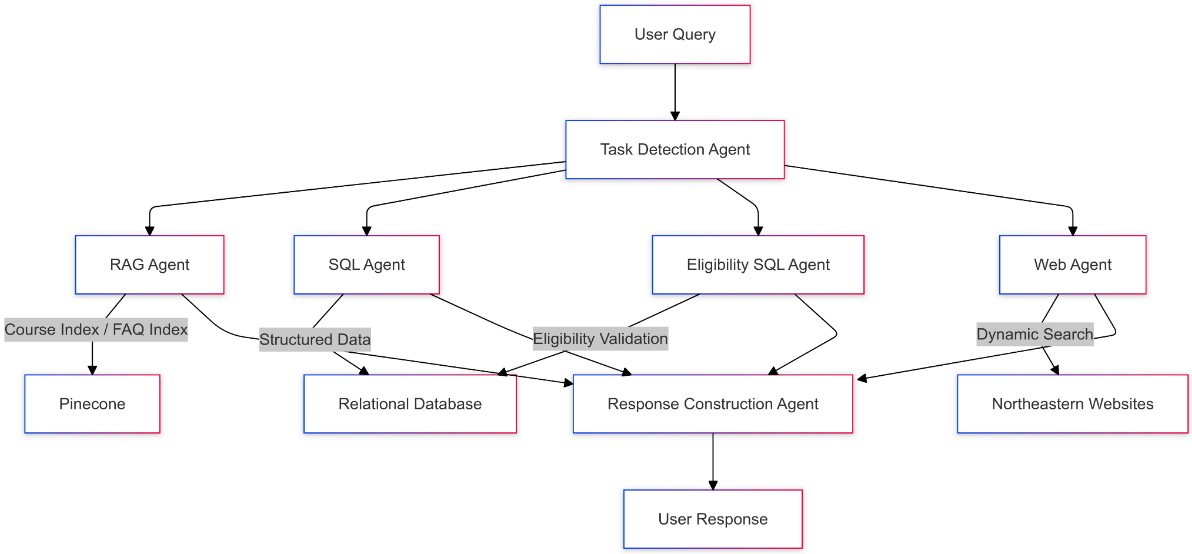
Airflow Pipeline Diagram - NEU-Assistance Project propsal

Technologies and Tools:

- **Poetry** - Required library and dependency management
- **Selenium, BeautifulSoup** - Scraping data from the data sources
- **Snowflake** - Course Catalog, Classes Table, Program Requirements, Core Requirements, Elective Requirements, User Profile, User Eligibility Tables
- **Pinecone** - Course description and prerequisites as chunks and metadata as course_code, title , credits and other index with other FAQs
- **Amazon Textract** - Preprocessing the user transcripts
- **Amazon S3** - User Transcripts
- **Nvidia Embedding model** - To embed the chunks
- **Langgraph** - Framework for our agentic architecture
- **OpenAI models** - As our main LLM
- **Airflow** - To schedule the pipelines

- **Fast Api** - OpenAI models for question-answering, combining relevant data from embeddings and external resources.
- **Streamlit** - Frontend
- **Docker** - Building image before deploying
- **Google Cloud Platform** - VM instances to deploy our frontend and backend

AI Agents and Tools:



Task Detection Agent (Main chat node)

Purpose - Serves as the main orchestrator, analyzing user queries to determine their intent and routing the request to the appropriate agents.
Responsibilities:

- Classifies queries into types: semantic, structured, or hybrid.
- Identifies the appropriate index or source (e.g course data, FAQs, or web search) for information retrieval.
- **Semantic Query**: Focused on conceptual matches (e.g "What is a big data course?"). Then it goes to RAG Agent
- **Structured Query**: Requires precise data (e.g "What are the timings for INFO 7245?"). Then we go to SQL Agent
- **Hybrid Query**: Combines semantic and structured aspects (e.g "What courses on big data are available, and when can I take them?").

Output: Provides the query type, extracted keywords, and the appropriate agent or source to route the query.

RAG Agent (Retrieval-Augmented Generation)

Purpose - Handles semantic search and retrieval from indexed data stored in Pinecone, regardless of the type of data (courses, FAQs, etc.).
Responsibilities:

- Retrieves data from the appropriate Pinecone index based on query intent:
    - **Course Index**: For course-related queries (e.g., "big data courses").
    - **FAQ Index**: For general questions about registration, policies, or processes.
- Combines retrieved information with additional metadata to construct a comprehensive response.
- Uses AI models to rank and refine the most relevant results.

Output: Returns the top matching chunks and metadata for further processing.

SQL Agent

Purpose - Retrieves structured data from relational databases (e.g., Snowflake) for term-specific details, schedules, instructors, and campus information.

