---

1. Overall Architecture

Your architecture consists of three main layers:

- **Frontend:** Built with React.js and styled with CSS.
- **Backend:** Developed using Node.js with Express for RESTful APIs.
- **Database:** MongoDB for data storage.

---

2. Frontend Structure

The frontend will serve as the user interface, interacting with the backend via API calls.

Key Components:

1. **Home Page**
    - Login/Signup for Admin, Teachers, and Students.
2. **Dashboards:**
    - **Admin Dashboard**
        - Class Scheduling Form.
        - Timings Management Section.
    - **Teacher Dashboard**
        - Timetable Viewer.
        - Assignment Upload Form.
        - Generate Online Class Links Form.
    - **Student Dashboard**
        - Timetable Viewer.
        - Assignment Download Section.
        - Virtual Class Links Viewer.
3. **Chatbot Integration:**
    - A React component integrated with Falcon 7B API for query handling.

API Calls from Frontend:

- Fetch and display timetables (GET /api/timetable).
- Post assignments and materials (POST /api/materials).
- Fetch attendance and schedule details (GET /api/schedule).

---

3. Backend Structure

The backend will handle business logic, APIs, and database interactions.

Endpoints (API):

1. **Authentication:**
    - POST /api/login: Authenticates users.
    - POST /api/register: Registers users.
2. **Admin Functions:**
    - POST /api/classes: Schedule a new class.
    - PUT /api/classes/:id: Update class schedule.
3. **Teacher Functions:**
    - GET /api/timetable: Get teacher's schedule.
    - POST /api/materials: Upload assignments/materials.
    - POST /api/meeting-links: Create meeting links.
4. **Student Functions:**
    - GET /api/timetable: Get class schedule.
    - GET /api/materials: Access assignments/materials.
    - GET /api/meeting-links: Get meeting links.
5. **Chatbot Endpoint:**
    - POST /api/chatbot: Process chatbot queries.
6. **Notifications:**
    - POST /api/notifications: Send alerts and reminders.

---

4. Database Design

Using MongoDB, the data will be stored in collections.

Collections:

1. **Users:**
    - Schema: { name, email, password, role (admin/teacher/student), ... }
2. **Classes:**
    - Schema: { subject, teacher, time, students: [student_ids], link, ... }
3. **Materials:**
    - Schema: { teacher_id, subject, files: [file_links], ... }

4. **Timetable:**
    - Schema: { user_id, schedule: [{ date, time, subject }] }
5. **Attendance:**
    - Schema: { class_id, student_id, status (present/absent), ... }
6. **Chatbot Queries:**
    - Schema: { user_id, query, response, timestamp }

---

5. Connecting Frontend with Backend

Use RESTful APIs to connect the frontend with the backend.

Integration Flow:

1. **API Calls in React:**
   Use axios or fetch to make API requests.
2. import axios from 'axios';
3. axios.get('/api/timetable').then(response => {
4. setTimetable(response.data);
5. });
6. **Express Routes:**
   Define routes in Node.js to handle frontend requests.
7. app.get('/api/timetable', async (req, res) => {
8. const timetable = await Timetable.find({ userId: req.user.id });
9. res.json(timetable);
10. });
11. **Database Query:**
    Use Mongoose to interact with MongoDB.
12. const timetable = await Timetable.find({ userId: req.params.id });
13. return timetable;

---

6. Real-Time Features

Integrate real-time updates using **WebSockets** or **Socket.IO** for notifications and timetable updates.

Example:

1. **Backend (Express):**
2. const io = require('socket.io')(server);

3. io.on('connection', (socket) => {
4. socket.emit('message', 'Welcome to Classify!');
5. });
6. **Frontend (React):**
7. import io from 'socket.io-client';
8. const socket = io('http://localhost:3000');
9. socket.on('message', msg => console.log(msg));

---

7. Deployment

1. **Frontend:**
    - Deploy using Vercel, Netlify, or AWS Amplify.
2. **Backend:**
    - Deploy using Heroku, AWS EC2, or DigitalOcean.
3. **Database:**
    - Host MongoDB using Atlas for cloud-based storage.

---

8. Directory Structure

```
project-root/
├── frontend/
|   ├── src/
|   |   ├── components/
|   |   ├── pages/
|   |   ├── services/  // API integrations
|   └── package.json
├── backend/
|   ├── controllers/
|   ├── models/
|   ├── routes/
|   └── server.js
├── database/
|   └── mongodb-schema.js
```

└── README.md