Detailed Description of the Solution

1. Framework: Express.js

- Express.js is a minimalist web framework for Node.js, known for its simplicity, flexibility, and strong community support. It was chosen to handle both static file serving and dynamic routing, making it an excellent fit for a project that requires serving HTML pages and streaming video content.

2. Libraries Used:

- **express:** The core framework used to create the server and handle routing.

- **fs**: Node.js built-in module for interacting with the file system, used to read video files.

- path: Another Node.js built-in module for handling and transforming file paths, ensuring compatibility across different operating systems.

- **video.js :** A popular HTML5 video player library, included in the HTML pages for advanced video playback features.

-mysql: MySQL client for Node.js, used to connect and interact with a MySQL database, which stores information such as user progress and video metadata.

3. HTML Pages Serving:

- The server responds to requests for specific routes (video,video2, video3) by sending back the corresponding HTML file. This provides the necessary user interface for different video modules in the course.

4. Video Streaming:

- The 'videoStream' function is used to stream video files to the client. The function checks for the `Range` header, which allows partial content delivery, a key feature for streaming. The video file is read in chunks (1MB by default), and these chunks are sent to the client to enable smooth streaming without needing to load the entire video at once.

CHUNK_SIZE = 10 ** 6 (1MB)

5. Page Structure:

- The page is designed to display video content alongside progress indicators and navigation buttons. The CSS styles ensure that the video is positioned at the right corner and displayed at an appropriate size.

- Progress Indicators: Simple text-based indicators show the user's progress through the course modules.

- Navigation: Navigation between modules is handled through buttons that link to other pages or trigger JavaScript functions.

- CSS: The CSS file `Module.css` (linked in the HTML) likely contains additional styles that help structure the page, while the inline styles position the video container.

6. JavaScript for Interactivity:

- External Script: `script.js` (linked in the HTML) is likely used to handle user interactions, such as navigating between modules or tracking user progress.

- Video.js Integration: The inclusion of the `video.js` library allows for enhanced video controls and compatibility across different browsers, ensuring a smooth user experience.