Responsibilities:

- Executes SQL queries to fetch:
    - Course registration data.
    - Class timings, instructors, and room locations.
- Ensures data is precise and matches the user's term or campus preferences.
- Executes SQL queries in Snowflake to fetch precise details.

Eligibility SQL Agent

Purpose - Validates the user's eligibility for a course or program based on prerequisites, completed credits, and program-specific rules.

Responsibilities:

- Cross-references user profile data (e.g., completed courses, GPA, credits) with course prerequisites and program requirements.
- Checks constraints like credit limits and core course completion.
- Provides a detailed reason if the user is ineligible (e.g., "Prerequisite INFO 6205 not completed").

Output: Returns eligibility status (true or false) and supporting details.

Web Agent

Purpose: Performs web scraping and targeted searches on Northeastern University websites to answer specific user questions.

Responsibilities:

- Searches university domains (e.g., northeastern.edu) for relevant pages.
- Answers queries like:
    - "How do I apply for financial aid at Northeastern University?"

Output: Returns relevant information from the web, such as links, summarized content, or direct answers.

Response Construction Agent

Purpose: Aggregates outputs from various agents and formats the final response for the user.

Responsibilities:

- Combines data from RAG, SQL, Eligibility SQL, and Web Agents.
- Ensures clarity, completeness, and user-friendliness in the response.
- Handles multi-step or follow-up queries seamlessly.

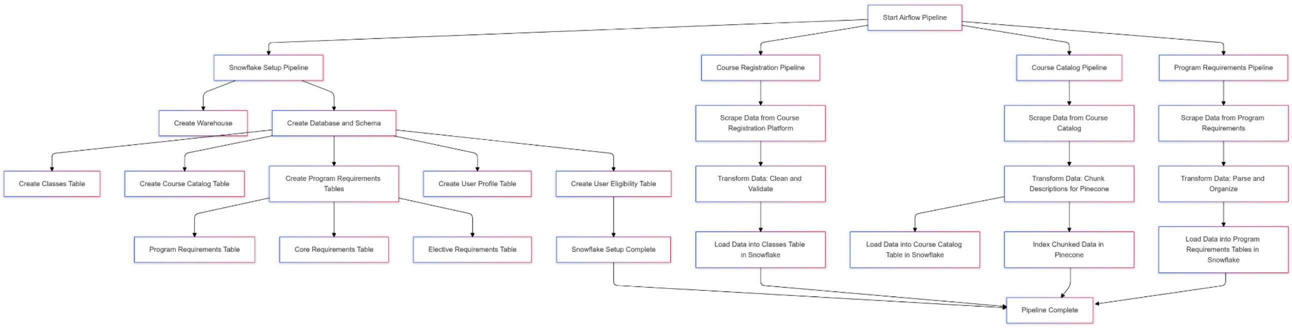
Output: Returns a well formatted response to the user.

Methodology (Setup perspective)

Check the data sources here - NEU-Assistance Project propsal

Course Registration Page - To get term-specific course registration details such as CRNs, class timings, and instructors.
Course Catalog - To get course descriptions, prerequisites, credit hours, and subject codes.
Program Requirements - To get program-specific requirements, including core courses, elective constraints, and credit hour requirements.

Airflow Pipelines:


Start Airflow Pipeline
Snowflake Setup Pipeline
Create Warehouse
Create Database and Schema
Create Classes Table
Create Course Catalog Table
Create Program Requirements Tables
Create User Profile Table
Create User Eligibility Table
Program Requirements Table
Core Requirements Table
Elective Requirements Table
Snowflake Setup Complete
Course Registration Pipeline
Scrape Data from Course Registration Platform
Transform Data: Clean and Validate
Load Data into Classes Table in Snowflake
Course Catalog Pipeline
Scrape Data from Course Catalog
Transform Data: Chunk Descriptions for Pinecone
Load Data into Course Catalog Table in Snowflake
Index Chunked Data in Pinecone
Program Requirements Pipeline
Scrape Data from Program Requirements
Transform Data: Parse and Organize
Load Data into Program Requirements Tables in Snowflake
Pipeline Complete

Course Registration Pipeline - Scrape and ingest data from the course registration platform into the ClassesTable in Snowflake.

Course Catalog Pipeline - Scrape and store data from the course catalog into both Snowflake and Pinecone.

Steps:

1. Extract:
    - Scrape course information including descriptions, prerequisites, and credit hours.
2. Transform:
    - Chunk descriptions for Pinecone indexing.
    - Validate and clean data for Snowflake insertion.
3. Load:
    - Store detailed course data in the CourseCatalogTable in Snowflake.
    - Index chunked descriptions with metadata in Pinecone.

