Team member:

1-Abdalrhman khalid Akl ; ID:22100270
2-Mohamed Ahmed Ibrahim ; ID:22101115
3-Abdalrhman Elsawy ; ID:22100563
4-Abdalrhman Ragab Marey ; ID:22101325

Knowledge-Based System for AIU CSE

Course Registration Advising

Report

## Overview:

The Knowledge Base System (KBS) for Computer Science and Engineering (CSE) Admission Guidelines at Alexandria International University (AIU) is designed to help computer science and engineering (CSE) students choose the appropriate courses to enroll in. The system analyzes a student's semester, cumulative GPA, and course record (passed and failed courses) to recommend a personalized course load aligned with the Big Data Science track. The system ensures compliance with university policies, such as credit limits and prerequisites, and provides a clear and user-friendly explanation of its recommendations.

## System Component:

### 1. Knowledge Base:

### 1.1 Definition of the Knowledge Base:

The Knowledge Base is the main component that contains all organized information about the Computer Science Department courses at Alexandria International University (AIU). It represents the knowledge source upon which the other system components, such as the Inference Engine and Recommendation System, rely.

### 1.2 Data Collection Method:

Course data were collected through:

Approved curriculums of the Computer Science Department at Alexandria University.

Official course registration files in Excel format (Corrected_CSE_Courses3ver2.xls).

Information from students' academic advisors or course supervisors.

Data was converted to a programmable CSV format using the pandas library.

### 1.3 Data format:

Course Code: Unique course code (eg: CSE101)

Course Name: Course name

Description: Brief description of the course content

Prerequisites: Codes for previously required courses, separated by commas (if applicable)

Co-requisites: Codes for accompanying courses, separated by commas (if any)

Credit Hours: Number of credit hours for the course

Semester Offered: The semester in which the course is offered (Fall/Spring/Both)

Year: Academic year for which the course is recommended (1-4)

:

:

## 1.4 How to handle data (Data Handling):

The data was loaded into the program using the pandas library
courses_data = pd.read_csv("Corrected_CSE_Courses3ver2.csv")

## This is handled through the following processes:

Data Reading:
View all courses or by year and semester.

Use filtering to view specific groups of courses.

Data Editing:
Add a new course and verify that the code is not duplicated.

Edit details of an existing course.

Delete courses.

Data Validation:
Verify that the code is not duplicated.

Ensure that the prerequisites/co-requisites actually exist in the database.

Ensure that the number of hours and academic year are correct.

Prevent the user from entering non-existent requirements.

## 1.5 The importance of the Knowledge Base in the system:

It is the foundation upon which the Rule-based Inference Engine relies.

It is used to verify prerequisites and semester availability.

Without it, intelligent recommendations cannot be provided to students or university policies verified.

## 2. Knowledge Base Editor Overview :

The Knowledge Base Editor is an administrative tool used to manage the computer science course database (CSV), allowing you to view, add, edit, and delete courses, as well as verifying entered data such as pre-requisites/co-requisites and number of credit hours.

### 2.1 Basic functions in the code:

#### ✓ display_courses():

Displays all available courses in the database.
It displays basic information: course code, name, number of hours, semester, and academic year.

#### ✓ display_courses_by_semester_and_year():

Displays courses based on:

Semester (Fall/Spring/Both).

Academic year (optional).

Contains validation of user input.

#### ✓ add_course():

Adds a new course after entering all required fields.

Strong Validation:

Ensures that the course code is not duplicated.

That the prerequisites and co-requisites actually exist in the database.

That the number of hours and academic year are valid numbers.

#### ✓ edit_course():

Edit an existing course's data using its code.

The user can leave any field blank if they don't want to change it.

Supports validation of the number of hours entered.

#### ✓ delete_course():

Delete a course using its code.

The course's existence is checked before deletion.

### ✅ save_to_file():

Saves the modified data to a new CSV file.

### 2.2 Strengths :

Strong user input validation.

Prevents duplicate code.

Supports filtering by semester and academic year.

Easy to modify and extend.

Perfectly suited as a back-end admin tool.

### 2.3 Example 1: Enter a new course correctly

### Sampels input :

Course Code: AI403
Course Name: Natural Language Processing
Description: Introduction to NLP techniques and models
Prerequisites (comma-separated): AI401
Co-requisites (comma-separated):
Credit Hours ( Inteqer) 3
Semester Offered (Fall/Spring/Both): Fall
Year (1-4): 4

### Expected Output :

✅ The course has been added: AI403 - Natural Language Processing

### Example 2: Trying to enter a course with duplicate code:

Course Code: CSE111
...

Expected Output:

The course is already available.

