COLLEGE CODE:[8223]

COLLEGE NAME: [vandayar engineering college]

DEPARTMENT: [Computer science and engineering]

STUDENT NM- ID: [AC686CDA24D3E0FA4E4461E6 489842FC]

ROLL NO:[822323104301]

DATE:[19.09.2025]

Completed the project named as

Phase-2

TECHNOLOGY PROJECT NAME: Event Scheduling App

SUBMITTED BY,

NAME:[HARIHARAN.M]

MOBILE NO:[8072911559]

1. Tech Stack Selection

Frontend (User Interface):

- **React.js - Fast, component-based, reusable Ul.**
- **Bootstrap / Tailwind CSS - For responsive design.**
- **Full Calendar Library - For calendar-based event visualization.**

Backend (Server Side):

- **Node.js (Express.js framework) Lightweight, scalable, event-driven.**
- **Alternative: Django (Python) or Spring Boot (Java) if required.**

Database:

- **MongoDB – NoSQL DB, flexible schema for storing event details.**

- **Alternative: MySQL / PostgreSQL (if relational schema preferred).**

Authentication & Security:

- **JWT (JSON Web Token) for session handling.**

- **Password encryption using bcrypt.**

- **Role-based access control (User /Admin).**

Hosting & Deployment:

- **Cloud: AWS / Google Cloud / Firebase.**

- **CI/CD: GitHub Actions for automated deployment.**

Other Tools:

- **Scheduler: Cron Jobs for reminders.**
- **Push Notifications Firebase Cloud Messaging (FCM**

2. UI Structure / API Schema Design

UI Structure (Screens / Pages):

1. **Login / Signup Page – Secure authentication for users.**
2. **User Dashboard – Shows upcoming & past events.**
3. **Event Creation Page – Form to create/edit/delete events.**
4. **Calendar View – Monthly/Weekly/Daily view with event markers.**

5. **Notifications Page – List of reminders & alerts.**

6. **Profile Page – Manage personal details, settings.**

7. **Admin Panel (Optional) – Manage users & overall events.**

API Schema Design (REST APIs):

User APIs

- **POST /api/users/register → Register new user**
- **POST /api/users/login → Login with authentication**
- **GET /api/users/:id → Fetch user profile**

Event APIs

- **POST /api/events → Create event**
- **GET /api/events → Get all events for logged-in user**
- **GET /api/events/:id → Get event details by ID**
- **PUT /api/events/:id → Update event details**
- **DELETE /api/events/:id → Delete event**

Notification APIs

- **POST /api/notifications → Create reminder for event**
- **GET /api/notifications/:userId → Get user reminders**

3. Data Handling Approach

Event Data Structure (MongoDB Example):

```
{
 “eventId”: “E101”,
 “title”: “Project Review Meeting”,
 “date”: “2025-10-05”,
 “time”: “10:00 AM”,
 “venue”: “Seminar Hall”,
 “createdBy”: “User123”,
 “participants”: [“UserA”, “UserB”],
 “reminder”: “30 minutes before”
}
```

User Data Structure:

```
{
 “userId”: “U123”,
 “name”: “John Doe”,
 “email”: john@example.com,
 “password”: “hashed_password”,
 “role”: “student”
}
```

4. Component / Module Diagram

Modules in System:

1. **Authentication Module → Login, signup, session handling.**

2. **Event Management Module → CRUD operations on events.**

3. **Calendar Module → Event visualization in calendar format.**

4. **Notification Module → Event reminders via email/SMS/push.**

5. **User Module → Profile management.**

6. **Admin Module (optional) → Manage users & system monitoring.**

5. Basic Flow Diagram

User Flow:

User → Login/Signup → Dashboard → Create Event → Save in DB → Display in Calendar → Trigger Reminder → Notification Sent

Detailed Steps:

1. **User registers/login.**

2. **Dashboard shows list of events.**

3. **User creates a new event (with title, date, participants, reminder).**

4. **Data stored in database & displayed in UI calendar.**

5. **Scheduler checks DB at regular intervals.**

6. **Reminder sent to user (via email/SMS/push notification).**