Program Requirements Pipeline - Scrape and store data about program requirements into Snowflake (ProgramRequirementsTable, CoreRequirementsTable, and ElectiveRequirementsTable).

Snowflake setup Pipeline - Creates all the required tables

Pipeline Scheduling

Course Registration Pipeline: Daily, to account for new or updated class schedules during course registration month.

Course Catalog Pipeline: Semesterly

Program Requirements Pipeline: Semesterly

Snowflake setup Pipeline : One time setup

Data Storage Design

SNOWFLAKE

User Profile Table

Column	Type	Description
user_id	INT	Unique identifier for the user
username	VARCHAR	User’s username
password	VARCHAR	Hashed password for authentication
campus	VARCHAR	Campus where the user is located
program_id	VARCHAR	Program the user is enrolled in
completed_courses	VARCHAR	Comma-separated list of

		completed courses
completed_credits	INT	Total credit hours the user has completed
gpa	FLOAT	User's current GPA

User Eligibility Table (We process it with all the other required information)

Column	Type	Description
user_id	INT	Unique identifier for the user
course_code	VARCHAR	Course code
eligible	BOOLEAN	Whether the user is eligible for the course
total_credits	INT	Total credit hours user has completed
max_credits	INT	Maximum allowed credits in the program
reason	VARCHAR	

Classes Table

Column	Type	Description
term	VARCHAR	Term (e.g Spring 2025)
course_code	VARCHAR	Course code
crn	VARCHAR	Course registration number
instructor	VARCHAR	Instructor name
timing	VARCHAR	Class timing
room	VARCHAR	Room and building
campus	VARCHAR	Campus location
schedule_type	VARCHAR	lecture
instructional_method	VARCHAR	Traditional or online

Course Catalog Table

Column	Type	Description
course_code	VARCHAR	Course code (e.g INFO 7245)
title	VARCHAR	Course title
description	TEXT	Detailed course description
prerequisites	VARCHAR	Prerequisites (e.g INFO 6205)
credit_hours	INT	Credit hours
subject_code	VARCHAR	Subject code (e.g INFO)

Program Requirements Table

Column	Type	Description
program_id	VARCHAR	Unique identifier for the program (e.g MSIS, DAMG)
max_credit_hours	INT	Total maximum credits required for the program
min_gpa	FLOAT	Minimum GPA required for program completion
core_credit_req	INT	Total credits required for core courses
elective_credit_req	INT	Total elective credits required
subject_credit_req	INT	Credits required from a specific subject area, if any

Core Requirements Table

Column	Type	Description
program_id	VARCHAR	Unique identifier for the program (e.g MSIS, DAMG)
course_code	VARCHAR	Course code of the core course (e.g INFO 5100)
course_title	VARCHAR	Title of the core course (e.g

		Application Engineering Development)
credit_hours	INT	TCredit hours for the core course

Elective Requirements Table

Column	Type	Description
program_id	VARCHAR	Unique identifier for the program (e.g MSIS, DAMG)
subject_code	VARCHAR	Subject code allowed for electives (e.g., INFO, DAMG)
excluded_courses	VARCHAR	Specific courses excluded, if any

