Department of Computer Science & Engineering

Course Title: Artificial Intelligence and Expert Systems Lab

Course Code: CSE 404

Date of Submission: 22/02/2025

Submitted To:	**Submitted By:**
Noor Mairukh Khan Arnob	Name: Swapna Khanam
Lecturer,	Reg no: 21201082
Department of CSE, UAP	Sec: B2

i) Problem Title:

Development of a Knowledge-Based “University Student Records System” Using Prolog.

ii) Problem Description:

Modern universities require efficient management of student and faculty records, including enrollments, courses, professor assignments, and grading. Traditional methods are error-prone, so we propose a knowledge-based system using Prolog for efficient data storage, retrieval, and manipulation through logical queries.

Key Features:

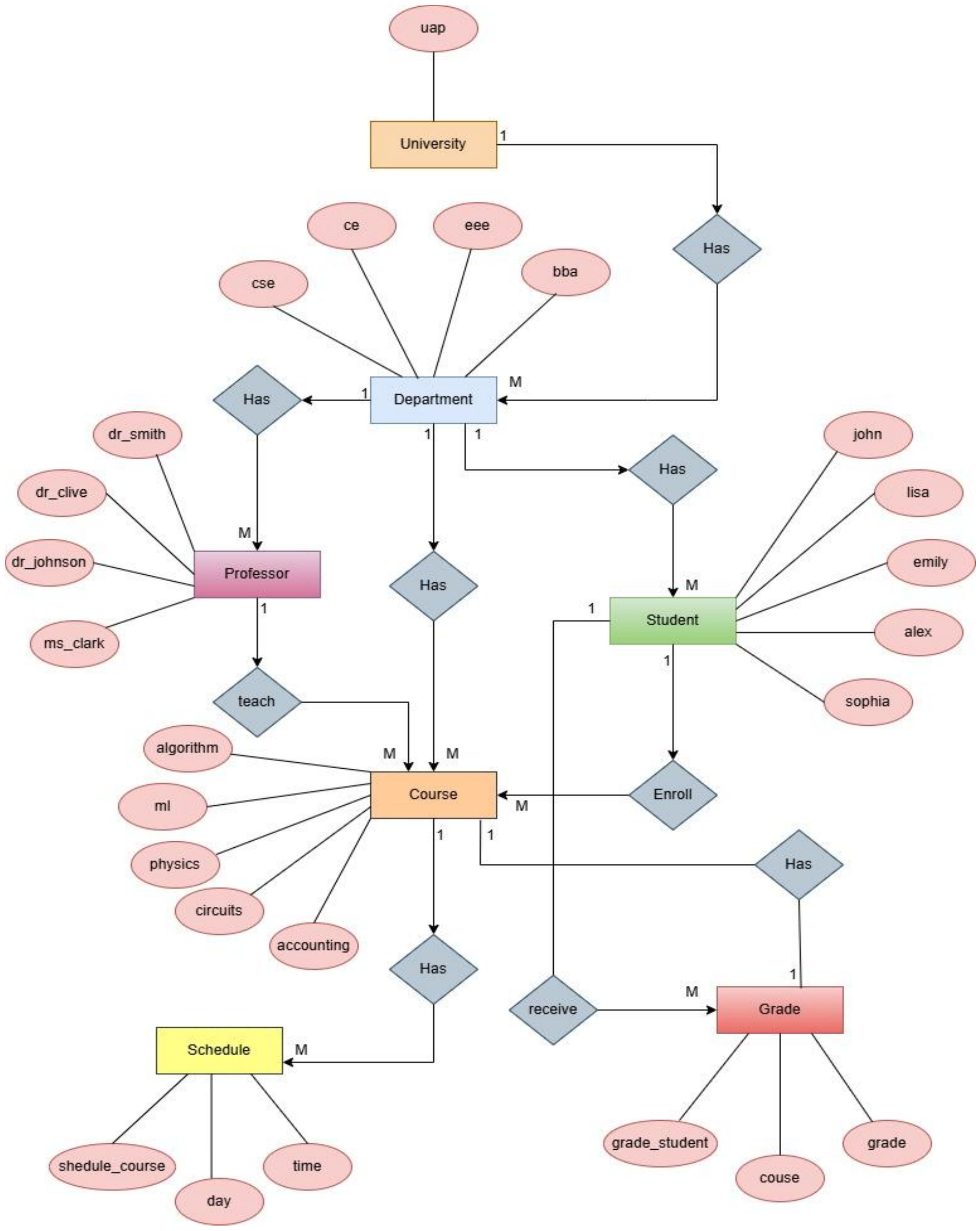
- Department-wise faculty and student categorization
- Course enrollment and professor assignments
- Student performance tracking (grades & schedules**)**
- Efficient information retrieval using logical queries

iii) Tools and Languages Used:

- **Language (Prolog):** Logic-based programming language used to define facts and rules.
- **SWI-Prolog:** SWI-Prolog is an open-source Prolog interpreter that allows users to execute Prolog code, making it easier to develop, test, and debug logic-based programs.
- **Draw.io:** Used to design the Tree Figure and ER diagram.
- **Notepad++ / VS Code:** Code editor for Prolog development

iv) <u>Diagram/Figure:</u>

Entity-Relationship (ER) Diagram:



Explanation:

1. University has multiple Departments.
2. Each Department has multiple Professors, Students, and Courses.
3. Professors teach Courses.
4. Students enroll in Courses.
5. Each Student receives Grades for enrolled Courses.
6. Each Course has a Schedule.

