Documentation for Learning Management System (LMS)

Features by User Role

Student

1. **Dashboard:** Overview of pending tasks, recent activities, and performance insights.
2. **Attendance Analysis:** Track attendance trends and percentages.
3. **Recordings/Live Sessions:** Access past class recordings and join live sessions.
4. **Performance Analysis:** View grades and performance metrics.
5. **Community Forum:** Interact with peers and trainers in discussion forums.
6. **Tasks/Assignments:** Submit assignments and view task statuses.
7. **Leave Management:** Apply for leave and track approvals.
8. **Materials/Resources:** Access course materials and resources.
9. **Certifications/Achievements:** View and download earned certifications.

Trainer

1. **Dashboard:** Track student progress, upcoming tasks, and live sessions.
2. **Attendance Analysis:** Manage and analyze attendance records.
3. **Recordings/Live Sessions:** Host and access session recordings.
4. **Performance Analysis:** Evaluate student performance and provide feedback.
5. **Community Forum:** Engage with students in discussions.
6. **Tasks/Assignments:** Create, assign, and review tasks.
7. **Leave Management:** Apply for and manage personal leaves.
8. **Materials/Resources:** Upload and manage study materials.
9. **Certifications/Achievements:** Issue certifications to students.

TPO

1. **Dashboard:** Monitor placement activities and student performance.
2. **Attendance Analysis:** Evaluate student attendance data.
3. **Performance Analysis:** Review overall performance and readiness for placements.
4. **Certifications/Achievements:** Track certifications and student achievements.

Admin

1. **Dashboard:** Administer the platform with insights into all user roles.
2. **Attendance Analysis:** Monitor overall attendance metrics.
3. **Recordings/Live Sessions:** Manage session recordings and live events.
4. **Performance Analysis:** Track performance across roles and batches.
5. **Community Forum:** Moderate discussions.
6. **Tasks/Assignments:** Assign or review tasks for all users.
7. **Leave Management:** Manage leave requests for trainers and students.
8. **Materials/Resources:** Oversee materials uploaded by trainers.
9. **Certifications/Achievements:** Administer certifications for students and trainers.

Technology Stack

Frontend

- **React.js:** For building a dynamic, component-based UI.
- **Redux/Context API:** For managing global state effectively.
- **Axios:** To handle HTTP requests.
- **React Router:** For single-page navigation.
- **Material-UI/Tailwind CSS:** For responsive design.
- **React Lazy & Suspense:** For lazy loading and optimizing performance.

Backend

- **Node.js:** For handling server-side logic.
- **Express.js:** To create RESTful APIs.
- **MongoDB:** A NoSQL database for scalable data storage.
- **Mongoose:** For schema validation and database interaction.
- **JWT:** For authentication and role-based access control.
- **Multer:** For handling file uploads.
- **Azure Blob Storage** : To store and serve static assets such as course materials, user-uploaded files, images, and videos.

Deployment and DevOps

- **Azure Blob Storage:** For storing static assets (e.g., videos, PDFs).
- **Azure Virtual Machines (VMs):** For hosting the backend application.
- **Azure App Service:** For managed deployment of the frontend.
- **Azure CDN:** For low-latency delivery of static assets.
- **Azure DevOps:** For CI/CD pipelines and automated testing.

Best Practices

Frontend

1. **Code Splitting:** Use lazy loading for large components to improve performance.
2. **State Management:** Use Redux or Context API for consistent state handling.
3. **Responsive Design:** Leverage Material-UI or Tailwind CSS for cross-device compatibility.

Backend

1. **Efficient Queries:** Use indexes in MongoDB and avoid unnecessary database operations.
2. **Authentication:** Use JWT for secure, stateless authentication.

Performance Optimization

1. **Lazy Loading:** Defer loading of non-essential components until required.
2. **CDN Integration:** Serve static assets through Azure CDN for faster delivery.
3. **Rate Limiting:** Prevent API abuse using express-rate-limit.

Security

1. **Encryption:** Use HTTPS and Bcrypt for secure data transfer and password hashing.
2. **Role-Based Access Control:** Grant permissions based on user roles.
3. **Environment Variables:** Store sensitive data like API keys in .env files.

Development Workflow

1. **Requirements Gathering:** Collaborate with stakeholders to finalize features.
2. **Design:** Develop a scalable architecture.
3. **Development:** Implement features iteratively.
4. **Testing:** Use tools like Jest for unit tests and Postman for API validation.
5. **Deployment:** Use Azure services for secure and efficient deployment.
6.

