ATTENDANCE TRACKING AND MANAGEMENT SYSTEM

DBMS PROJECT

Team 23

Sai Sumakar - CS22B1077

M Sandesh - CS22B1076

M Bhuvan - CS22B1075

A Sai Dinesh - CS22B1074

S Aditya - CS22B1101

Software Requirements System(SRS)

Purpose

The purpose of the College Attendance Tracking and Management System is to provide a comprehensive solution for efficient and accurate tracking of attendance in a college environment, enhancing administrative efficiency and student engagement.

Scope

The system will include features for user authentication, user roles, class management, student and teacher databases, attendance recording, reports and alerts, leave management, integration with timetables, and interactive UI and dashboards.

Abbreviations

UI: User interface

Overall Description

Product Perspective

The College Attendance Tracking and Management System will be a standalone web-based application.

Product Features

- User authentication and role management
- Class creation, scheduling, and management
- Student and teacher database management
- Attendance recording
- Generation of reports and alerts
- Leave management for students and faculty
- Integration with college timetables
- Interactive UI and dashboards for users

User Characteristics

Define different roles (administrator, faculty, student) with specific permissions and access levels.

Product Operation

The system will be web-based, accessible through modern web browsers such as Chrome, Firefox, and Safari. It should also support mobile devices for on-the-go access for faculty.

System Features

User Authentication

Secure login for administrators, faculty, and students with username and password authentication.

User Roles

- Administrators: change students courses, timetable, leave approving, user accounts, and overall system functionality.
- Faculty: Record attendance, manage classes, access student information.
- Students: View attendance records, submit leave requests, access personal information.

Class Management

Create, schedule, and manage classes including subjects, instructors, rooms, and timings. Courses are taught by multiple faculty in multiple rooms.

Student Database

Maintain a database of student information including profiles, enrollment status, and academic details.

Teacher Database

Manage faculty profiles including qualifications, courses taught, and schedules.

Attendance Recording

- Allow faculty to mark attendance for each class session.
- Access is given to only the faculty who is teaching that specific class session.
- Attendance is taken in multiple ways (options):
  1. By entering all the rolls of absent students
  2. By entering all the rolls of students who are present
  3. By manually selecting present or absent for each student in that class.
- Faculty can remove and add the attendance of students.

Reports and Alerts

- Generate attendance reports for individual students, classes, and faculty.
- Automated alerts for absent students, low attendance rates ($<85\%$), or important events (non-instructional days, fests, etc.).

Leave Requests

- Allow students to submit leave requests with reasons, and faculty to approve or deny leave requests.
- Leave request once approved affect the attendance of the student by marking no class for respective leave days.

Integration with Timetables

Integrate with college timetables to align attendance tracking with class schedules.

Interactive UI

Develop intuitive and user-friendly interfaces with interactive dashboards for easy access to attendance data, reports, and alerts. While taking attendance, faculty is able to select the class they are taking attendance for and the way of taking attendance. If chosen option 3, faculty should be displayed with each student's roll and options to mark present or absent one by one.

External Interface Requirements

User Interfaces

Intuitive web-based interfaces for administrators, faculty, and students with responsive design for mobile devices.

Hardware Interfaces

Compatibility with standard computer systems and mobile devices.

Non-Functional Requirements

Performance

Ensure system responsiveness and scalability to handle concurrent users and large datasets.

Security

Implement robust security measures to protect sensitive attendance data, including encryption, access controls, and secure authentication methods.

Reliability

Minimize system downtime and ensure data integrity through regular backups and disaster recovery plans.

Usability

Design interfaces that are intuitive, easy to navigate, and accessible to users with varying levels of technical expertise.

Other Requirements

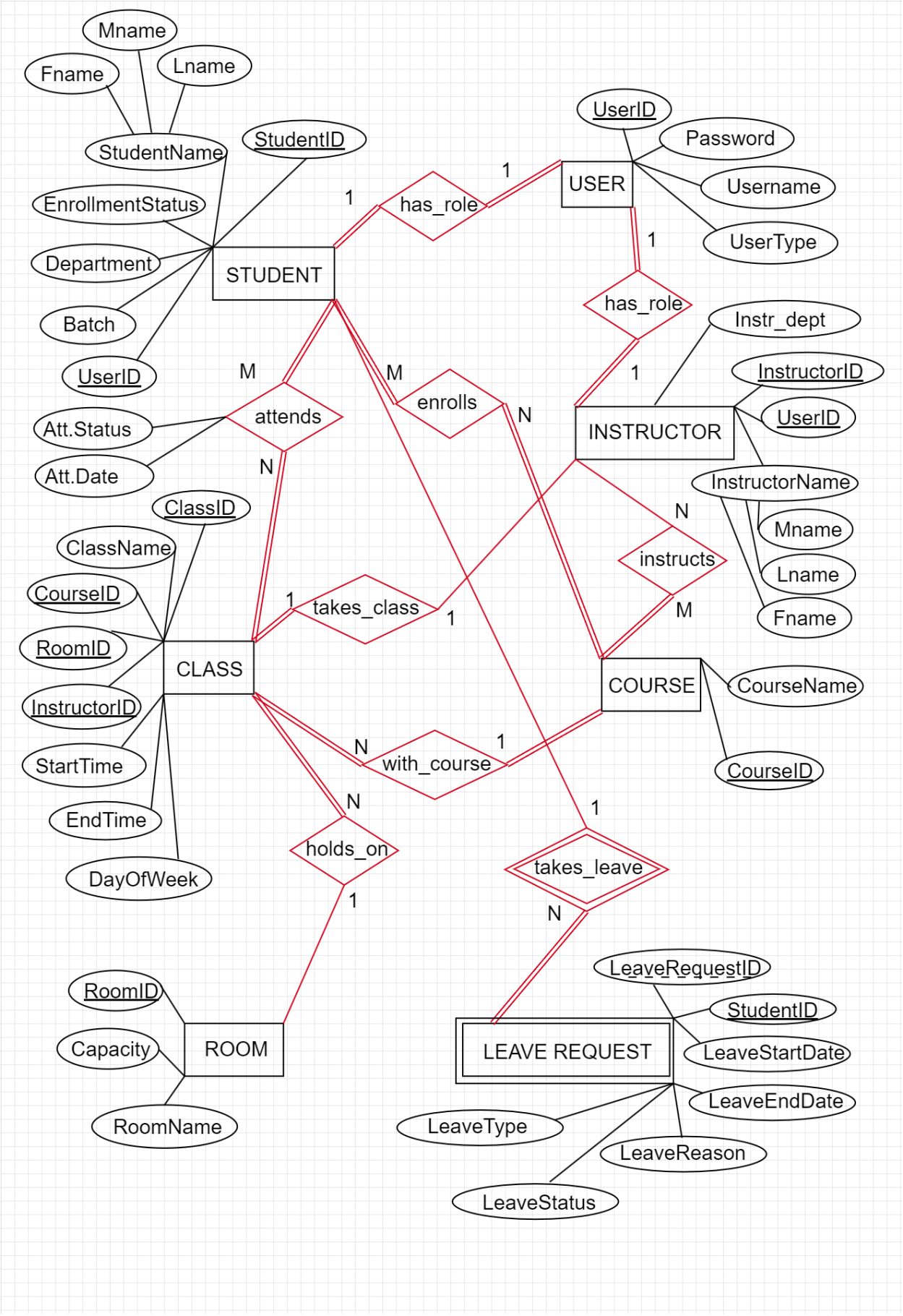
- Obtain necessary approvals and permissions for collecting, storing, and processing personal data of students and faculty members.
- Provide comprehensive user manuals, system documentation, and training materials for administrators, faculty, and students.
- Maintain up-to-date documentation for system architecture and data schemas.

