Database Project
Lecture Hall Booking System

Project Id: 6

S.No.	Name	Roll Number	Email
1	Anitej Jain	220153	anitej22@iitk.ac.in
2	Ayushmaan Jay Singh	220276	ayushmaanj22@iitk.ac.in
3	Ayan Gupta	220258	ayangupta22@iitk.ac.in
4	Manan Kumar	220607	manankumar22@iitk.ac.in
5	Nandini Akolkar	220692	anandini22@iitk.ac.in

April 24, 2025

1 Introduction

The LHC Booking System is a web application that streamlines the management and reservation of facilities within a lecture hall complex (LHC). Developed primarily in JavaScript and MySQL as the database language, this system provides a friendly interface for administrators and users to handle booking requests. Features currently include event scheduling, real-time updates, and role-based access management to ensure secure and organized operations. The system aims to optimize resource utilization and reduce administrative overhead, making the booking process seamless and practical.

2 Methodology

Utilities Used in Client-Sever Architectural Design Pattern:

- Filess.io for online mysql database hosting.
- Tailwind for modern web development frontend.
- React to build user interfaces out of individual pieces called components.
- Vite, a JavaScript build tool used to significantly speed up web development by leveraging native ES Modules for a faster development experience and optimized production builds.

Figure 1: USAGE : Institutes with no automated booking system

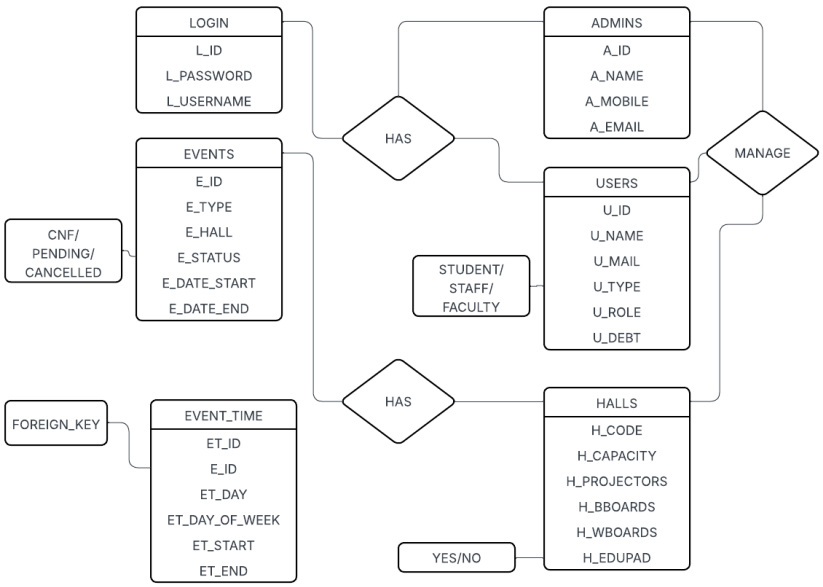


Figure 2: Entity Relationship Diagram

3 How to Run

3.0.1 Install Packages: root and backend directory

```
> npm install
```

3.0.2 Start Server: in backend directory

```
> npm start
```

3.0.3 Persistent relations creation, currently set to online hosted database credentials: in database directory

```
> node run-sql-file.js
```

3.0.4 Host Frontend: root directory

```
> npm run dev
```

4 Code Description and Implementation

GitHub Repository : https://github.com/Vulcan-Fire/CS315-PROJECT.git

4.1 Features

- **User Roles:** Student\Faculty, or Admin with different access levels.
- **Hall Availability Check:** Users can only book for Halls in time slots. And can check the complete time table in calendar tab.
- **Booking System:** Users can request lecture halls for available and desired time slots.
- **Approval Mechanism:** Admins can approve or reject booking requests.
- **Search & Filter:** Users can find halls based on capacity and features.