Entities and Relationships:

1. University ↔ (1:M) ↔ Department
2. Department ↔ (1:M) ↔ Professor
3. Department ↔ (1:M) ↔ Student
4. Department ↔ (1:M) ↔ Course
5. Professor ↔ (1:M) ↔ Course
6. Student ↔ (1:M) ↔ Course
7. Student ↔ (1:M) ↔ Grade
8. Course ↔ (1:M) ↔ Schedule
9. Course ↔ (1:1) ↔ Grade

v) <u>Sample Input/Output:</u>

1. Find students in a university:

<u>Input:</u> ?- student_in_university(Student, University).

<u>Output:</u>

```
Student = john,

University = uap ;

Student = lisa,

University = uap ;

Student = emily,

University = uap ;

Student = alex,

University = uap ;

Student = sophia,

University = uap.
```

2. Find students in a department:

<u>Input:</u> ?- students_in_department(Dept, Student).

<u>Output:</u>

```
Dept = cse,

Student = john ;

Dept = ce,

Student = lisa ;

Dept = eee,

Student = emily ;
```

```
Dept = bba,
Student = alex ;
Dept = cse,
Student = sophia.
```

3. Find courses in a department:

Input: ?- course_in_department(Course, Department).

Output:

```
Course = algorithm,
Department = cse ;
Course = ml,
Department = cse ;
Course = physics,
Department = ce ;
Course = circuits,
Department = eee ;
Course = accounting,
Department = bba.
```

4. Check if a student is enrolled in a course:

Input: ?- student_enrolled_in_course(Student, Course).

Output:

```
Student = john,
Course = algorithm ;
Student = john,
Course = ml ;
```

```
Student = lisa,
Course = physics ;
Student = emily,
Course = circuits ;
Student = alex,
Course = accounting ;
Student = sophia,
Course = ml.
```

5. Get professor teaching a course:

Input: ?- professor_of_course(Professor, Course).

Output:

```
Professor = dr_smith,
Course = algorithm ;
Professor = dr_smith,
Course = ml ;
Professor = dr_clive,
Course = physics ;
Professor = dr_johnson,
Course = circuits ;
Professor = ms_clark,
Course = accounting.
```

6. Find courses taught by a professor:

Input: ?- courses_by_professor(Professor, Course).

Output:

```
Professor = dr_smith,
Course = algorithm ;
Professor = dr_smith,
Course = ml ;
Professor = dr_clive,
Course = physics ;
Professor = dr_johnson,
Course = circuits ;
Professor = ms_clark,
Course = accounting.
```

7. Find students taught by a professor:

Input: ?- students_of_professor(Professor, Student).

Output:

```
Professor = dr_smith,
Student = john ;
Professor = dr_smith,
Student = john ;
Professor = dr_smith,
Student = sophia ;
Professor = dr_clive,
Student = lisa ;

Professor = dr_johnson,

Student = emily ;

Professor = ms_clark,

Student = alex.
```

8. Get a student's grade in a course:

Input: ?- student_grade(Student, Course, Grade).

Output:

```
Student = john,

Course = algorithm,

Grade = a ;

Student = john,

Course = ml,

Grade = b ;

Student = lisa,

Course = physics,

Grade = a ;

Student = emily,

Course = circuits,

Grade = a ;

Student = alex,

Course = accounting,

Grade = c ;
```

```
Student = sophia,

Course = ml,

Grade = a.
```

9. Get the schedule of a course:

Input: ?- course_schedule(Course, Day, Time).

Output:

```
Course = algorithm,

Day = monday,

Time = '10:00 AM' ;

Course = ml,

Day = wednesday,

Time = '2:00 PM' ;

Course = physics,

Day = wednesday,

Time = '2:00 PM' ;

Course = circuits,

Day = tuesday,

Time = '11:00 AM' ;

Course = accounting,

Day = thursday,

Time = '1:00 PM'.
```

10. Detect scheduling conflicts (same time, same day, different courses:

Input: ?- schedule_conflict(Course1, Course2, Day, Time).

Output:

```
Course1 = ml,

Course2 = physics,

Day = wednesday,

Time = '2:00 PM' ;

Course1 = physics,

Course2 = ml,

Day = wednesday,

Time = '2:00 PM' ;

false.
```

vi) Conclusion and Challenges:

Conclusion:

The “University Student Records System” developed using Prolog provides an efficient knowledge-based approach to managing student enrollments, faculty assignments, and grading records. By defining relationships and facts, users can quickly retrieve and manipulate university data. The logical inference capabilities of Prolog allow advanced queries, making the system scalable and adaptable.

Challenges Faced:

1. **Complex Query Handling** - Prolog queries require precise syntax and can be challenging for users unfamiliar with logic programming.
2. **Scalability Issues** - As the number of students and courses grows, managing large datasets in Prolog becomes challenging compared to SQL-based databases.
3. **Visualization Limitations** - Unlike relational databases with UI-based tools, Prolog lacks built-in visualization for records, requiring external tools for representation.