Conclusion

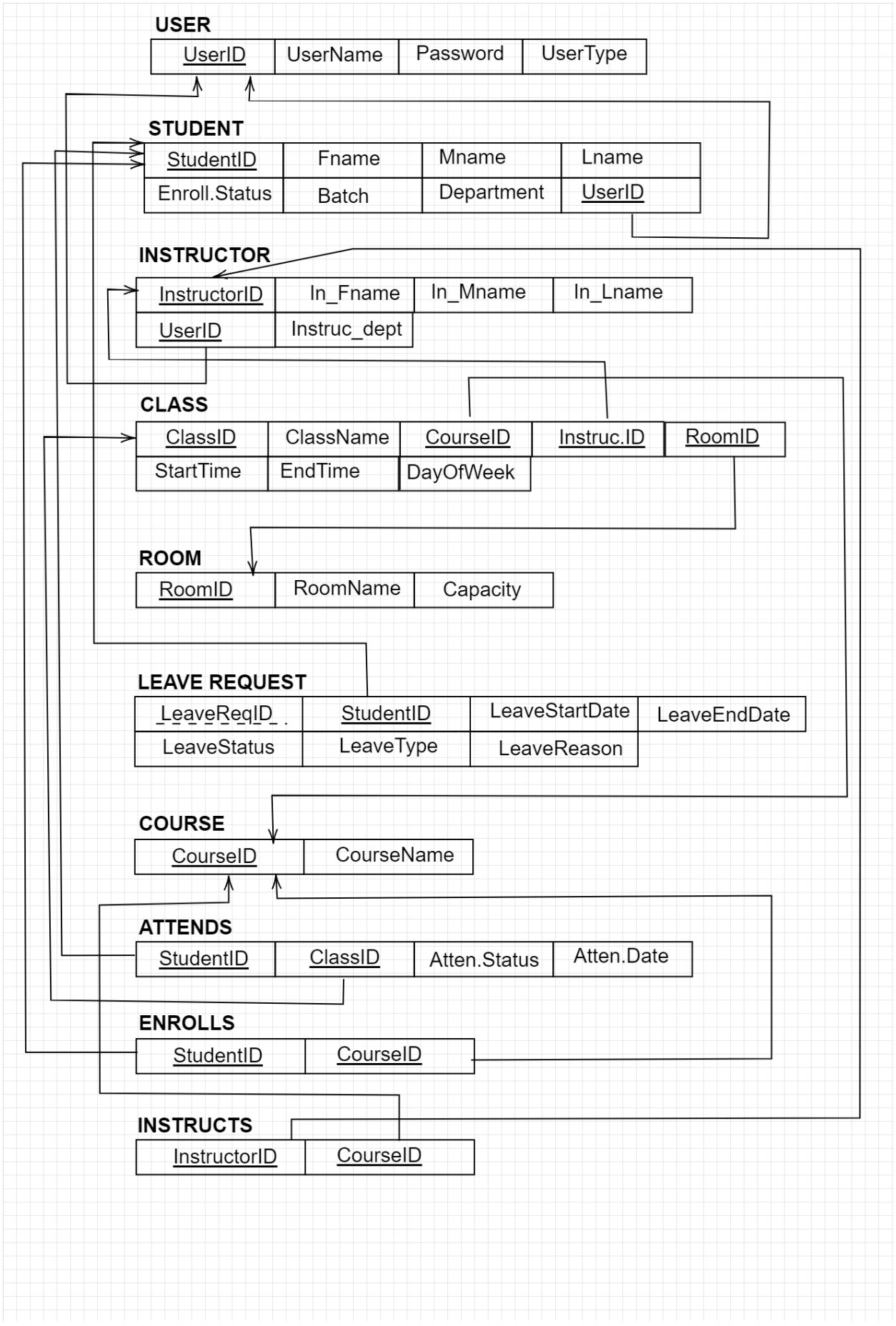
This Software Requirements Specification outlines the key features, functionalities, and requirements for the Attendance Tracking and Management System, providing a comprehensive framework for development and implementation.

By addressing the needs of administrators, faculty, and students, the system aims to streamline attendance tracking processes, enhance user experience, and improve overall administrative efficiency in a college environment.

ER Model


Mname
Fname
Lname
StudentName
StudentID
EnrollmentStatus
Department
Batch
UserID
STUDENT
has_role
1
1
USER
UserID
Password
Username
UserType
1
has_role
1
Instr_dept
InstructorID
UserID
INSTRUCTOR
InstructorName
Mname
Lname
Fname
M
attends
M
enrolls
N
N
Att.Status
Att.Date
N
instructs
M
ClassID
ClassName
CourseID
RoomID
InstructorID
StartTime
EndTime
DayOfWeek
1
takes_class
1
CLASS
COURSE
CourseName
CourseID
N
with_course
1
N
holds_on
1
1
takes_leave
N
RoomID
Capacity
RoomName
ROOM
LeaveRequestID
StudentID
LeaveStartDate
LEAVE REQUEST
LeaveEndDate
LeaveType
LeaveReason
LeaveStatus

Relational Schema



Physical Data Model

Entities and Attributes:

User:

Attribute Name	Data Type	Key Type
UserID	varchar	Primary Key
Username	varchar	Non-key
Password	varchar	Non-key
UserType	varchar	Non-key

Student:

Attribute Name	Data Type	Key Type
StudentID	varchar	Primary Key
Fname	varchar	Non-key
Mname	varchar	Non-key
Lname	varchar	Non-Key
Enroll.status	varchar	Non-key
Batch	varchar	Non-key
Department	varchar	Non-key
UserID	varchar	Foreign Key

Instructor:

Attribute Name	Data Type	Key Type
InstructorID	varchar	Primary Key
Instruc.Fname	varchar	Non-key
Instruc.Mname	varchar	Non-key
Instruc.Lname	varchar	Non-Key
Instruc.Dept	varchar	Non-key
UserID	varchar	Foreign key

Class:

Attribute Name	Data Type	Key Type
ClassID	varchar	Primary Key
ClassName	varchar	Non-key
CourseID	varchar	Foreign key
InstrucID	varchar	Foreign key
RoomID	varchar	Foreign key
StartTime	time	Non-key
EndTime	time	Non-key
DayOfWeek	varchar	Foreign Key

Room:

Attribute Name	Data Type	Key Type
RoomID	varchar	Primary Key
RoomName	varchar	Non-key
Capacity	int	Non-key

Leave Request:

Attribute Name	Data Type	Key Type
LeaveReqID	varchar	Partial Key
StudentID	varchar	Foreign key
LeaveStartDate	date	Non-key
LeaveEndDate	date	Non-key
LeaveType	varchar	Non-key
LeaveReason	varchar	Non-key
LeaveStatus	varchar	Non-Key

Course:

Attribute Name	Data Type	Key Type
CourseID	varchar	Primary Key
CourseName	varchar	Non-Key

Attends:

Attribute Name	Data Type	Key Type
StudentID	varchar	Foreign Key
ClassID	varchar	Foreign key
Atten.Status	varchar	Non-key
Atten.Date	date	Non-Key

Enrolls:

Attribute Name	Data Type	Key Type
StudentID	varchar	Foreign Key
CourseID	varchar	Foreign Key

Instructs:

Attribute Name	Data Type	Key Type
InstructorID	varchar	Foreign Key
CourseID	varchar	Foreign Key

Relationships:

User-Student (1:1):

Relation specifying UserType of a user(here Student). Each student in the system corresponds to exactly one user for authentication and access purpose. Similarly each user account is associated with exactly one student.

User-Instructor (1:1):

Relation specifying UserType of a user(here Instructor). Each instructor in the system corresponds to exactly one user similarly to that of student-user relationship. Similarly each user account is associated with exactly one instructor.

Student-Course (M:N):

Describes the registered courses of a student. A student can be enrolled in multiple courses and a course can have multiple students enrolled in it.

Course-Instructor (M:N):

Describes the courses which are taught by an Instructor. An instructor can teach multiple courses and a course can be taught by multiple instructors.

Student-Class (M:N):

Relation describring the classes attended by a student. A student can attend multiple classes and a class can be attended by multiple students.

Instructor-Class (1:N):

Relation describring the classes handled by an Instructor. Each class is typically taught by one instructor and each instructor teaches multiple classes.

Class-Course (N:1):

Describes the courses taught in a class. Each class is exactly associated with one course, but a course may have multiple classes associated with it.

Class-Room (N:1):

Describes the rooms in which classes are taught. Similarly, each class is typically conducted in one room, but a room may host multiple classes.

Student-Leave Request (1:N):

Relation providing all the leave requests of a student. A student can submit multiple leave requests, but each leave request is associated with exactly one student.