Application Structure

Frontend Directory Structure

```
frontend/

├── public/

│   ├── index.html          # Root HTML file

│   ├── favicon.ico         # App favicon

├── src/

│   ├── assets/             # Static assets (images, fonts, icons)

│   │   ├── images/

│   │   ├── fonts/

│   ├── components/         # Reusable components

│   │   ├── Sidebar.js      # Sidebar for navigation

│   │   ├── Dashboard.js    # Dashboard layout

│   ├── pages/              # Pages for different roles

│   │   ├── student/        # Student-specific features
│   │   │   ├── Dashboard.js
│   │   │   ├── Attendance.js
│   │   │   ├── Assignments.js
│   │   │   ├── CommunityForum.js
│   │   │   ├── Materials.js
│   │   │   ├── Performance.js
│   │   │   ├── Recordings.js
│   │   │   ├── LeaveManagement.js
│   │   │   ├── Certifications.js
│   │   ├── tpo/            # Training and Placement Officer-specific features
│   │   │   ├── Dashboard.js
│   │   │   ├── Attendance.js
│   │   │   ├── Performance.js
│   │   │   ├── Certifications.js
│   │   ├── trainer/        # Trainer-specific features
│     │     │     ├──── Dashboard.js
│     │     │     ├──── Attendance.js
│     │     │     ├──── Assignments.js
│     │     │     ├──── CommunityForum.js
│     │     │     ├──── Materials.js
│     │     │     ├──── Performance.js
│     │     │     ├──── Recordings.js
│     │     │     ├──── LeaveManagement.js
│     │     │     ├──── Certifications.js
│     │     ├──── admin/          # Admin-specific features
│     │     │     ├──── Dashboard.js
│     │     │     ├──── Attendance.js
│     │     │     ├──── Assignments.js
│     │     │     ├──── CommunityForum.js
│     │     │     ├──── Materials.js
│     │     │     ├──── Performance.js
│     │     │     ├──── Recordings.js
│     │     │     ├──── LeaveManagement.js
│     │     │     ├──── Certifications.js
│     ├──── hooks/              # Custom React hooks

│     │     ├──── useAuth.js         # Authentication hook

│     │     ├──── useFetch.js        # Data fetching hook

│     ├──── utils/              # Utility functions

│     │     ├──── api.js             # Axios API configurations

│     │     ├──── formatDate.js      # Helper for date formatting

│     ├──── styles/             # Styling and theme management
│     │     ├──── global.css         # Global CSS styles
│     │     ├──── variables.css      # CSS variables and themes
│     │     ├──── components/        # Component-specific styles
│     │        ├──── Navbar.css
│     │        ├──── Sidebar.css
│     ├──── App.js              # Main React component

│     ├──── index.js            # React DOM entry point

├──── .env                      # Environment variables (API URLs)

├──── .gitignore                # Exclude build files, node_modules

├──── package.json              # Frontend dependencies

├──── webpack.config.js         # Webpack configuration (for bundling)

├──── README.md                 # Project documentation
```

backend/

```
├──── config/
│    ├──── db.js                # MongoDB Atlas connection config
│    ├──── default.json         # JWT, API keys, and other sensitive configurations
├──── controllers/              # Logic for handling API requests
│    ├──── authController.js    # User authentication logic
│    ├──── studentController.js # Student-specific API routes
│    ├──── tpoController.js     # TPO-specific API routes
│    ├──── trainerController.js # Trainer-specific API routes
│    ├──── adminController.js   # Admin-specific API routes
├──── middlewares/              # Middlewares for authentication and authorization
│    ├──── authMiddleware.js    # JWT authentication middleware
│    ├──── roleMiddleware.js    # Role-based access control middleware
├──── models/                   # MongoDB models (Schemas)
│    ├──── User.js              # User schema with roles (Student, Trainer, Admin, TPO)
│    ├──── Attendance.js        # Attendance schema
│    ├──── Assignment.js        # Assignments schema
│    ├──── Certification.js     # Certifications schema
│    ├──── Resource.js          # Resource schema (materials)
│    ├──── Leave.js             # Leave management schema
│    ├──── CommunityPost.js     # Community forum schema
├──── routes/                   # API routes
│    ├──── authRoutes.js        # Routes for authentication (login, register)
│    ├──── studentRoutes.js     # Routes for student APIs
│    ├──── tpoRoutes.js         # Routes for TPO APIs
│    ├──── trainerRoutes.js     # Routes for trainer APIs
│    ├──── adminRoutes.js        # Routes for admin APIs
├──── utils/                     # Utility functions
│    ├──── generateToken.js      # Generate JWT tokens
│    ├──── sendEmail.js          # Send email utility (for notifications)
│    ├──── uploadFile.js         # File upload handler (S3, Cloudinary, etc.)
├──── .env                       # Environment variables (MongoDB URI, JWT secret)
├──── .gitignore                 # Exclude node_modules, logs, sensitive files
├──── package.json               # Backend dependencies
├──── server.js                  # Main entry point for the backend
├──── cronJobs/                  # Scheduled jobs (for cleanup, periodic tasks)
│    ├──── cleanupOldData.js     # Cleanup old data (optional)
├──── README.md                  # Project documentation
```