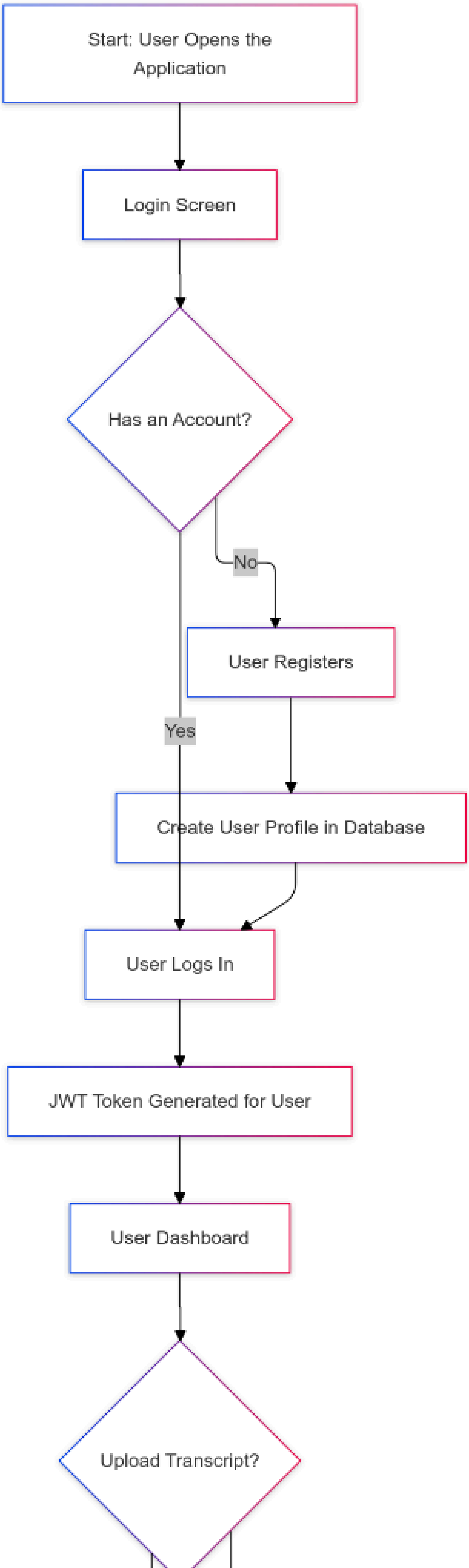
PINECONE

The Course Catalog data will also be indexed in Pinecone for semantic search

Chunking Strategy:
Divide course descriptions into chunks.
Along with meta data like

```
{
  "course_code": "INFO 7245",
  "title": "Big-Data Systems and Intelligence Analytics",
  "prerequisites": "INFO 6205"
}
```

Methodology (User Flow perspective)


Start: User Opens the Application
Login Screen
Has an Account?
No
Yes
User Registers
Create User Profile in Database
User Logs In
JWT Token Generated for User
User Dashboard
Upload Transcript?

Amazon Textract Processing

The transcript is processed using Amazon Textract to extract relevant information, such as completed courses, grades, and credits.

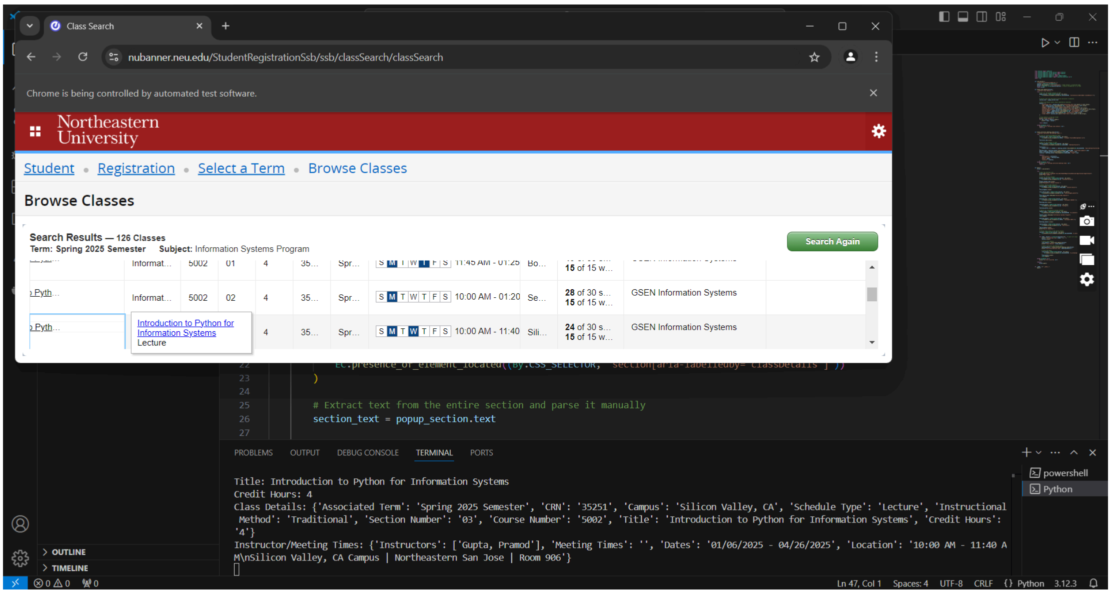
Eligibility Table Update

The extracted data is used to populate the User Eligibility Table, enabling personalized course recommendations and eligibility checks.

Proof of Concept

Web Scraping

Web scraping for other websites is simple as the links are not static but for the course registration page the web link is static so we use selenium and python to navigate to get the relevant information and extract it. We tested it with a code you can understand how it works from the below snippet



Chrome is being controlled by automated test software.

Browse Classes

Search Results — 126 Classes
Term: Spring 2025 Semester Subject: Information Systems Program