Population Queries(MySQL):

- Database and tables

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| attendance         |
| govtoffice         |
| information_schema |
| mysql              |
| performance_schema |
| sailors            |
| sys                |
| workson            |
+--------------------+
8 rows in set (0.01 sec)

mysql> use attendance;
Database changed
mysql> show tables;
+----------------------+
| Tables_in_attendance |
+----------------------+
| attends              |
| class                |
| course               |
| enrolls              |
| instructor           |
| instructs            |
| leaverequest         |
| room                 |
| student              |
| user                 |
+----------------------+
10 rows in set (0.00 sec)
```

- Example entries in course table

```
mysql> select * from course;
+----------+-----------------------+
| courseid | coursename            |
+----------+-----------------------+
| cs001    | databases             |
| cs002    | theory of computation |
| cs003    | computer organisation |
| cs004    | data structures       |
| ec001    | signal processing     |
| ec002    | embedded systems      |
| ec003    | machine language      |
| ec004    | machine learning      |
| me001    | nano tech             |
| me002    | materials             |
| me003    | smart manfacturing    |
| me004    | 3d printing           |
+----------+-----------------------+
12 rows in set (0.00 sec)
```

- Example entries in student table

```
mysql> select * from student;
+-----------+----------+-------+------------+--------------+---------+------------+--------+
| studentid | fname    | mname | lname      | enrollstatus | batch   | department | userid |
+-----------+----------+-------+------------+--------------+---------+------------+--------+
| s001      | Maria    |       | Montessori | enrolled     | batch-a | cse        | u006   |
| s002      | John     |       | Dewey      | enrolled     | batch-a | cse        | u007   |
| s003      | Aarav    |       | Patel      | enrolled     | batch-b | ece        | u008   |
| s004      | Aanya    |       | Sharma     | enrolled     | batch-b | ece        | u009   |
| s005      | Advik    |       | Singh      | enrolled     | batch-b | ece        | u010   |
| s006      | Ananya   |       | Gupta      | enrolled     | batch-b | ece        | u011   |
| s007      | Aaradhya |       | Reddy      | enrolled     | batch-b | ece        | u012   |
| s008      | Aadi     |       | Joshi      | enrolled     | batch-b | ece        | u013   |
| s009      | Aanya    |       | Mehra      | enrolled     | batch-c | mech       | u014   |
| s010      | Aarav    |       | Kapoor     | enrolled     | batch-c | mech       | u015   |
| s011      | Aarush   |       | Khanna     | enrolled     | batch-c | mech       | u016   |
| s012      | Aahana   |       | Patel      | not enrolled | batch-c | mech       | u017   |
| s013      | Aanya    |       | Tiwari     | enrolled     | batch-c | mech       | u018   |
| s014      | Advait   |       | Sharma     | enrolled     | batch-c | mech       | u019   |
| s015      | Aarav    |       | Singhania  | enrolled     | batch-c | mech       | u020   |
| s016      | Aaradhya |       | Bhatia     | enrolled     | batch-c | mech       | u021   |
| s017      | Aarya    |       | Sharma     | enrolled     | batch-c | mech       | u022   |
| s018      | Aadit    |       | Singh      | not enrolled | batch-c | mech       | u023   |
| s019      | Aadi     |       | Verma      | enrolled     | batch-c | mech       | u024   |
| s020      | Aarav    |       | Gupta      | enrolled     | batch-a | cse        | u025   |
+-----------+----------+-------+------------+--------------+---------+------------+--------+
20 rows in set (0.00 sec)
```

- Example entries in user table

```
mysql> select * from user;
+--------+---------------------+-------------------------+----------+
| userid | username            | passkey                 | usertype |
+--------+---------------------+-------------------------+----------+
| u001   | Socrates            | Password123!            | teach    |
| u002   | Confucius           | Password$ecur1ty        | teach    |
| u003   | Plato               | 3xamplePassword@        | teach    |
| u004   | Aristotle           | Password456Sunshine     | teach    |
| u005   | Anne Sullivan       | P@ssword789!            | teach    |
| u006   | Maria Montessori    | 9PurplePassword!        | student  |
| u007   | John Dewey          | OceanWavePassword123    | student  |
| u008   | Aarav Patel         | Ch0c0l@tePassword&123   | student  |
| u009   | Aanya Sharma        | M0onLightPassword!      | student  |
| u010   | Advik Singh         | W@terf@llPassword789    | student  |
| u011   | Ananya Gupta        | 4F0r3stPassword&Fun     | student  |
| u012   | Aaradhya Reddy      | S@ndCastl3Password$1    | student  |
| u013   | Aadi Joshi          | 2StarL!ghtPassword      | student  |
| u014   | Aanya Mehra         | W1ldFlow3rPassword@     | student  |
| u015   | Aarav Kapoor        | P@lmTr33$Password123    | student  |
| u016   | Aarush Khanna       | B3@chBum!Password456    | student  |
| u017   | Aahana Patel        | 8M@gicPasswordM00n      | student  |
| u018   | Aanya Tiwari        | 5R@inbow$!Password      | student  |
| u019   | Advait Sharma       | L@k3Sid3#Password123    | student  |
| u020   | Aarav Singhania     | F1r3Fly!ght@Password    | student  |
| u021   | Aaradhya Bhatia     | 6SunRis3$Password       | student  |
| u022   | Aarya Sharma        | B3ll@Fl0w3rPassword789  | student  |
| u023   | Aadit Singh         | @utumNLe@f1Password     | student  |
| u024   | Aadi Verma          | 1Sn0wF1@k3sPassword     | student  |
| u025   | Aarav Gupta         | P@ssword123!            | student  |
| u026   | Surya Raj Prakash   | 23P@ssw234ord123!       | teach    |
| u027   | Arya Kumar Singh    | dfgP@sswor2323d123!     | teach    |
| u028   | Raj Kumar Verma     | 22P@sswefword123!       | teach    |
| u029   | Varun Sinha Sharma  | wef2P@ssword123!        | teach    |
| u030   | Kavya Malhotra Gupta | 823fM@gicPasswordM00n  | teach    |
+--------+---------------------+-------------------------+----------+
30 rows in set (0.00 sec)
```

- Example entries in instructor table

```
mysql> select * from instructor;
+--------------+-----------------+-----------------+-----------------+----------------+--------+
| instructorid | instructorfname | instructormname | instructorlname | instructordept | userid |
+--------------+-----------------+-----------------+-----------------+----------------+--------+
| i001         | Socrates        |                 |                 | cse            | u001   |
| i002         | Confucius       |                 |                 | ece            | u002   |
| i003         | Plato           |                 |                 | ece            | u003   |
| i004         | Aristotle       |                 |                 | mech           | u004   |
| i005         | Anne            | Helen           | Sullivan        | mech           | u005   |
| i006         | Surya           | Raj             | Prakash         | cse            | u026   |
| i007         | Arya            | Kumar           | Singh           | ece            | u027   |
| i008         | Raj             | Kumar           | Verma           | ece            | u028   |
| i009         | Varun           | Sinha           | Sharma          | mech           | u029   |
| i010         | Kavya           | Malhotra        | Gupta           | mech           | u030   |
+--------------+-----------------+-----------------+-----------------+----------------+--------+
10 rows in set (0.00 sec)
```

- Example entries in attends table

```
mysql> select * from attends;
+-----------+---------+------------------+----------------+
| studentid | classid | attendancestatus | attendancedate |
+-----------+---------+------------------+----------------+
| s001      | c001    | present          | 2025-08-01     |
| s001      | c001    | absent           | 2024-05-13     |
| s002      | c001    | absent           | 2024-05-13     |
| s003      | c001    | absent           | 2024-05-13     |
| s004      | c001    | present          | 2024-05-13     |
| s005      | c001    | present          | 2024-05-13     |
| s001      | c004    | absent           | 2024-05-14     |
| s002      | c004    | present          | 2024-05-14     |
| s003      | c004    | absent           | 2024-05-14     |
| s004      | c004    | present          | 2024-05-14     |
| s005      | c004    | present          | 2024-05-14     |
| s001      | c004    | absent           | 2024-05-13     |
| s002      | c004    | present          | 2024-05-13     |
| s003      | c004    | absent           | 2024-05-13     |
| s004      | c004    | present          | 2024-05-13     |
| s005      | c004    | present          | 2024-05-13     |
| s001      | c004    | absent           | 2024-05-08     |
| s002      | c004    | present          | 2024-05-08     |
| s003      | c004    | absent           | 2024-05-08     |
| s004      | c004    | present          | 2024-05-08     |
| s005      | c004    | present          | 2024-05-08     |
| s006      | c002    | present          | 2024-05-13     |
| s007      | c002    | present          | 2024-05-13     |
| s008      | c002    | present          | 2024-05-13     |
| s009      | c002    | present          | 2024-05-13     |
| s010      | c002    | present          | 2024-05-13     |
| s011      | c002    | present          | 2024-05-13     |
| s012      | c002    | present          | 2024-05-13     |
| s013      | c002    | present          | 2024-05-13     |
| s014      | c002    | present          | 2024-05-13     |
| s015      | c002    | present          | 2024-05-13     |
| s001      | c001    | absent           | 2024-05-06     |
| s002      | c001    | absent           | 2024-05-06     |
| s003      | c001    | present          | 2024-05-06     |
| s004      | c001    | present          | 2024-05-06     |
| s005      | c001    | present          | 2024-05-06     |
| s001      | c001    | present          | 2024-04-29     |
| s002      | c001    | absent           | 2024-04-29     |
| s003      | c001    | absent           | 2024-04-29     |
| s004      | c001    | present          | 2024-04-29     |
| s005      | c001    | present          | 2024-04-29     |
+-----------+---------+------------------+----------------+
```

- Example entries in leaverequest table

