***Full-Stack Video Training Module***

**Introduction**

This report provides a detailed overview of the frontend and backend implementation for a video training module application. The application is designed to offer video playback with sequential viewing, progress tracking, and limited playback controls.

**Frontend Implementation**

**Technology Stack:**

* **React:** A JavaScript library for building user interfaces.
* **React Router:** For routing and navigation within the application.
* **Axios:** For making HTTP requests to the backend server.
* **CSS:** For styling the components and layout.

**Components:**

1. **VideoLibrary Component:**
   * **Purpose:** Displays a list of videos available for viewing.
   * **Functionality:**
     + Fetches the list of videos from the backend.
     + Displays each video as a link that navigates to the VideoPlayer component when clicked.
2. **VideoPlayer Component:**
   * **Purpose:** Plays a selected video.
   * **Functionality:**
     + Fetches video details based on the video ID from the URL.
     + Autoplays the video and allows only pause and rewind controls.
     + Saves the current playback position and resumes from where it left off if paused.
     + Triggers a completion event when the video ends, marking it as watched.
3. **ProgressDashboard Component:**
   * **Purpose:** Displays the user's progress through the training module.
   * **Functionality:**
     + Shows the percentage of videos completed based on the total number of videos and the number of videos watched.

**Styling:**

* **CSS Styling:** Applied to centre-align components and add a grey background colour for better visual appeal.

**Backend Implementation**

**Technology Stack:**

* **Node.js:** A JavaScript runtime built on Chrome's V8 engine.
* **Express:** A web application framework for Node.js.
* **MongoDB:** A NoSQL database to store video data.
* **Mongoose:** An ODM (Object Data Modeling) library for MongoDB and Node.js.

**Components:**

1. **Server Setup:**
   * **Purpose:** Sets up the Express server and connects to the MongoDB database.
   * **Functionality:**
     + Initializes the server on a specified port.
     + Uses middleware to handle JSON requests and enable CORS.
     + Connects to MongoDB using Mongoose.
2. **Video Model:**
   * **Purpose:** Defines the schema for video data.
   * **Functionality:**
     + Contains fields such as video ID, title, and URL.
3. **Video Routes:**
   * **Purpose:** Handles HTTP requests related to videos.
   * **Functionality:**
     + Provides endpoints to fetch the list of videos and specific video details based on ID.

**Troubleshooting and Testing**

* **Frontend Testing:**
  + Ensure that components are correctly fetching and displaying data.
  + Verify that videos autoplay, and controls are limited as specified.
  + Check that progress is accurately reflected in the ProgressDashboard.
* **Backend Testing:**
  + Use tools like Postman to test API endpoints.
  + Verify that data is correctly retrieved from and stored in MongoDB.

**Conclusion**

This report outlines the implementation details and functionality of the video training module's frontend and backend components. The setup ensures a smooth user experience with sequential video playback, progress tracking, and controlled playback features.