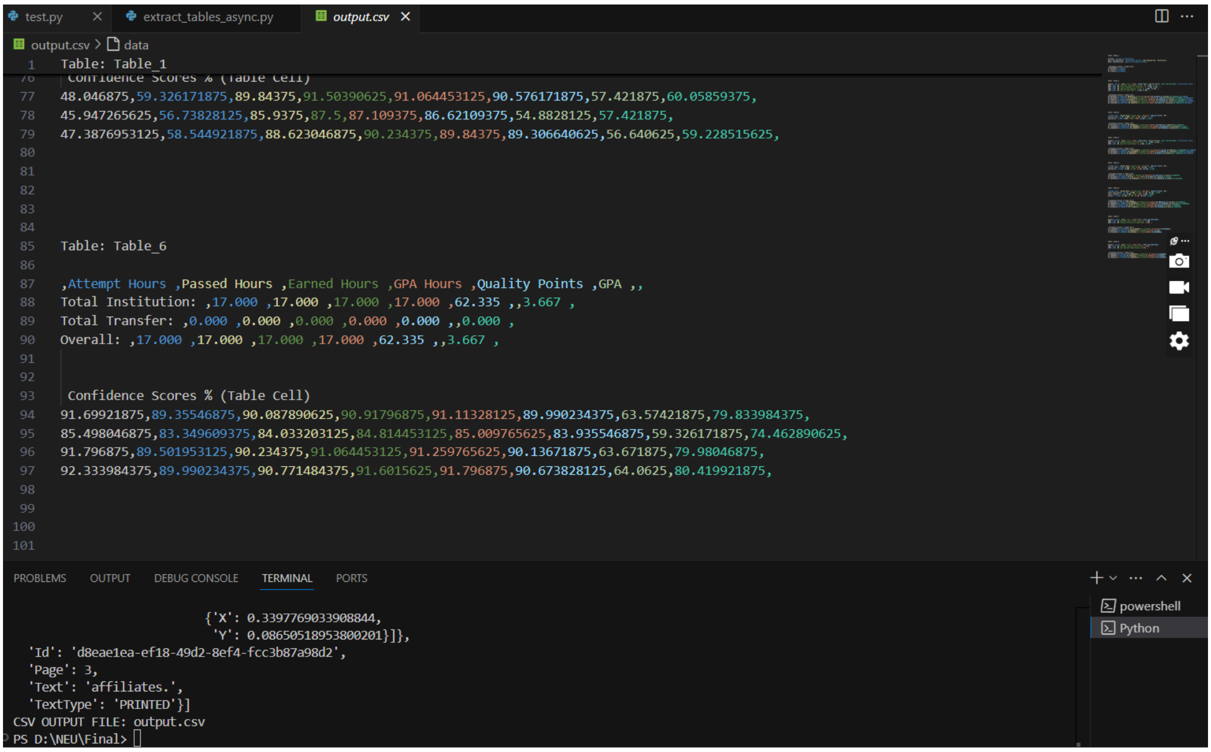
	Informat…	5002	01	4	35…	Spr…	S M T W T F S 11:45 AM - 01:25	Bo…	15 of 15 w…	
› Pyth…	Informat…	5002	02	4	35…	Spr…	S M T W T F S 10:00 AM - 01:20	Se…	28 of 30 s… 15 of 15 w…	GSEN Information Systems
› Pyth…	Introduction to Python for Information Systems Lecture			4	35…	Spr…	S M T W T F S 10:00 AM - 11:40	Sili…	24 of 30 s… 15 of 15 w…	GSEN Information Systems

```
        EC.presence_of_element_located((By.CSS_SELECTOR, 'section[aria-labelledby="classDetails"]'))
    )

    # Extract text from the entire section and parse it manually
    section_text = popup_section.text

```

```
Title: Introduction to Python for Information Systems
Credit Hours: 4
Class Details: {'Associated Term': 'Spring 2025 Semester', 'CRN': '35251', 'Campus': 'Silicon Valley, CA', 'Schedule Type': 'Lecture', 'Instructional
 Method': 'Traditional', 'Section Number': '03', 'Course Number': '5002', 'Title': 'Introduction to Python for Information Systems', 'Credit Hours':
'4'}
Instructor/Meeting Times: {'Instructors': ['Gupta, Pramod'], 'Meeting Times': '', 'Dates': '01/06/2025 - 04/26/2025', 'Location': '10:00 AM - 11:40 A
M\nSilicon Valley, CA Campus | Northeastern San Jose | Room 906'}
```