```
mysql> select * from leaverequest;
+------------+-----------+---------------+--------------+-----------+--------------------------------+-------------+
| leavereqid | studentid | leavestartdate | leaveenddate | leavetype | leavereason                    | leavestatus |
+------------+-----------+---------------+--------------+-----------+--------------------------------+-------------+
| l001       | s001      | 2024-01-01     | 2024-02-01   | vacation  | family trip                    | pending     |
| l002       | s007      | 2024-01-01     | 2024-01-10   | medical   | fever                          | pending     |
| l003       | s011      | 2024-01-01     | 2024-02-01   | vacation  | family trip                    | pending     |
| l004       | s018      | 2024-01-01     | 2024-01-10   | medical   | fever                          | pending     |
| L006       | NULL      | 2024-05-01     | 2024-05-05   | personal  | acd                            | pending     |
| l007       | s001      | 2024-05-03     | 2024-05-05   | sick      | not weelll                     | pending     |
| l008       | s001      | 2024-04-30     | 2024-05-03   | sick      | sdwq                           | pending     |
| l009       | s001      | 2024-04-29     | 2024-04-30   | sick      | qqewfegwr                      | pending     |
| l010       | s001      | 2024-05-08     | 2024-05-11   | sick      | not well due to fever and cold | pending     |
| l011       | s001      | 2024-05-13     | 2024-05-16   | sick      | high fever, headache           | pending     |
| l012       | s001      | 2024-05-10     | 2024-05-12   | sick      | vacation                       | pending     |
+------------+-----------+---------------+--------------+-----------+--------------------------------+-------------+
11 rows in set (0.00 sec)
```

- Example entries in instructs table