### Example 3: Entering a course with a prerequisite that does not exist in the database:

Course Code: AI405
Course Name: Robotics
Description: Basics of robotics and control
Prerequisites (comma-separated): XYZ123
Co-requisites (comma-separated):
Credit Hours (integer): 3
Semester Offered (Fall/Spring/Both): Spring
Year (1-4): 4

Validation Feedback:

Prerequisites missing: XYZ123

# 3. INFERENCE ENGINE

The inference engine is the core of the recommendation system, implemented using the **Experta** library, a Python-based expert system framework. It processes student data and course information to generate tailored course recommendations based on predefined rules and constraints.

## 3.1 PURPOSE

The inference engine evaluates student attributes (e.g., CGPA, completed courses, failed courses, academic year, and semester) against course attributes (e.g., prerequisites, co-requisites, semester offered, and credit hours) to recommend suitable courses while respecting credit limits and academic requirements.

## 3.2 IMPLEMENTATION

The inference engine is encapsulated in the CourseRecommendationEngine class, which inherits from KnowledgeEngine in Experta. Below are the key components and their functionalities:

- **Facts**:
  - Student: Represents student information, including student_id, cgpa, completed_courses, failed_courses, year, and semester.
  - Course: Represents course details, such as course_id, prerequisites, corequisites, semester, credits, track, and year.
  - Recommendation: Stores recommended course IDs.
- **Rules**: The engine uses a rule-based approach with prioritized rules (via salience) to determine course recommendations:
  - **Credit Limit Rules** (salience=100):
    - If CGPA $\geq$ 3.5, the maximum credit limit is set to 22 (overload).
    - If 2.0 $\leq$ CGPA < 3.5, the limit is 20 (full load).
    - If CGPA < 2.0, the limit is 13 (half load).
    - These rules ensure that the total credits of recommended courses do not exceed the student's allowed limit.
  - **Failed Course Recommendation** (salience=20):
    - Recommends courses that the student has failed, provided prerequisites and co-requisites are met, the course is offered in the selected semester, and the student's academic year is sufficient.
  - **Core CS Course Recommendation** (salience=15):
    - Recommends core Computer Science (CS) courses (identified by the CSE prefix, excluding capstone courses like CSE493/CSE494) if prerequisites and co-requisites are satisfied and the course is available in the selected semester and year.
    - Special handling for capstone courses requires senior standing (90+ credits).
  - **Other Course Recommendation** (salience=0):

        - Recommends additional courses (e.g., AI, electives, or university courses) that meet prerequisites, co-requisites, semester, and year requirements, filling any remaining credit allowance.
- **Helper Methods**:
    - prerequisites_met: Checks if all prerequisites for a course are satisfied by the student's completed courses. Handles special cases like "SENIOR STANDING" (90+ credits).
    - corequisites_satisfied: Verifies that co-requisites are either completed or included in the current recommendations.
    - has_senior_standing: Determines if the student has accumulated 90 or more credits for senior-level courses.

### 3.3 Workflow

1. The engine is initialized with the course data (courses_df) and resets its state.
2. Student data is declared as a Student fact, and each course is declared as a Course fact with attributes like track (CS, AI, Elective, or University).
3. The engine runs the rules in order of salience, prioritizing credit limit setting, followed by failed courses, core CS courses, and other courses.
4. Recommendations are stored in self.recommendations, and the total credits are tracked to ensure compliance with the credit limit.

### 3.4 Significance

The inference engine ensures that recommendations are logical, adhere to academic policies (e.g., credit limits based on CGPA), and prioritize critical courses (failed or core CS courses). It handles complex dependencies like prerequisites and co-requisites, making it a robust decision-making tool.

## 4. EXPLANATION SYSTEM

The explanation system enhances the transparency of the recommendation process by providing clear reasons for why certain courses are recommended or excluded. This is critical for user trust and understanding.

### 4.1 Purpose

The explanation system logs the reasoning behind each recommendation or non-recommendation, addressing factors like credit limits, prerequisite/co-requisite satisfaction, semester availability, and senior standing requirements.