Transcript Processing

Downloaded unofficial transcript from northeastern university and used Amazon Textract to extract the tables. It works fine. Below is the snippet of the extracted data



```
Table: Table_1
 Confidence Scores % (Table Cell)
48.046875,59.326171875,89.84375,91.50390625,91.064453125,90.576171875,57.421875,60.05859375,
45.947265625,56.73828125,85.9375,87.5,87.109375,86.62109375,54.8828125,57.421875,
47.3876953125,58.544921875,88.623046875,90.234375,89.84375,89.306640625,56.640625,59.228515625,




Table: Table_6

,Attempt Hours ,Passed Hours ,Earned Hours ,GPA Hours ,Quality Points ,GPA ,,
Total Institution: ,17.000 ,17.000 ,17.000 ,17.000 ,62.335 ,,3.667 ,
Total Transfer: ,0.000 ,0.000 ,0.000 ,0.000 ,0.000 ,,0.000 ,
Overall: ,17.000 ,17.000 ,17.000 ,17.000 ,62.335 ,,3.667 ,


 Confidence Scores % (Table Cell)
91.69921875,89.35546875,90.087890625,90.91796875,91.11328125,89.990234375,63.57421875,79.833984375,
85.498046875,83.349609375,84.033203125,84.814453125,85.009765625,83.935546875,59.326171875,74.462890625,
91.796875,89.501953125,90.234375,91.064453125,91.259765625,90.13671875,63.671875,79.98046875,
92.333984375,89.990234375,90.771484375,91.6015625,91.796875,90.673828125,64.0625,80.419921875,
```

```
                        {'X': 0.3397769033908844,
                         'Y': 0.08650518953800201}]},
  'Id': 'd8eae1ea-ef18-49d2-8ef4-fcc3b87a98d2',
  'Page': 3,
  'Text': 'affiliates.',
  'TextType': 'PRINTED'}]
CSV OUTPUT FILE: output.csv
PS D:\NEU\Final> 
```

Project Plan and Timeline

Phase 1: Initial Setup and Data Ingestion (Nov 26 - Nov 30)

Goals:

- Set up the backend infrastructure and pipelines for data ingestion.
- Scrape and preprocess data from Northeastern University websites.

Tasks:

+ Configure **Airflow Pipelines**
    - Snowflake Setup Pipeline: Create warehouse, database, and all required tables.
    - Course Registration Pipeline: Scrape term-specific course registration data into the Classes Table.
    - Course Catalog Pipeline: Scrape and chunk course descriptions, store in Course Catalog Table (Snowflake) and Pinecone.

        - **Program Requirements Pipeline**: Scrape program requirements and store them in Program Requirements, Core Requirements, and Elective Requirements tables.
- Test pipelines for consistency and error handling.

Deliverables:

- Functional Airflow pipelines.
- Data populated in Snowflake and indexed in Pinecone.

Phase 2: Backend API Development (Dec 1 - Dec 3)

Goals:

- Build the backend APIs using FastAPI to query and interact with the processed data.
- Integrate agents for query resolution.

Tasks:

- Develop APIs for
    - Course search and retrieval from Snowflake.
    - Eligibility checks based on the User Eligibility Table.
    - FAQ and general query handling using Pinecone.
- Implement and test Agents:
    - **Task Detection Agent**: Orchestrate query routing.
    - **RAG Agent**: Handle semantic search in Pinecone.
    - **SQL Agent**: Fetch structured data from Snowflake.
    - **Eligibility SQL Agent**: Validate user eligibility.
    - **Web Agent**: Search Northeastern websites for additional information.
    - **Response Construction Agent** to aggregate responses from multiple agents.

Deliverables:

- Fully tested backend with all agents integrated.
- Backend capable of handling course queries, eligibility checks, and general FAQs.

Phase 3: Frontend Development (Dec 4 - Dec 6)

Goals:

- Create a user-friendly frontend using Streamlit.
- Enable interactions with the backend APIs.

Tasks:

- Design and implement
    - Login/Registration Page (JWT authentication and user profile creation)
    - Transcript Upload Page (Integration with Amazon Textract to process transcripts)
    - Query Interface (Chatbot-style UI for querying courses and general FAQs)
    - Dashboard (Display user-specific course suggestions, eligibility, and CRN/professor details)

- Connect the frontend with the backend APIs.
- Test the end-to-end flow for data consistency and error handling.

Deliverables:

- Functional frontend deployed locally with backend integration.

Phase 4: Deployment and Testing (Dec 7 - Dec 9)

Goals:

- Deploy the system to Google Cloud Platform (GCP).
- Perform testing

Tasks:

- Containerize the frontend and backend using **Docker**.
- Deploy containers on **GCP VM Instances**.
- Functional testing to validate API responses and frontend behavior.

Deliverables:

- Deployed application accessible via GCP.
- Testing report with identified issues resolved.

Phase 5: Finalization and Documentation (Dec 10 - Dec 12)

Goals:

- Prepare the final deliverables, including documentation and a project demo.

Tasks:

- Create user (User guide) and developer documentation (setup guide, including pipeline configuration and deployment steps)
- Record the project introduction video.
- Address any feedback from stakeholders or testers.
- Conduct the final project demo with all features.

Deliverables:

- Finalized platform with complete functionality.
- Comprehensive documentation.
- Project introduction video and demo presentation.

