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.