4.2 SQL DDL PART

```
CREATE TABLE LOGIN (
    L_ID INT PRIMARY KEY AUTO_INCREMENT,
    L_PASSWORD VARCHAR(255) NOT NULL,
    L_USERNAME VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE ADMINS (
    A_ID INT PRIMARY KEY,
    A_NAME VARCHAR(100) NOT NULL,
    A_MOBILE VARCHAR(20),
    A_EMAIL VARCHAR(100) UNIQUE
);

CREATE TABLE USERS (
    U_ID INT PRIMARY KEY AUTO_INCREMENT,
    U_NAME VARCHAR(100) NOT NULL,
    U_MAIL VARCHAR(100) UNIQUE NOT NULL,
    U_TYPE ENUM('STUDENT', 'STAFF', 'FACULTY') NOT NULL,
```

```
    -- U_ROLE field has been removed
    U_DEBT DECIMAL(10, 2) DEFAULT 0
);

CREATE TABLE HALLS (
    H_CODE VARCHAR(20) PRIMARY KEY,
    H_CAPACITY INT NOT NULL,
    H_PROJECTORS INT DEFAULT 0,
    H_BBOARDS INT DEFAULT 0,
    H_WBOARDS INT DEFAULT 0,
    H_EDUPAD ENUM('YES', 'NO') DEFAULT 'NO'
);

CREATE TABLE EVENTS (
    E_ID INT PRIMARY KEY AUTO_INCREMENT,
    E_TYPE VARCHAR(50) NOT NULL,
    E_HALL VARCHAR(20) NOT NULL,
    E_STATUS ENUM('CONFIRMED', 'PENDING', 'CANCELLED') DEFAULT 'PENDING',
    E_DATE_START DATE NOT NULL,
    E_DATE_END DATE NOT NULL,
    FOREIGN KEY (E_HALL) REFERENCES HALLS(H_CODE)
);

CREATE TABLE EVENT_TIME (
    ET_ID INT PRIMARY KEY AUTO_INCREMENT,
    E_ID INT NOT NULL ,
    ET_DAY DATE NOT NULL,
    ET_DAY_OF_WEEK VARCHAR(10) GENERATED ALWAYS AS (DAYNAME(ET_DAY)) STORED,
    ET_START TIME NOT NULL,
    ET_END TIME NOT NULL,
    FOREIGN KEY (E_ID) REFERENCES EVENTS(E_ID)
);

CREATE TABLE LOGIN_ADMIN (
    L_ID INT NOT NULL,
    A_ID INT NOT NULL,
    PRIMARY KEY (L_ID, A_ID),
    FOREIGN KEY (L_ID) REFERENCES LOGIN(L_ID),
    FOREIGN KEY (A_ID) REFERENCES ADMINS(A_ID)
);

CREATE TABLE LOGIN_USER (
    L_ID INT NOT NULL,
    U_ID INT NOT NULL,
    PRIMARY KEY (L_ID, U_ID),
    FOREIGN KEY (L_ID) REFERENCES LOGIN(L_ID),
    FOREIGN KEY (U_ID) REFERENCES USERS(U_ID)
);

CREATE TABLE ADMIN_USER (
    A_ID INT NOT NULL,
    U_ID INT NOT NULL,
    PRIMARY KEY (A_ID, U_ID),
    FOREIGN KEY (A_ID) REFERENCES ADMINS(A_ID),
```

```
    FOREIGN KEY (U_ID) REFERENCES USERS(U_ID)
);

CREATE TABLE USER_EVENT (
    U_ID INT NOT NULL,
    E_ID INT NOT NULL,
    PRIMARY KEY (U_ID, E_ID),
    FOREIGN KEY (U_ID) REFERENCES USERS(U_ID),
    FOREIGN KEY (E_ID) REFERENCES EVENTS(E_ID)
);

CREATE TABLE ADMIN_HALL (
    A_ID INT NOT NULL,
    H_CODE VARCHAR(20) NOT NULL,
    PRIMARY KEY (A_ID, H_CODE),
    FOREIGN KEY (A_ID) REFERENCES ADMINS(A_ID),
    FOREIGN KEY (H_CODE) REFERENCES HALLS(H_CODE)
);

CREATE INDEX idx_event_status ON EVENTS(E_STATUS);
CREATE INDEX idx_event_dates ON EVENTS(E_DATE_START, E_DATE_END);
CREATE INDEX idx_user_type ON USERS(U_TYPE);
CREATE INDEX idx_hall_capacity ON HALLS(H_CAPACITY);
```

