**Proposal for LetsScale: Empowering Learners Worldwide**

**Baraa Othman**

**Objective:**

LetsScale is an innovative online learning platform dedicated to provide accessible and comprehensive learning resources to empower individuals worldwide. Our mission is to break down barriers to education and foster lifelong learning by offering high-quality courses, tutorials, and tools across various domains. Additionally, LetsScale encourages users to contribute their own experiences and expertise by sharing videos, enriching the learning community and allowing others to benefit from their insights.

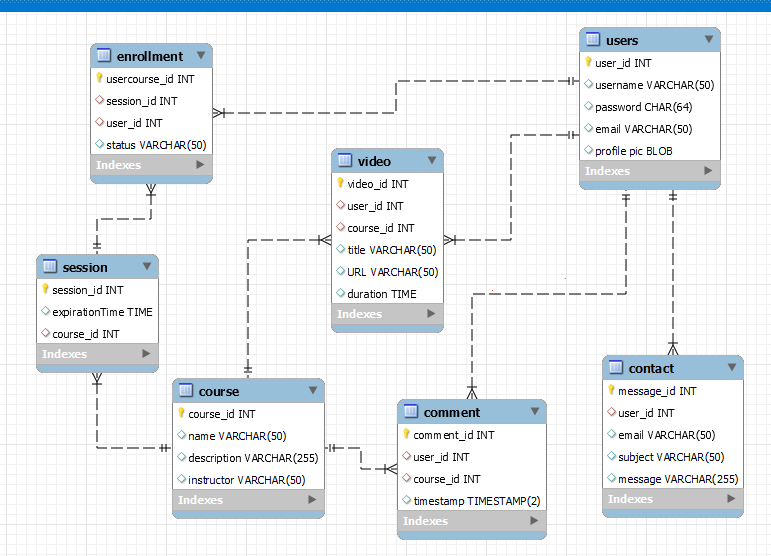
**Database Design:**

The database of letsScale will consist of 7 tables :

Users - Courses - Videos - Comments - Contacts - Enrollments - Sessions.

**Relationships:**

1 - Users - Course (Many-to-Many): This relationship indicates that many users can enroll in many courses, and conversely, many courses can have many users enrolled. Typically, this is implemented using a junction table to represent the many-to-many relationship.

1. Users - Comment (One-to-Many): This relationship suggests that a user can post multiple comments, and each comment is related to a single user.
2. Course - Comment (One-to-Many): This relationship signifies that courses can have multiple comments associated with them, while each comment belongs to a single course.
3. Users - Video (One-to-Many): This relationship indicates that a user can upload multiple videos, and each video is associated with a single user.
4. Users- Session (One-to-Many): This relationship suggests that a user can participate in multiple sessions across different course.
5. Users - Contact (One-to-Many): This relationship indicates that a user can have multiple contact messages, and each contact message is related to a single user. 

**Controllers:**

User Controller:

~~- registerUser(userData)~~

~~- loginUser(credentials)~~

~~- logOutUser()~~

~~- getUserProfile(user\_id)~~

~~- updateUserProfile(user\_id,userData)~~

~~- deleteUser(user\_id)~~

Course Controller:

~~- createCourse(courseData)~~

~~- getCourseById(course\_id)~~

~~- updateCourse(course\_id, updatedData)~~

~~- deleteCourse(course\_id)~~

~~- withdrawCourse(course\_id,user\_id)~~

~~- getEnrolledCourses(user\_id)~~

Video Controller:

~~- uploadVideo(user\_id,course\_id,videoData)~~

~~- deleteVideo(video\_id)~~

~~- getVideo(video\_id)~~

**- playVideo(video\_id,user\_id)**

Comment Controller:

~~- createComment(user\_id,course\_id,commentData)~~

~~- deleteComment(comment\_id)~~

~~- getComments(course\_id)~~

Session Controller:

~~- createSession(course\_id,sessionData)~~

~~- updateSession(session\_id,updatedData)~~

- ~~deleteSession(session\_id)~~

~~- getSessions(course\_id)~~

Contact Controller:

~~- sendMessage(user\_id,data)~~

~~- getMessages(~~)

~~Search Controller:~~

~~- searchCourses(query)~~

~~- searchVideos(query)~~

~~- searchContent(query)~~

Enrollment Controller:

~~- enrollCourse(user\_id,course\_id)~~

**- getUserEnrollements(user\_id)**

**- getCourseEnrollments(course\_id)**

**Services:**

User Services:

~~- registerUser(userData)~~

~~- loginUser(credentials)~~

~~- logoutUser()~~

- ~~getUserProfile(user\_id)~~

- ~~updateUserProfile(user\_id,userData)~~

~~- deleteUser(user\_id)~~

Course Services:

~~- createCourse(courseData)~~

~~- getCourseById(course\_id)~~

~~- updateCourse(course\_id,updatedData)~~

- ~~deleteCourse(course\_id)~~

~~- withdrawCourse(course\_id,user\_id)~~

~~- getEnrolledCourses(user\_id)~~

Video services:

~~- uploadVideo(user\_id,course\_id,videoData)~~

~~- deleteVideo(video\_id)~~

~~- getVideo(video\_id)~~

- playVideo(video\_id,user\_id)

Comment services:

~~- createComment(user\_id,course\_id,commentData)~~

~~- deleteComment(comment\_id)~~

~~- getComments(course\_id)~~

Session services:

~~- createSession(course\_id,sessionData)~~

~~- updateSession(session\_id,updatedData)~~

~~- deleteSession(session\_id)~~

- ~~getSessions(course\_id)~~

Contact services:

~~- sendMessage(user\_id,messageData)~~

~~- getMessages()~~

~~Search services:~~

~~- searchCourses(query)~~

~~- searchVideos(query)~~

~~- searchContent(query)~~

Enrollment services:

- enrollCourse(user\_id,course\_id)

- getUserEnrollements(user\_id)

- getCourseEnrollments(course\_id)

**Routes:**User Routes:s

~~POST /users/register~~

~~POST /users/login~~

~~POST /users/logout~~

~~GET /users/:userId/profile~~

~~PUT /users/:userId/profile~~

~~DELETE /users/:userId~~

Course Routes:

~~POST /course~~

~~GET /course/:courseId~~

~~PUT /course/:courseId~~

~~DELETE /course/:courseId~~

~~DELETE /course/:courseId/withdraw~~

~~GET /users/:userId/course~~

Video Routes:

~~POST /video~~

~~DELETE /video/:videoId~~

~~GET /videos/:videoId~~

GET /video/:videoId/play

Comment Routes:

~~POST /comment~~

~~DELETE /comment/:commentId~~

~~GET /course/:courseId/comment~~

Session Routes:

~~POST /session~~

~~PUT /session/:sessionId~~

~~DELETE /session/:sessionId~~

~~GET /session/:courseId~~

Contact Routes:

~~POST /contact~~

~~GET /contact/message~~

~~Search Routes:~~

~~GET /search/course~~

~~GET /search/video~~

~~GET /search/content~~

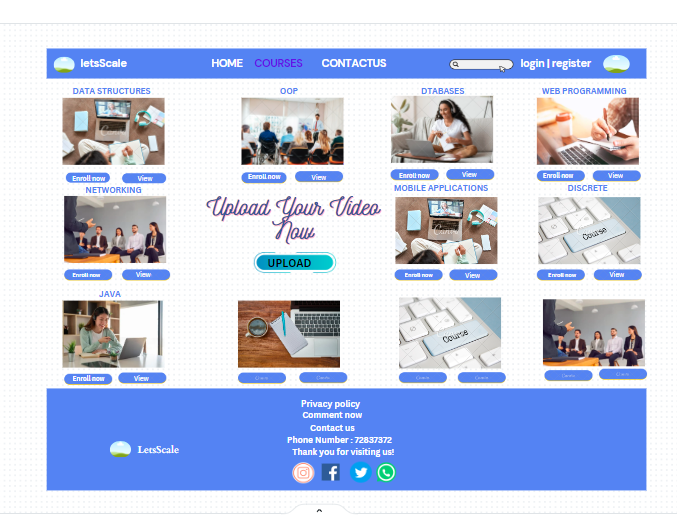
Enrollment Routes:

~~POST /course/:courseId/enroll~~

GET /users/:userId/enrollment

GET /course/:courseId/enrollment

**I will use sendGrid v3 api to receive notifications by email when someone add a contact message in my contact page.**

**Courses page: Note: Please understand that this layout serves as a basic representation of my concept. I plan to further develop the front-end design.** 

**Conclusion:**

LetsScale is committed to revolutionizing online learning by providing a dynamic and inclusive platform powered by advanced database technologies. With a focus on scalability, security, and data-driven insights, we aim to empower learners worldwide to achieve their educational goals and unlock their full potential.