```
mysql> select * from instructs;
+--------------+----------+
| instructorid | courseid |
+--------------+----------+
| i001         | cs001    |
| i001         | cs002    |
| i002         | ec001    |
| i003         | ec002    |
| i004         | me001    |
| i005         | me002    |
| i002         | ec002    |
| i003         | ec001    |
| i004         | me002    |
| i005         | me001    |
+--------------+----------+
10 rows in set (0.00 sec)
```

- Example entries in room table

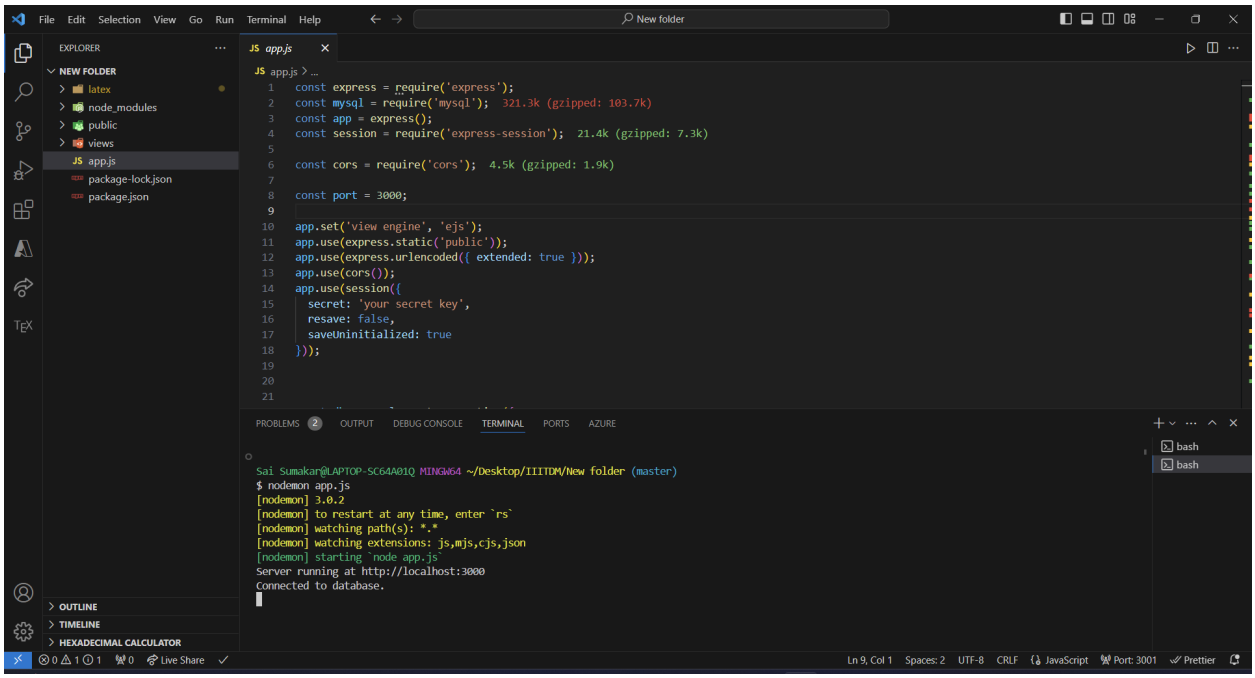
```
mysql> select * from room;
+--------+----------+----------+
| roomid | roomname | capacity |
+--------+----------+----------+
| r001   | h05      |       10 |
| r002   | h06      |       10 |
| r003   | h15      |        8 |
| r004   | h03      |       10 |
| r005   | h26      |       11 |
| r006   | h25      |        5 |
+--------+----------+----------+
6 rows in set (0.00 sec)
```

- Example entries in enrolls table

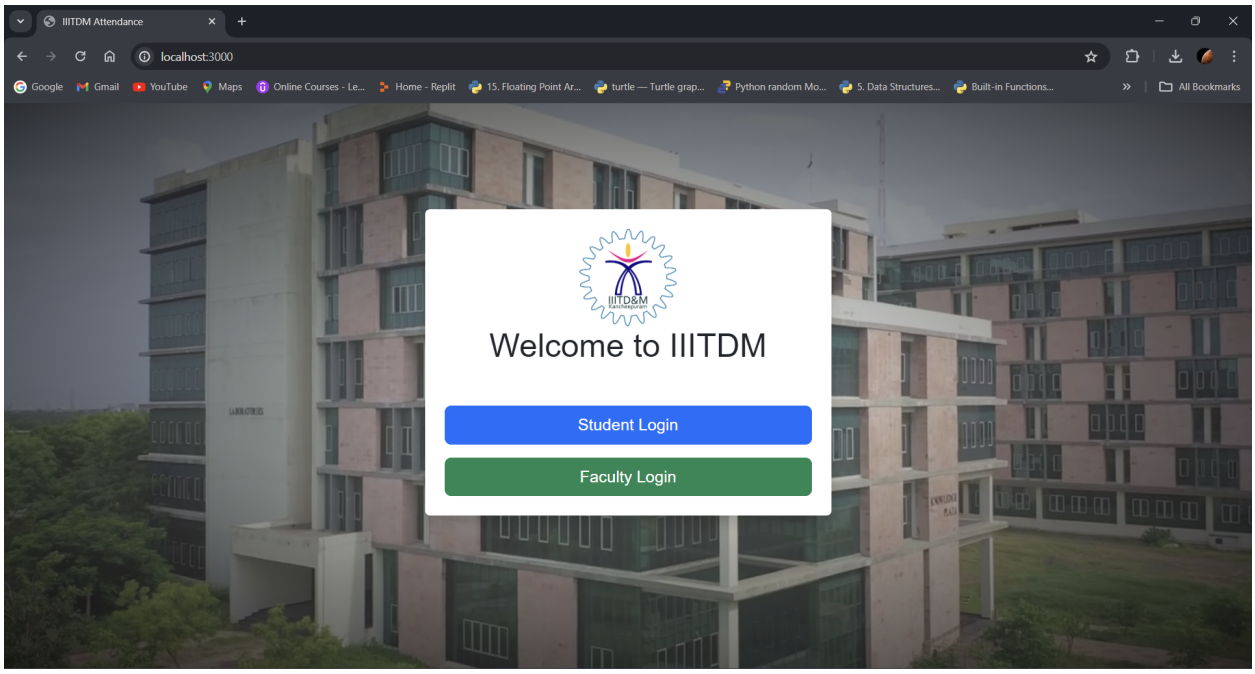
```
mysql> select * from enrolls;
+-----------+----------+
| studentid | courseid |
+-----------+----------+
| s001      | cs001    |
| s001      | cs002    |
| s002      | cs001    |
| s002      | cs002    |
| s003      | cs001    |
| s003      | cs002    |
| s004      | cs001    |
| s004      | cs002    |
| s005      | cs001    |
| s005      | cs002    |
| s006      | ec001    |
| s006      | ec002    |
| s007      | ec001    |
| s007      | ec002    |
| s008      | ec001    |
| s008      | ec002    |
| s009      | ec001    |
| s009      | ec002    |
| s010      | ec001    |
| s010      | ec002    |
| s011      | ec001    |
| s011      | ec002    |
| s012      | ec001    |
| s012      | ec002    |
| s013      | ec001    |
| s013      | ec002    |
| s014      | ec001    |
| s014      | ec002    |
| s015      | ec001    |
| s015      | ec002    |
| s016      | me001    |
| s016      | me002    |
| s017      | me001    |
| s017      | me002    |
| s018      | me001    |
| s018      | me002    |
| s019      | me001    |
| s019      | me002    |
| s020      | me001    |
| s020      | me002    |
+-----------+----------+
```

Front-End:

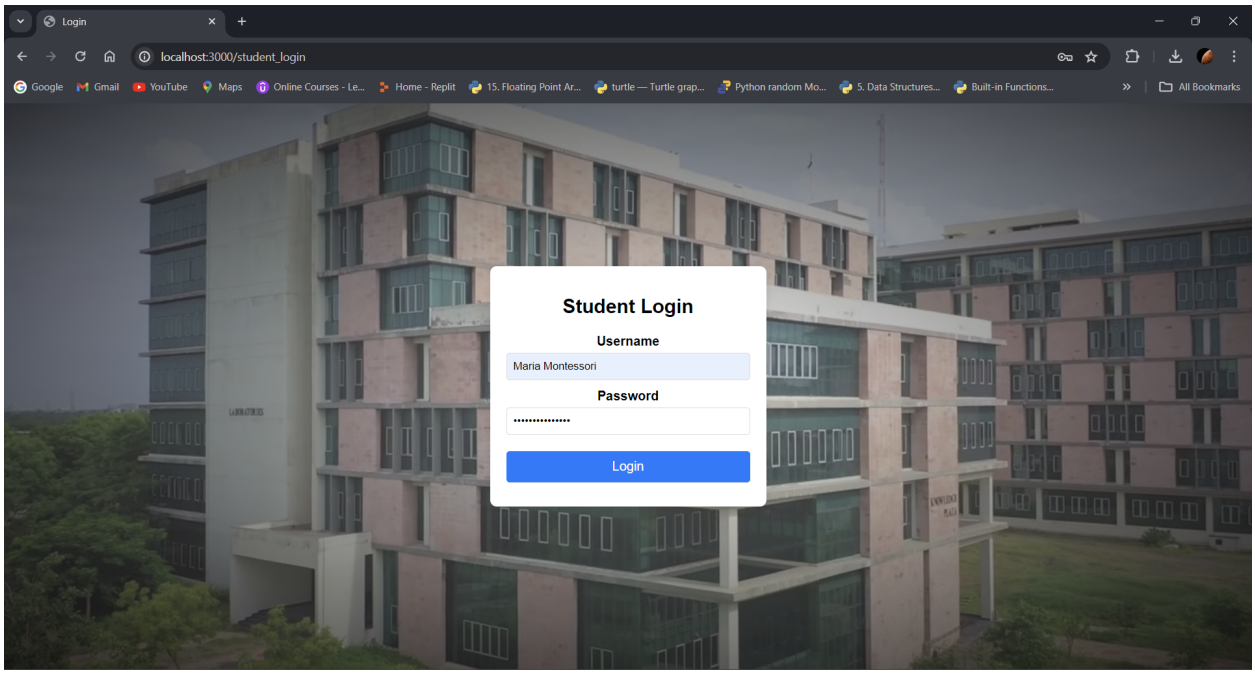
- Running the application using "nodemon". It automatically restarts the node application when a file change occurs



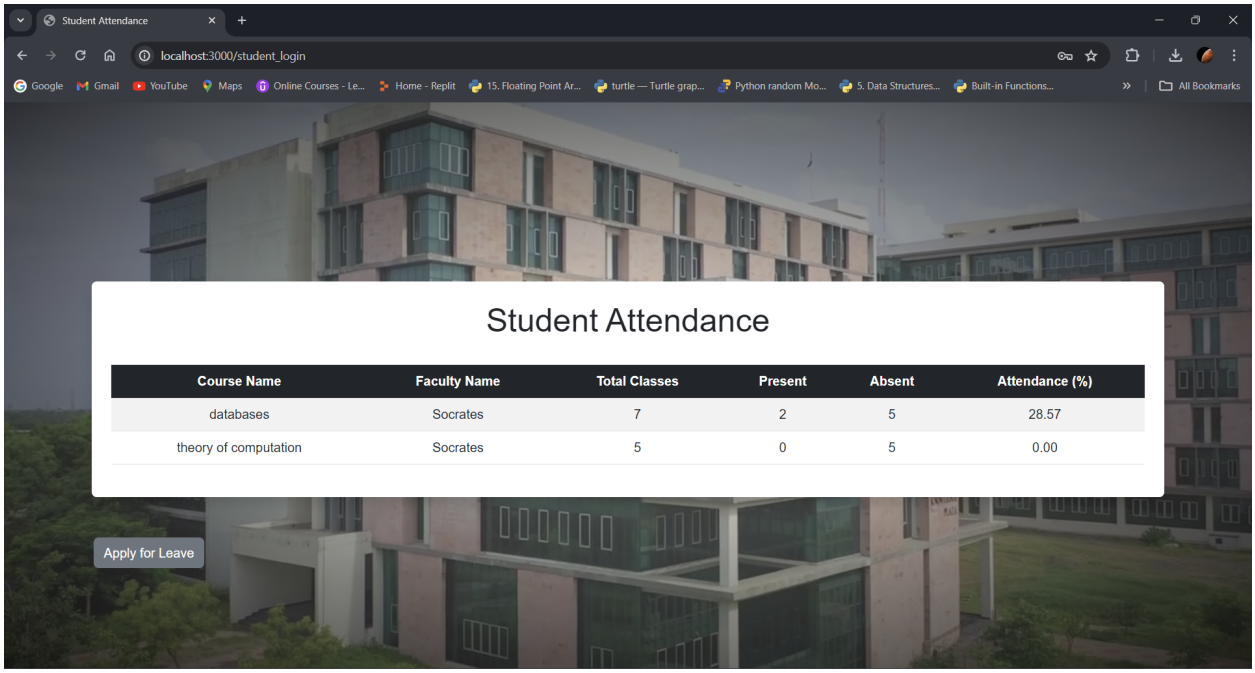
- Landing Page(Login). Either a student or a faculty can login using their credentials