4.3 MAIN QUERY CODE SNIPPETS

4.3.1 To Be Noted

- ? is acts as placeholder and can be thought be replaced by the 'obvious' term during execution.

4.3.2 User's Events

```
    SELECT e.*,
       GROUP_CONCAT(DISTINCT et.ET_DAY, ' ', et.ET_START, '-', et.ET_END SEPARATOR ',
    ') as TIME_SLOTS
    FROM EVENTS e
    JOIN USER_EVENT ue ON e.E_ID = ue.E_ID
    LEFT JOIN EVENT_TIME et ON e.E_ID = et.E_ID
    WHERE ue.U_ID = ?
    GROUP BY e.E_ID
    ORDER BY e.E_DATE_START DESC
```

4.3.3 Event by E_ID

```
    SELECT e.*,
         u.U_NAME as REQUESTER_NAME,
         h.H_CAPACITY, h.H_PROJECTORS, h.H_BBOARDS, h.H_WBOARDS, h.H_EDUPAD
    FROM EVENTS e
```

```
JOIN USER_EVENT ue ON e.E_ID = ue.E_ID
JOIN USERS u ON ue.U_ID = u.U_ID
JOIN HALLS h ON e.E_HALL = h.H_CODE
WHERE e.E_ID = ?
```

4.3.4 Hall Availability Data

```
SELECT
  e.E_ID, e.E_TYPE, e.E_STATUS, e.E_DATE_START, e.E_DATE_END,
  et.ET_DAY, et.ET_DAY_OF_WEEK, et.ET_START, et.ET_END
FROM EVENTS e
JOIN EVENT_TIME et ON e.E_ID = et.E_ID
WHERE e.E_HALL = ?
  AND e.E_STATUS != 'CANCELLED'
  AND (
    (e.E_DATE_START <= ? AND e.E_DATE_END >= ?)
    OR (e.E_DATE_START <= ? AND e.E_DATE_END >= ?)
    OR (e.E_DATE_START >= ? AND e.E_DATE_END <= ?)
  )
ORDER BY et.ET_DAY, et.ET_START
```

4.3.5 User Login Check

```
SELECT l.L_ID, l.L_USERNAME, l.L_PASSWORD, u.U_ID, u.U_NAME, u.U_MAIL, u.U_TYPE
FROM LOGIN l
JOIN LOGIN_USER lu ON l.L_ID = lu.L_ID
JOIN USERS u ON lu.U_ID = u.U_ID
WHERE l.L_USERNAME = ?
```