### 4.2 Implementation

The explanation system is integrated into the CourseRecommendationEngine class and operates as follows:

- **Storage**:
    - Explanations are stored in self.explanations (a list) and self.explanation_set (a set to avoid duplicates).
    - Each explanation is a string describing why a course was recommended or not.
- **Generation**:
    - Explanations are generated within the rule methods and helper functions:
        - **Credit Limit**: When setting the credit limit based on CGPA, an explanation is added (e.g., "Credit limit set to 22 (overload) because CGPA ≥ 3.5.").
        - **Prerequisites/Co-requisites**: If a course is not recommended due to unmet prerequisites or co-requisites, an explanation is logged (e.g., "Not recommended for CS301: Prerequisites not met (CS201, CS202).").
        - **Semester/Year Mismatch**: Courses not offered in the selected semester or year are flagged (e.g., "Not recommended for CS401: Not available in Fall Year 3, available in Spring Year 4.").
        - **Credit Limit Exceeded**: If a course would exceed the credit limit, an explanation is provided (e.g., "Not recommended for CS301: Exceeds credit limit of 20 credits.").
        - **Senior Standing**: For courses requiring senior standing, an explanation is added if the student lacks sufficient credits (e.g., "Not recommended for CSE493: Requires senior standing (90+ credits).").
        - **Failed Courses**: If a failed course is not recommended due to semester/year mismatch, a note is added (e.g., "Note: Failed course CS101 is not recommended in Fall Year 2. Retake it in Spring Year 1.").
- **Display**:
    - In the Streamlit GUI, explanations are displayed in a dedicated "Explanations" section.
    - Recommended course explanations are styled with a green background (explanation-success), while warnings (e.g., why a course was not recommended) use a yellow background (explanation-warning).

### 4.3 Significance

The explanation system makes the recommendation process transparent, helping students understand the rationale behind their course suggestions. It also aids administrators in debugging or verifying the system's logic, ensuring compliance with academic rules.

## 5. STREAMLIT GUI

The Streamlit GUI provides an interactive and user-friendly interface for both administrators and students to interact with the course management and recommendation system.

### 5.1 Purpose

The GUI enables:

- **Administrators** to manage courses (view, filter, add, edit, delete, and save changes).
- **Students** to input their details and receive personalized course recommendations with explanations.

## 5.2 Implementation

The Streamlit GUI is structured with a modular design, using custom styling and interactive components. Key features include:

- **Custom Styling**:
  - A CSS block defines styles for the sidebar, cards, buttons, and explanation sections to enhance visual appeal and usability.
  - Examples include rounded buttons, shadowed cards, and color-coded explanation boxes (green for success, yellow for warnings).
- **Admin Mode**:
  - **Password Authentication**: A simple SHA-256 hashed password check (admin123) restricts access to admin functions.
  - **Functionalities**:
    - **Display All Courses**: Shows a table of all courses with columns for code, name, credits, semester, and year.
    - **Filter by Semester/Year**: Allows filtering courses by semester (Fall, Spring, or Both) and academic year (1-4 or all).
    - **Add Course**: A form to input course details (code, name, description, prerequisites, co-requisites, credits, semester, year) with validation for duplicate codes and invalid prerequisites/co-requisites.
    - **Edit Course**: A form to update an existing course's details, pre-filled with current values.
    - **Delete Course**: A dropdown and button to remove a course from the dataset.
    - **Save Changes**: Saves the updated course data to a new CSV file (Corrected_CSE_Courses3ver2_UPDATED.csv).
- **Student Mode**:
  - **Input Form**: Collects student details (ID, CGPA, year, semester, completed courses, failed courses) using text inputs, number inputs, and multiselect dropdowns.
  - **Validation**: Checks for valid inputs (e.g., CGPA between 0.0 and 4.0, no overlapping completed and failed courses).
  - **Recommendations Display**: Shows recommended courses with their credit hours, semester, and year in green-styled boxes.
  - **Explanations Display**: Lists explanations for recommendations and non-recommendations, styled according to their type (success or warning).
- **Sidebar**:
  - Contains a mode selector (Admin or Student) and an "About the System" section describing the system's purpose and features.
  - Styled with a light background for better readability.
- **Data Handling**:

- Uses @st.cache_data to load the course CSV file (Corrected_CSE_Courses3ver2.csv) efficiently.
- Stores course data in st.session_state to persist changes during the session.
- Handles errors gracefully (e.g., missing CSV file, invalid columns).

### 5.3 SIGNIFICANCE

The Streamlit GUI provides an intuitive interface that simplifies course management for administrators and course selection for students. Its responsive design, clear feedback (success/error messages), and visual hierarchy (via cards and styling) enhance user experience. The integration with the inference engine and explanation system ensures that recommendations are both accessible and understandable.

## 6. CONCLUSION

The **Inference Engine**, **Explanation System**, and **Streamlit GUI** form the backbone of the AIU Course Management & Recommendation System. The inference engine leverages rule-based logic to deliver accurate and policy-compliant course recommendations. The explanation system enhances transparency by providing clear reasons for each decision. The Streamlit GUI offers a user-friendly interface for both administrators and students, making the system accessible and efficient. Together, these components create a comprehensive tool that supports academic planning and decision-making at AIU.