Resources and Team

Team Members:

Dharun Karthick Ramaraj
Linata Rajendra Deshmukh
Nikhil Godalla

Project-work plan can be found in the github projects - https://github.com/orgs/BigDataIA-Fall2024-Team-5/projects/12

Risks and Mitigation Strategies

Risks : Errors in extracting data from transcripts using Amazon Textract

Mitigation : Add manual verification options for transcripts. Implement fallback mechanisms for users to input data manually if parsing fails

Risks : Unable to get a properly configured webagent

Mitigation : Use TAVILY API to get top 3-4 matches for the related query in northeastern websites

Risks : Unable to get data in the required format

Mitigation : Make use of an LLM to get it into the format

Expected Outcomes and Benefits

Expected Outcomes

1. Multi-Agent Chatbot for Assistance
    - A user-friendly chatbot that answers general queries and provides course recommendations for Northeastern University graduate students.
    - Supports both semantic queries (e.g course suggestions) and structured queries (e.g course timings, CRNs).

2. Automated Data Processing Pipelines
    - Efficient scraping and preprocessing of course-related and program-related data using Airflow
    - Structured storage in Snowflake for efficient querying and retrieval
3. User Personalization
    - Integration of Amazon Textract to process transcripts, enabling personalized course suggestions based on completed courses and GPA
4. Scalable and Extendable System
    - A modular architecture that allows scaling to other institutions by modifying data scraping scripts.
    - Integration of FAQ-based queries and eligibility checks for comprehensive assistance.
5. Deployment on GCP
    - Secure and accessible platform hosted on Google Cloud Platform with frontend (Streamlit) and backend (FastAPI) deployment

Benefits

1. Improved User Experience
    - Reduces the time and effort students spend searching for information across multiple platforms
    - Provides tailored course recommendations aligned with program requirements and completed courses
2. Centralized Information Access
    - Combines data from multiple sources (e.g., course registration, catalog, program requirements) into a single, intuitive interface
3. Automation and Efficiency
    - Automates data ingestion, preprocessing, and indexing workflows, minimizing manual intervention and ensuring data consistency
4. Personalized Course Planning
    - Offers recommendations based on user profiles, enabling students to make informed decisions about their course schedules
5. Scalable Solution

Conclusion

The Northeastern University Student Assistance Platform (NEU-SA) streamlines course planning and general query resolution for graduate students by combining a multi-agent chatbot with automated data pipelines. By integrating technologies like Amazon Textract, Pinecone, and OpenAI models, the platform provides personalized course recommendations and centralized access to information. Its design ensures scalability, making it a valuable tool for improving student experiences and potentially extending to other institutions. NEU-SA simplifies decision-making and lets students focus on their academic success.

Addressing Feedback

Feedback 1: Detail how students are informed about courses across departments, Enable users to upload transcripts to assess prerequisites.
-> When a user submits a transcript, it is processed using Amazon Textract, and the extracted data is stored in Snowflake. This data is compared with eligibility criteria, keeping prerequisites in mind, to recommend suitable subjects. Courses from other departments are also accounted for, ensuring they align with credit requirements.

Feedback 2: Specify agents and their tools.
-> We have added AI Agents and Tools slide to address agent specific information in detail.

Feedback 3: News Scraping: Justify its relevance to course selection.
-> The news scraping feature was removed after feedback from the professor as it did not add value to the course registration process.

Feedback 4: S3 Storage: What textual data is being stored, and how?
-> S3 stores transcript pdf details, we are not storing any structured data in S3. All the structured data will be stored in Snowflake for more information we have added the slide Data Storage Design. Kindly refer the information

Feedback 5: NVIDIA Embeddings: Explain their choice for the project.
-> We have used the same model before, and it works well with the data, so we planned to use it again.

Feedback 6: Include a feature to show course prerequisites to help students plan their schedules better.
-> Yes, it will be included while suggesting the subjects to the students.

Feedback 7: The team discussed the feature about providing assistance to the end users (i.e., Northeaster students) about course registration. Northeastern University follows a process where in students who are closer to graduation are more likely to get higher preference over junior students when it comes to course registration. It would be nice if the application took the user's current semester into consideration when recommending courses, and also provided similar or alternate courses in case seats are unavailable for the user's preferred course. Overall, it is a great application which would turn out to be very helpful for new graduate students.
->We’re planning to show seat information while giving the details about the subject, but during busy times like course registration, the Northeastern site gets overloaded because so many people are using it. This makes it really hard to get accurate seat info in real-time. It’s possible to do, but with our project deadline, we are focusing on the main functionality. This is an awesome idea. If we have time we will definitely look into it.