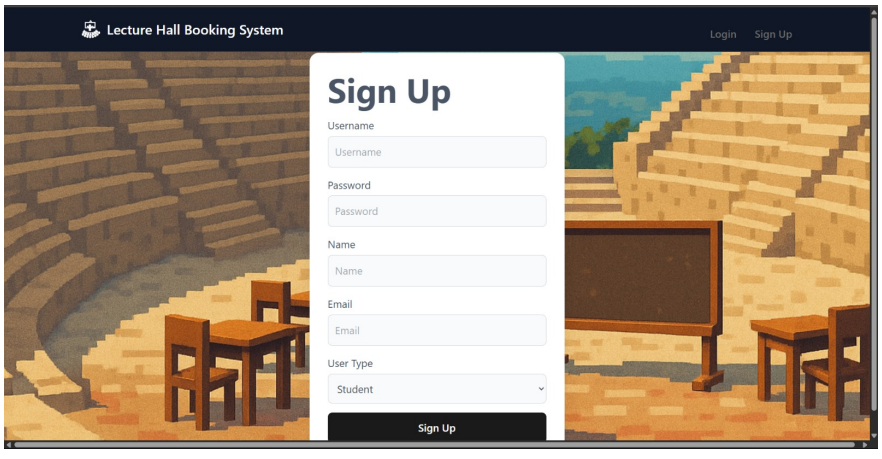


Figure 3: Sign-up page

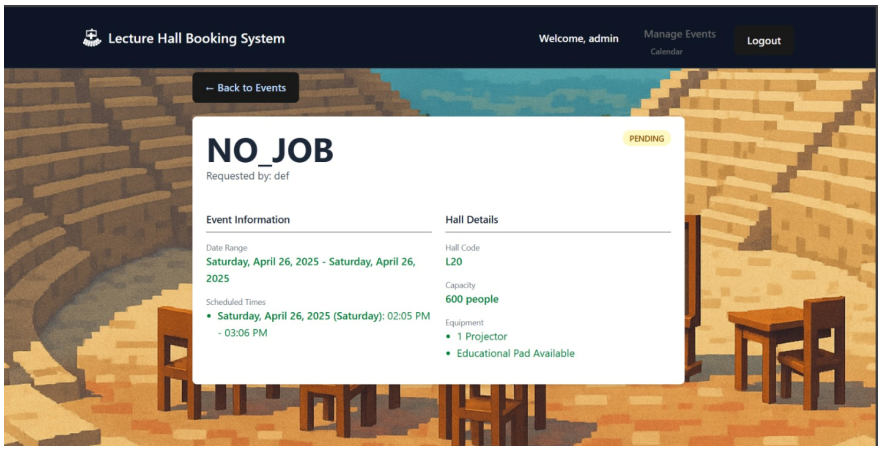
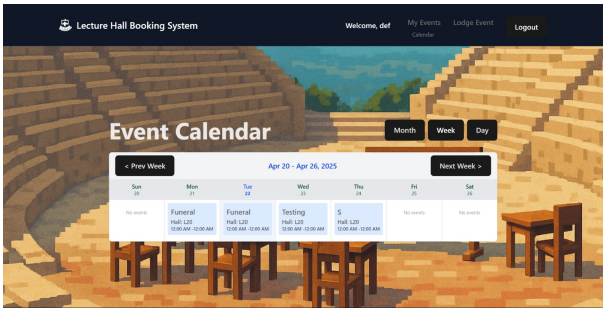
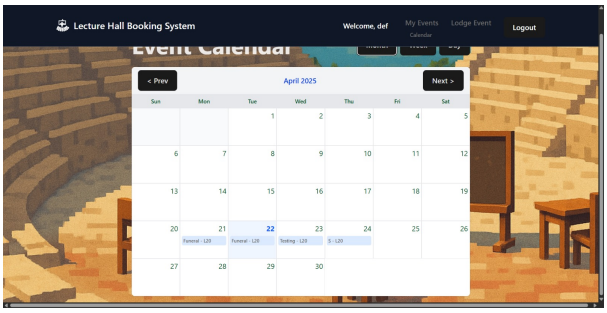