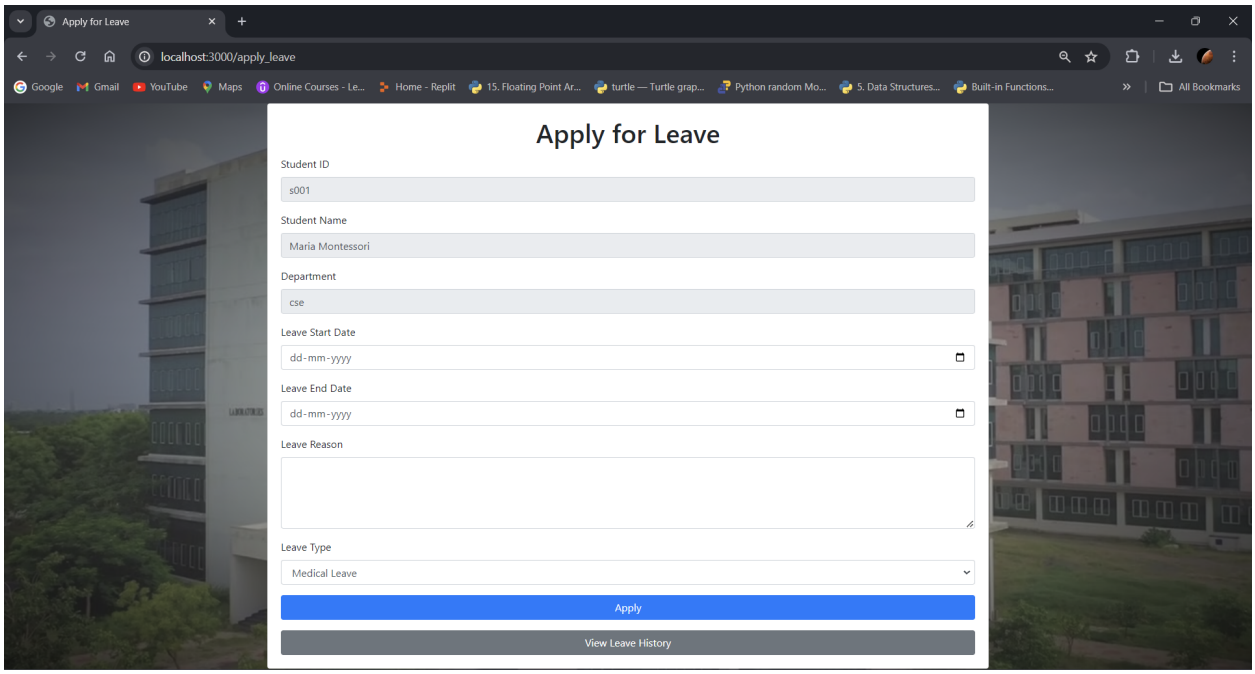
- Student login Page



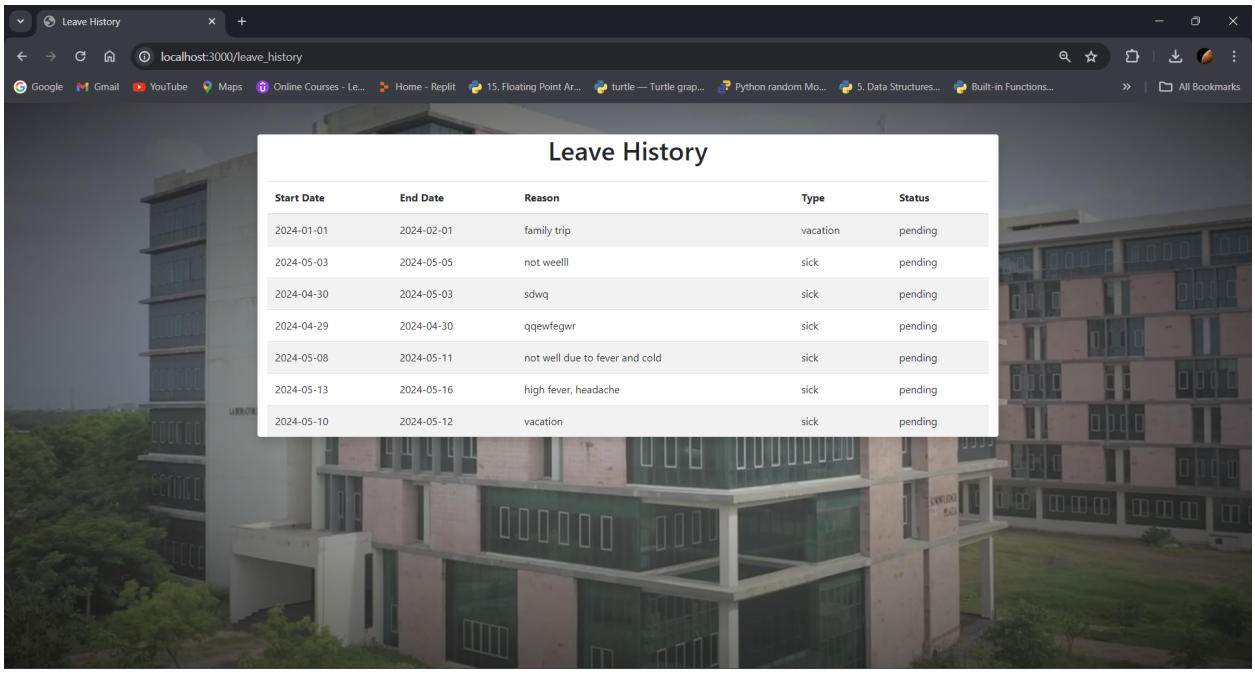
- Student attendance Page showing the attendance of the student in their enrolled courses.



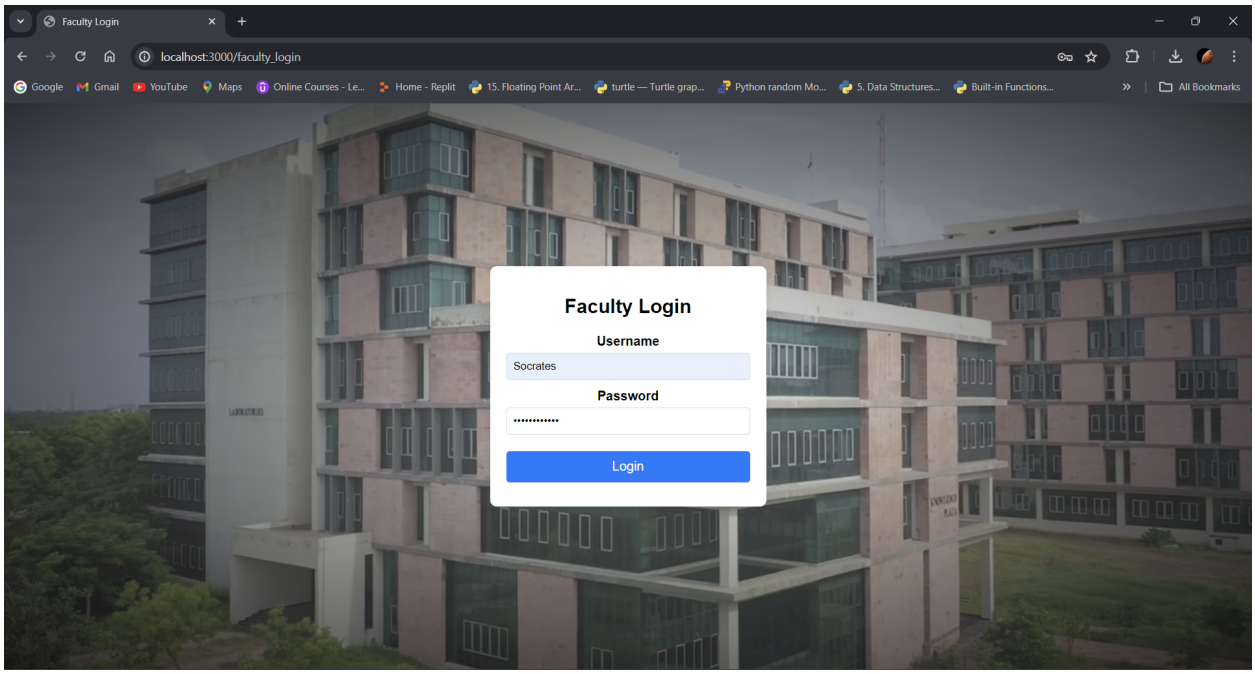
- Leave Application Page



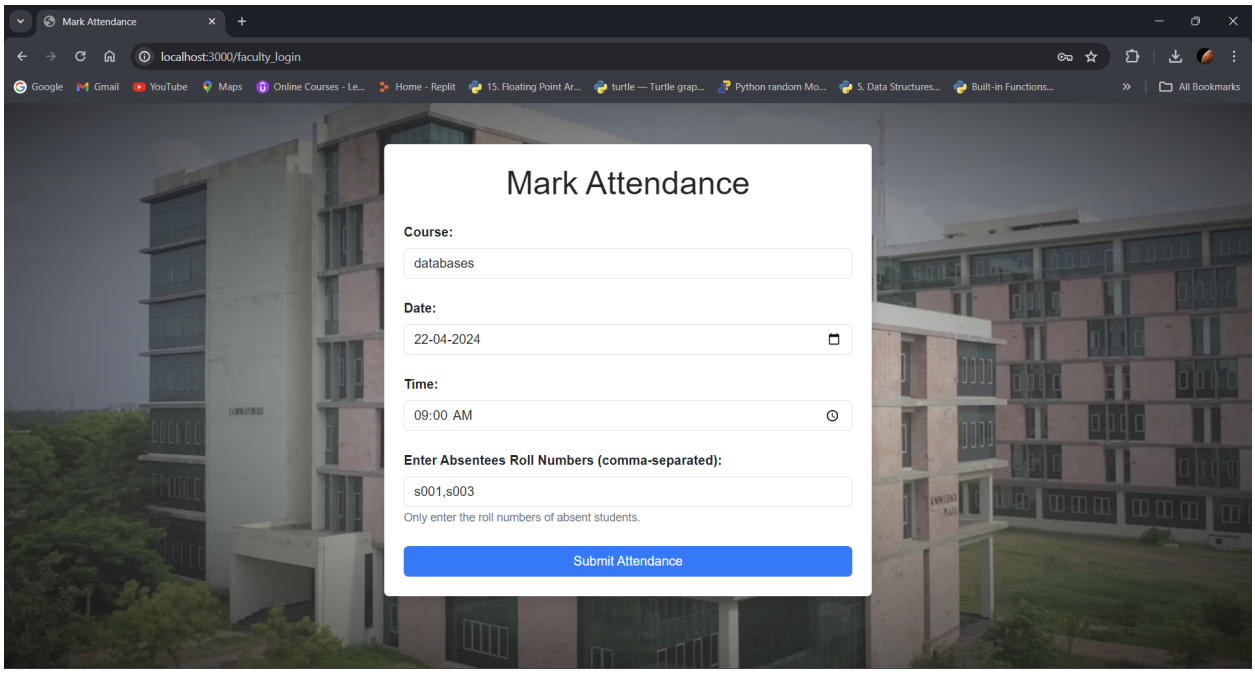
- Leave History Page



- Faculty Login Page



- Attendance Marking Page



Technology Stack used:

1. Front-End:
    - HTML : To design the structure of the webpages
    - CSS : To style the webpages
    - BootStrap : An external framework used for styling the webpages.
    - Embedded JavaScript (EJS) : A templating language used to generate HTML markup with plain javascript.
2. Back-End:
    - JavaScript : A scripting language used to add functionalities to the webpages
    - NodeJS : A javascript runtime environment used for executing javascript outside a web browser.
    - ExpressJS : Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application.
    - MySQL : A Relational Database Management System. Used for storing various records.

Methodology:

The front end has been created and designed using HTML and CSS. HTML has been used to give structure to the website. CSS and along with BootStrap, an external CSS framework are used for styling, positioning of elements of different webpages.

Coming to the Back-end the website utilises javascript, NodeJS, ExpressJS for the creation of server which handle the requests coming from the webpages. Nodemon is an NodeJs framework which automatically restarts the .js file whenver there are any file changes occuring the directory.

Using ExpressJS we create a server and host our website on the port 3000(say). So upon loading the website, we render the landing page of the application by sending a GET request to the server. Upon logging in by the student or faculty the website sends a POST request to the server. Then the server sends the appropriate webpage as a response for student and faculty. The website sends appropriate GET or POST requests based on the necessity of the user. If something is to be shows or updated in the webpage, sending data securing in the request body (in case if forms) we use POST. GET is for fetching data.

Using MySQL Querying we obtain the details of the student attendance by joining the required tables from the database. To Obtain the present classes of the student we use the following query:

```
SELECT
    course.coursename,
    CONCAT(instructor.instructorfname, ' ', instructor.instructormname,
        ' ', instructor.instructorlname) AS faculty_name,
    COUNT(class.classid) AS total_classes,
    SUM(CASE WHEN attends.attendancestatus = 'present'
        THEN 1 ELSE 0 END) AS present_classes
FROM
    enrolls
    JOIN course ON enrolls.courseid = course.courseid
    JOIN instructs ON course.courseid = instructs.courseid
    JOIN instructor ON instructs.instructorid = instructor.instructorid
    JOIN class ON course.courseid = class.courseid AND
        class.instructorid = instructor.instructorid
    LEFT JOIN attends ON class.classid = attends.classid AND
        attends.studentid = ?
WHERE
    enrolls.studentid = ?
GROUP BY
    course.coursename, faculty_name
```

To get courses taught by a paricular instructor

```
SELECT course.courseid, course.coursename
FROM instructs
JOIN course ON instructs.courseid = course.courseid
WHERE instructs.instructorid = ?
```

To get student details enrolled in a particular course:

```
SELECT student.studentid, student.fname, student.lname
FROM enrolls
JOIN student ON enrolls.studentid = student.studentid
WHERE enrolls.courseid = ?
```

To mark attendance of the students wwe only enter the absentees roll number/student id Attendance is marked as present for all other students. This is done by first initialising attendance to "present" for all students and changing to "absent" for only those who are absent

This is done by the following code:

```
absenteesArray= absentees?absentees.split(',').map(id => id.trim()):[];
status= absenteesArray.includes(student.studentid)?'absent':'present';
insertAttendanceQuery = `
    INSERT INTO attends (studentid,classid,attendancestatus,attendancedate)
    VALUES (?, ?, ?, ?)
    ON DUPLICATE KEY UPDATE attendancestatus=VALUES(attendancestatus)
    `;
```

To get the leave requests of a particular student:

```
SELECT DATE_FORMAT(leavestartdate, '%Y-%m-%d') AS leavestartdate,
       DATE_FORMAT(leaveenddate, '%Y-%m-%d') AS leaveenddate,
       leavetype, leavereason, leavestatus
FROM leaverequest
WHERE studentid = ?
```

Below is code for Back-End: (app.js file)

```
const express = require('express');
const mysql = require('mysql');
const app = express();
const session = require('express-session');
const cors = require('cors');
const port = 3000;

app.set('view engine', 'ejs');
app.use(express.static('public'));
app.use(express.urlencoded({ extended: true }));
app.use(cors());

app.use(session({
    secret: 'your secret key',
    resave: false,
    saveUninitialized: true
}));



const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'Sumakar@mysql5',
    database: 'attendance'
});

db.connect(err => {
    if (err) {
```

```
        console.error('Database connection failed: ' + err.stack);
        return;
    }
        console.log('Connected to database.');
});

app.get('/', (req, res) => {
    res.render('landing');
});

app.get('/student_login', (req, res) => {
    res.render('student_login');
});

app.get('/faculty_login', (req, res) => {
    res.render('faculty_login');
});


app.post('/student_login', (req, res) => {
    const { username, password} = req.body;


    const query = 'SELECT * FROM user WHERE username = ? AND
        passkey = ? AND usertype = "student"';
    db.query(query, [username, password], (err, results) => {
        if (err) throw err;



        if (results.length > 0) {
        const userId = results[0].userid;
        req.session.userid = userId;
        const studentQuery = 'SELECT * FROM student WHERE userid = ?';

        db.query(studentQuery, [userId], (err, studentResults) => {
            if (err) throw err;

            if (studentResults.length > 0) {
            const studentId = studentResults[0].studentid;
            const attendanceQuery = `
                SELECT
                course.coursename,
                CONCAT(instructor.instructorfname, ' ',
                    instructor.instructormname, ' ',
                    instructor.instructorlname) AS faculty_name,
                COUNT(class.classid) AS total_classes,
                SUM(CASE WHEN attends.attendancestatus = 'present'
                    THEN 1 ELSE 0 END)
                    AS present_classes
                FROM
                enrolls
```