Reference

Apache Airflow Documentation: https://airflow.apache.org/docs
FastAPI Documentation: https://fastapi.tiangolo.com
Snowflake Documentation: https://docs.snowflake.com
AWS S3 - Getting Started: https://aws.amazon.com/s3/getting-started
Pinecone Vector Database Documentation: https://docs.pinecone.io
OpenAI API Documentation: https://platform.openai.com/docs
Docker Documentation: https://docs.docker.com/
GitHub Repository Best Practices: https://guides.github.com/features/wikis/
Streamlit Documentation: https://docs.streamlit.io/
Amazon Textract: https://docs.aws.amazon.com/textract/

Main Data Sources

- Course Registration platform: https://nubanner.neu.edu/StudentRegistrationSsb/ssb/registration
- Course Catalog: https://catalog.northeastern.edu/course-descriptions/
- Program Requirements: https://catalog.northeastern.edu/graduate/engineering/multidisciplinary/information-systems-msis/#programrequirementstext

Other Data Sources

- Course Registration: https://coe.northeastern.edu/academics-experiential-learning/graduate-school-of-engineering/graduate-student-services/course-registration/
- New Student Information: https://coe.northeastern.edu/academics-experiential-learning/graduate-school-of-engineering/graduate-student-services/new-student-information/
- Graduate Resources: https://coe.northeastern.edu/academics-experiential-learning/graduate-school-of-engineering/graduate-student-services/university-graduate-resources/
- FAQ for Students: https://coe.northeastern.edu/academics-experiential-learning/graduate-school-of-engineering/graduate-student-services/faqs-for-newly-admitted-and-current-students/
- Graduation Commencement:https://coe.northeastern.edu/academics-experiential-learning/graduate-school-of-engineering/graduate-student-services/graduation-commencement/

Mermaid code for Agents

```
---
config:
  layout: fixed
---
```

```
flowchart TD
    User["User Query"] --> TaskDetection["Task Detection Agent"]
    TaskDetection --> RAG["RAG Agent"] & SQL["SQL Agent"] & Eligibility["Eligibility SQL Agent"] & Web["Web Agent"]
    RAG -- Course Index / FAQ Index --> Pinecone["Pinecone"]
    SQL -- Structured Data --> Database["Relational Database"]
    Eligibility -- Eligibility Validation --> Database
    Web -- Dynamic Search --> Websites["Northeastern Websites"]
    RAG --> Response["Response Construction Agent"]
    SQL --> Response
    Eligibility --> Response
    Web --> Response
    Response --> UserResponse["User Response"]
```

Mermaid code for Airflow pipelines

```
---
config:
  layout: fixed
---
flowchart TD
    A["Start Airflow Pipeline"] --> B["Snowflake Setup Pipeline"] & K["Course Registration Pipeline"] & O["Course Catalog Pipeline"] & T["Program Requirements Pipeline"]
    B --> C["Create Warehouse"] & D["Create Database and Schema"]
    D --> E["Create Classes Table"] & F["Create Course Catalog Table"] & G["Create Program Requirements Tables"] & H["Create User Profile Table"] & I["Create User Eligibility Table"]
    G --> G1["Program Requirements Table"] & G2["Core Requirements Table"] & G3["Elective Requirements Table"]
    I --> J["Snowflake Setup Complete"]
    K --> L["Scrape Data from Course Registration Platform"]
    L --> M["Transform Data: Clean and Validate"]
    M --> N["Load Data into Classes Table in Snowflake"]
    O --> P["Scrape Data from Course Catalog"]
    P --> Q["Transform Data: Chunk Descriptions for Pinecone"]
    Q --> R["Load Data into Course Catalog Table in Snowflake"] & S["Index Chunked Data in Pinecone"]
    T --> U["Scrape Data from Program Requirements"]
    U --> V["Transform Data: Parse and Organize"]
    V --> W["Load Data into Program Requirements Tables in Snowflake"]
    J --> End["Pipeline Complete"]
    N --> End
    S --> End
    W --> End
```

Mermaid code for Userflow:

```
---
config:
  layout: fixed
---
flowchart TD
    Start["Start: User Opens the Application"] --> A["Login Screen"]
    A --> B{"Has an Account?"}
    B -- Yes --> C["User Logs In"]
    B -- No --> D["User Registers"]
    D --> E["Create User Profile in Database"]
    E --> C
    C --> F["JWT Token Generated for User"]
    F --> G["User Dashboard"]
    G --> H{"Upload Transcript?"}
    H -- Yes --> I["User Uploads Transcript"]
    I --> J["Amazon Textract Processes Transcript"]
    J --> K["Eligibility Table Updated"]
    H -- No --> L["Proceed Without Transcript"]
    L --> M["Query Chatbot"]
    K --> M
    M --> N["Query Sent to Backend Agents"]
    N --> O["Response Constructed and Returned"]
    O --> P["Display Response to User"]
    P --> Q{"Another Query?"}
    Q -- Yes --> M
    Q -- No --> End["User Logs Out"]
```