Figure 4: Job Detailed View



(a) Weekly Calendar



(b) Monthly Calendar

Figure 5: Calendar

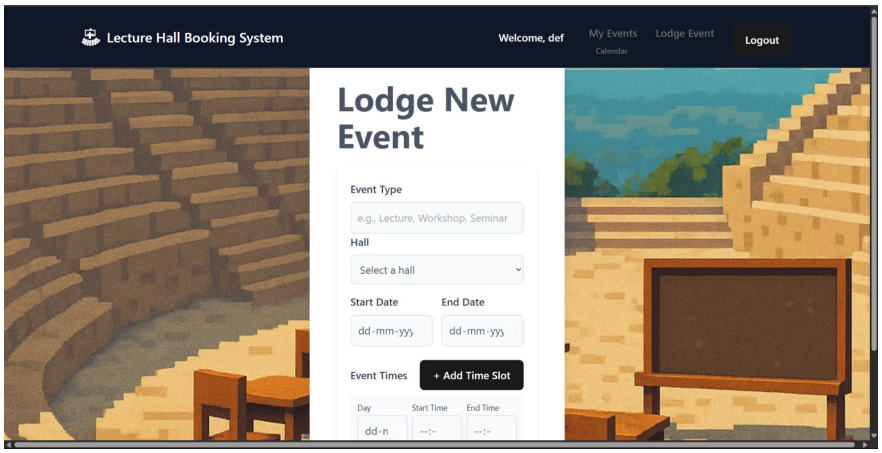


Figure 6: Lodging Page

5 #TO_DO_NEXT

Improvement Areas and Features to be added:

- Debt notification to be sent to User's mail.

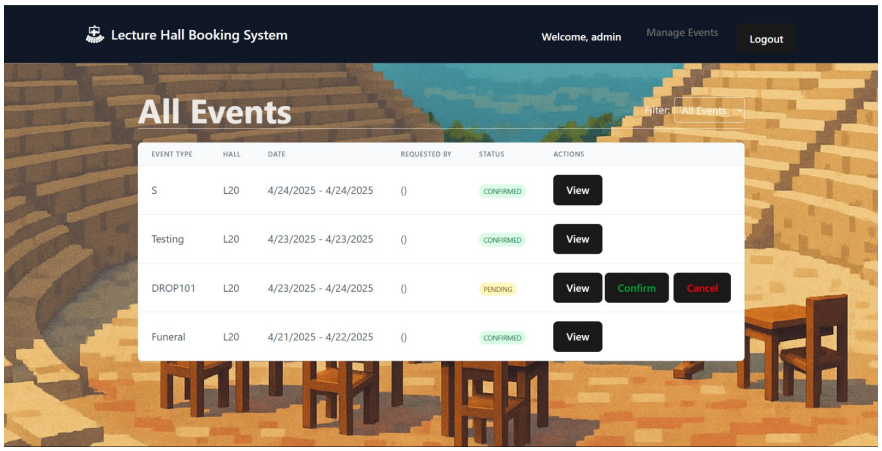


Figure 7: Approval Dashboard

- Calendar for a particular hall.
- Feature to allow different Admins overlooking different set of halls.
- Improved display of hall amenities to the user.
- Researching and implementing database level triggers.

6 Contributions

*DOCUMENTATION CONTENT WAS DECIDED ON COLLABORATIVELY.

MEMBER	Contributions
Manan Kumar	**Relations:** LOGIN, USERS **Design:** ER diagram, Development Stack, Code Maintenance, Latex Code **Frontend:** AdminEvents.jsx, Calendar.jsx **Backend:** API :: getConfirmedEvents, getUserEvents, server.js
Anitej Jain	**Relations:** ADMINS, ADMIN_USER **Design:** Tailwind CSS, Development Stack, Integration Testing **Frontend:** EventForm.jsx, Events.jsx **Backend:** API:: getEventById, getEventByStatus, getAllEvents
Ayan Gupta	**Relations:** EVENTS, EVENT_TIME **Design:** Tailwind CSS, Relation Index Creation, Manual Testing **Frontend:** Navbar.jsx, EventDetails.jsx, App.js **Backend:** API:: createEvent, updateEventStatus
Aysuhmaan Jay Singh	**Relations:** HALLS, LOGIN_ADMIN **Design:** Tailwind + Root CSS, Relation Index Creation, Code Cleaning **Frontend:** Login.jsx, SignUp.jsx **Backend:** API:: getAllHalls, getHallAvailability AND jwt middleware
Nandini Akolkar	**Relations:** LOGIN_USER, USER_EVENT, ADMIN_HALL **Design:** ER diagram, Report Template and Editing, Manual Testing **Frontend:** ProtectedRoute.jsx, AdminRoute.jsx **Backend:** API:: login, signup AND run-sql-file.js