**1. Project Overview**

This project allows user to input a YouTube playlist link for a course, specify the duration within which they want to complete the course, and generate a study routine to complete the course within the stipulated time.

**2. Software Requirements Specification (SRS)**

**2.1 Functional Requirements**

* **User Registration & Authentication:** Users can register and log in to the platform.
* **Input Playlist:** Users can input a YouTube playlist link.
* **Time Specification:** Users can specify the time frame(in days) to complete the course.
* **Routine Generation:** The system generates a daily routine based on the number of videos and their lengths in the playlist.
* **Progress Tracking:** Users can track their progress and mark the videos as complete.

**2.2 Non-Functional Requirements**

* **Scalability:** The system should handle multiple users and playlists simultaneously.
* **Performance:** The routine generation should be efficient, even for lengthy playlists.
* **Security:** User data should be stored securely.

**3. System Architecture**

**­­­3.1 Frontend (React.js)**

* **Components:**
  + Login.jsx
  + Register.jsx
  + Dashboard.jsx
  + InputPlaylist.jsx
  + Routine.jsx
  + ProgressTracker.jsx
* **Pages:**
  + /login: User login page.
  + /register: User registration page.
  + /dashboard: User dashboard showing routine and progress.
  + /input-playlist: Page to input playlist and time frame.
* **State Management:**
  + Use Redux or Context API for managing user state, playlist data, and progress.

**3.2 Backend (Node.js/Express)**

* **Routes:**
  + /api/auth/register: Registers a new user.
  + /api/auth/login: Authenticates a user.
  + /api/playlist/input: Accepts the YouTube playlist link and time frame.
  + /api/routine/generate: Generates the study routine.
  + /api/progress/update: Updates the user’s progress.
* **Controllers:**
  + authController.js
  + playlistController.js
  + routineController.js
  + progressController.js

**3.3 Database (SQL)**

* **Tables:**
  + **users:**
    - id (Primary Key)
    - username
    - email
    - password
  + **playlists:**
    - id (Primary Key)
    - user\_id (Foreign Key referencing users)
    - youtube\_link
    - total\_videos
    - total\_duration
  + **routines:**
    - id (Primary Key)
    - playlist\_id (Foreign Key referencing playlists)
    - day
    - videos
    - duration
  + **progress:**
    - id (Primary Key)
    - user\_id (Foreign Key referencing users)
    - playlist\_id (Foreign Key referencing playlists)
    - videos\_completed
    - date

**4. API Endpoints**

* **Auth Routes**
  + POST /api/auth/register: Register a new user.
  + POST /api/auth/login: Login and receive a JWT token.
* **Playlist Routes**
  + POST /api/playlist/input: Accept the YouTube playlist link and the desired time frame.
* **Routine Routes**
  + POST /api/routine/generate: Generate the routine based on the playlist and time frame.
* **Progress Routes**
  + PATCH /api/progress/update: Update the progress of the user.

**5. Database Schema**

**5.1 Users Table**

This table stores user information.

CREATE TABLE users (

id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE,

password\_hash VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

**5.2 Playlists Table**

This table stores information about the YouTube playlists entered by users.

CREATE TABLE playlists (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

playlist\_url VARCHAR(255) NOT NULL,

playlist\_title VARCHAR(255),

video\_count INT NOT NULL,

total\_duration INT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

**5.3 Videos Table**

This table stores the individual videos from the YouTube playlist.

CREATE TABLE videos (

id INT PRIMARY KEY AUTO\_INCREMENT,

playlist\_id INT NOT NULL,

video\_url VARCHAR(255) NOT NULL,

video\_title VARCHAR(255) NOT NULL,

duration INT NOT NULL, -- in minutes

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (playlist\_id) REFERENCES playlists(id) ON DELETE CASCADE

);

**5.4 Routines Table**

This table stores the routines generated for users based on their playlist.

CREATE TABLE routines (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

playlist\_id INT NOT NULL,

days\_to\_complete INT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (playlist\_id) REFERENCES playlists(id) ON DELETE CASCADE

);

**5.5 Routine\_Days Table**

This table stores the tasks for each day of the routine.

CREATE TABLE routine\_days (

id INT PRIMARY KEY AUTO\_INCREMENT,

routine\_id INT NOT NULL,

day\_number INT NOT NULL, -- Represents Day 1, Day 2, etc.

video\_ids VARCHAR(255) NOT NULL,

total\_duration INT NOT NULL, -- in minutes

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (routine\_id) REFERENCES routines(id) ON DELETE CASCADE

);

**Explanation of the Schema**

1. **Users Table**: Stores user details like username, email, and password hash.
2. **Playlists Table**: Stores the playlists entered by users, including the total video count and total duration of the playlist.
3. **Videos Table**: Stores details of each video in the playlist, such as the video title, URL, and duration.
4. **Routines Table**: Stores the generated routine for the user, linking it to a specific playlist and the number of days to complete it.
5. **Routine\_Days Table**: Breaks down the routine into individual days, with each day having a list of video IDs that the user is supposed to watch on that day.

**Schema Usage Flow**

* A user registers or logs in.
* The user inputs a YouTube playlist link.
* The backend fetches playlist details and stores them in the playlists and videos tables.
* A routine is generated based on the user’s input (days to complete) and is stored in the routines and routine\_days tables.
* The user can view their routine, with the system fetching the relevant information from the routine\_days and videos tables.