```
                JOIN course ON enrolls.courseid = course.courseid
                JOIN instructs ON course.courseid = instructs.courseid
                JOIN instructor ON instructs.instructorid =
                    instructor.instructorid
                JOIN class ON course.courseid = class.courseid AND
                    class.instructorid = instructor.instructorid
                LEFT JOIN attends ON class.classid = attends.classid AND
                    attends.studentid = ?
                WHERE
                enrolls.studentid = ?
                GROUP BY
                course.coursename, faculty_name
            `;

            db.query(attendanceQuery, [studentId, studentId],
                (err, attendanceResults) => {
                    if (err) throw err;

                res.render('student_attendance', {student:studentResults[0],
                    attendance:attendanceResults });
            });
            } else {
            res.send('Student not found.');
            }
        });
        } else {
        res.send('Invalid username or password');
        }
    });
});


app.post('/faculty_login', (req, res) => {
    const { username, password, userid } = req.body;

    const query = 'SELECT * FROM user WHERE username = ? AND
        passkey = ? AND usertype = "teach"';
    db.query(query, [username, password], (err, results) => {
        if (err) throw err;

        if (results.length > 0) {
        const userId = results[0].userid;
        const facultyQuery = 'SELECT * FROM instructor WHERE userid = ?';

        db.query(facultyQuery, [userId], (err, facultyResults) => {
            if (err) throw err;

            if (facultyResults.length > 0) {
            const instructorId = facultyResults[0].instructorid;
            const coursesQuery = `
                SELECT course.courseid, course.coursename
                FROM instructs
```

```
                JOIN course ON instructs.courseid = course.courseid
                WHERE instructs.instructorid = ?
            `;

            db.query(coursesQuery, [instructorId], (err, courses) => {
                if (err) throw err;

                res.render('mark_attendance', { instructorid: instructorId,
                    courses: courses });
            });
            } else {
            res.send('Faculty not found.');
            }
        });
        } else {
        res.send('Invalid username or password');
        }
    });
});



app.get('/get_students', (req, res) => {
    const { courseid } = req.query;

    const studentsQuery = `
        SELECT student.studentid, student.fname, student.lname
        FROM enrolls
        JOIN student ON enrolls.studentid = student.studentid
        WHERE enrolls.courseid = ?
    `;

    db.query(studentsQuery, [courseid], (err, results) => {
        if (err) throw err;

        res.json(results);
    });
});



// Add this route to handle the form submission from mark_attendance.ejs
app.post('/mark_attendance', (req, res) => {
    const { instructorid, courseid, date, time, absentees } = req.body;

// Fetch all students enrolled in the selected course
    const enrolledStudentsQuery = `
        SELECT s.studentid
        FROM enrolls e
        JOIN student s ON e.studentid = s.studentid
        WHERE e.courseid = ?
    `;
```

```
    db.query(enrolledStudentsQuery, [courseid], (err, students) => {
        if (err) throw err;

        // Parse absentees from the request
        const absenteesArray=absentees?absentees.split(',').map(id=>id.trim()):[];

        // Fetch the class ID for the selected course,
        // instructor, date, and time
        const classQuery = `
        SELECT classid FROM class
        WHERE courseid = ? AND instructorid = ? AND
            starttime <= ? AND endtime >= ?
        `;

        db.query(classQuery, [courseid, instructorid,
            time, time], (err, classResults) => {
        if (err) throw err;

        if (classResults.length > 0) {
            const classid = classResults[0].classid;

            // Mark all students as present initially
            students.forEach(student => {
            const status =
                absenteesArray.includes(student.studentid)?'absent':'present';
            const insertAttendanceQuery = `
                INSERT INTO
                    attends (studentid,classid,attendancestatus,attendancedate)
                VALUES (?, ?, ?, ?)
                ON DUPLICATE KEY UPDATE
                    attendancestatus = VALUES(attendancestatus)
            `;

            db.query(insertAttendanceQuery,
                [student.studentid, classid, status, date], (err, results) => {
                if (err) throw err;
            });
            });

            res.send('Attendance marked successfully.');
        } else {
            res.send('Class not found.');
        }
        });
    });
});



app.get('/apply_leave', (req, res) => {
    const userid = req.session.userid;
```

```
    const query = `
    SELECT studentid
    FROM student
    WHERE userid = ?
    `;

    var studentID = ""
    db.query(query, [userid], (err, results) => {
        if (err) {
        console.error('Error retrieving student ID:', err);
        res.sendStatus(500);
        return;
        }

        if (results.length === 0) {
        console.log('Student ID not found for user ID:', userid);
        res.sendStatus(404);
        return;
        }

        studentID = results[0].studentid;
    });

    const studentQuery = `
        SELECT s.fname, s.lname, s.studentid, s.department
        FROM student s
        WHERE s.userid = ?
    `;
    db.query(studentQuery, [userid], (err, studentResult) => {
        if (err) {
            console.error('Error fetching student details:', err);
            res.sendStatus(500);
            return;
        }

        if (!studentResult[0]) {
            res.render('apply_leave', { student: null });
        } else {
            res.render('apply_leave', { student: studentResult[0] });
        }
    });
 });




app.get('/apply_leave', (req, res) => {
    const userid = req.session.userid;

    var studentID = ""
    db.query(query, [userid], (err, results) => {
        if (err) {
```

```
        console.error('Error retrieving student ID:', err);
        res.sendStatus(500);
        return;
        }

        if (results.length === 0) {
        console.log('Student ID not found for user ID:', userid);
        res.sendStatus(404);
        return;
        }

        studentID = results[0].studentid;
        req.session.studentID = studentID
    });

    const studentQuery = `
        SELECT s.fname, s.lname, s.studentid, s.department
        FROM student s
        WHERE s.userid = ?
    `;
    db.query(studentQuery, [userid], (err, studentResult) => {
        if (err) {
        console.error('Error fetching student details:', err);
        res.sendStatus(500);
        return;
        }

        if (!studentResult || studentResult.length === 0) {
        console.error('No student details found.');
        res.render('apply_leave', { student: null });
        return;
        }

        res.render('apply_leave', { student: studentResult[0] });
    });
});




app.post('/apply_leave', (req, res) => {
    const userid = req.session.userid;
    const query = `
        SELECT studentid
        FROM student
        WHERE userid = ?
    `;
    db.query(query, [userid], (err, results) => {
        if (err) {
        console.error('Error retrieving student ID:', err);
        res.sendStatus(500);
```

```
        return;
        }
        if (results.length === 0) {
        console.log('Student ID not found for user ID:', userid);
        res.sendStatus(404);
        return;
        }
        const studentID = results[0].studentid;

        const { startDate, endDate, reason, type } = req.body;

        const latestLeaveRequestQuery = `
        SELECT leavereqid
        FROM leaverequest
        ORDER BY leavereqid DESC
        LIMIT 1
        `;
        db.query(latestLeaveRequestQuery, (err, results) => {
        if (err) {
            console.error('Error retrieving latest leave request ID:', err);
            res.sendStatus(500);
            return;
        }

        let latestRequestId = 0;
        if (results.length > 0) {
            latestRequestId = parseInt(results[0].leavereqid.substr(1));
        }

        const newRequestId = 'l' + ('000' + (latestRequestId + 1)).slice(-3);

        const stts = 'pending';

        const insertLeaveQuery = `
            INSERT INTO leaverequest (leavereqid, studentid, leavestartdate,
                leaveenddate, leavetype, leavereason, leavestatus)
            VALUES (?, ?, ?, ?, ?, ?, ?)
        `;

        db.query(insertLeaveQuery, [newRequestId, studentID,
            startDate, endDate, type, reason, stts], (err, result) => {
            if (err) {
            console.error('Error inserting leave request:', err);
            res.sendStatus(500);
            return;
            }

            res.send('Leave request inserted successfully');
        });
        });
});
});
```

```
app.get('/leave_history', (req, res) => {
    const userid = req.session.userid;

    const query1 = `
        SELECT studentid
        FROM student
        WHERE userid = ?
    `;

    db.query(query1, [userid], (err, results) => {
        if (err) {
        console.error('Error retrieving student ID:', err);
        res.sendStatus(500);
        return;
        }

        if (results.length === 0) {
        console.log('Student ID not found for user ID:', userid);
        res.sendStatus(404);
        return;
        }

        const studentID = results[0].studentid;

        const query = `
        SELECT DATE_FORMAT(leavestartdate, '%Y-%m-%d') AS leavestartdate,
                DATE_FORMAT(leaveenddate, '%Y-%m-%d') AS leaveenddate,
                leavetype, leavereason, leavestatus
        FROM leaverequest
        WHERE studentid = ?
        `;

        db.query(query, [studentID], (err, results) => {
        if (err) {
            console.error('Error retrieving leave history:', err);
            res.sendStatus(500);
            return;
        }
        res.render('leave_history', { leaves: results });
        });
    });
});


app.listen(port, () => {
    console.log(`Server running at http://localhost:${port}`);
});
```