1. Register
2. Login
3. Profile
4. Edit profile
5. Admin\_manage\_blogs (create blogs, delete blogs)
6. Comment
7. Contact
8. Admin\_manage\_message (respone messages)
9. Admin\_manage\_teachers ( add teacher, remove teacher)
10. Admin\_manage\_enrollments (remove enrollments ”student”)

Admin\_manage\_lessons (add lessons, update lessons, remove lessons)

1. Admin\_manage\_courses (add new course, update course name description and date , remove course ,update assigned teacher )
2. Admin\_assign\_teacher (assign teacher to course)
3. Teacher\_manage\_lessons( create, update, remove lessons)
4. Teacher\_manage\_assignment( create(2 types images or mp3), remove assignments [2 types fill in blank and mcqs], view submissions)
5. Student\_join\_courses
6. Student\_view\_lessons
7. Student\_submit\_assignments
8. Student\_view\_grades
9. Student\_view\_correct\_answer

Implementation

1. Register

* Front-end : create a register form with fields : username, email, password,re-enter password, select your role (teacher, student)
* Back-end : create an API end point route **(/register)** to handle user registration

Validate user input and store user data in the database in **users table.**

**Database :** users table with columns : 'id','username', 'email', 'password', 'role'

**Security :** Hash password before storing them in the database using a library bcrypt

1. Login

* Front-end Create a login form with fields for email and password
* Back-end : Create an API end point (/login) to handle authentication

Verify the email and password against the database and return it to the front-end

1. Profile and Edit profile

* Front-end: create a user profile page showing user details and users can edit their details
* Back-end : create API end points (/profile and /edit\_profile) to fetch user data and update user data from the database
* Database : Store profile fields (id, user\_id, name, account ,phonenumber ,Dateofbirth, live, nationality ,speciality ,description ,created\_at ,updated\_at, profile\_picture)

1. Admin manage blogs

* Frontend : create a dashboard for admin to create and delete blogs
* Blogs include (title,content,images,videos,others)
* Backend : create API endpoints (/admin/manage\_blogs)( /create\_blog) (/delete\_blog/<int:blog\_id>)( '/blog/<int:blog\_id>') store in a blogs table in database

1. Comment

* Frontend add a comment section below blogs
* Backend create API endpoint ('/comment/<int:blog\_id>') to add comments
* ( '/blog/<int:blog\_id>') is using for fetching and displaying comments

1. Contact

* Frontend Create a contact form for users to send messages
* Backend : Create an Api endpoint (/contact) to store messages in contact\_messages table
* Database : id, name, email, phone, subject, message, submitted at

1. Admin manage messages

* Front-end : create a respond form for admin to respond messages via mail
* Back-end: create an Api endpoint (/admin/respond/<int:message\_id>) to store messages in contact\_messages table
* Database : id, name, email, phone, subject, message, submitted at

1. Admin manage teachers

* Front end create a form for adding and remove teachers
* Backend : create api endpoints (/admin/manage\_teachers)   
  Store teacher data in the users table a role column as ‘teacher’

1. Admin manage enrollments

* Front end : create a list of enrolled students with options to remove them
* Back end create an API endpoints (/admin/manage\_enrollments) to remove them from the

Enrollments table

1. Admin manage lessons

* Front end prodive form for adding updating and removing lessons
* Back end create API endpoint (/admin/manage\_lessons)
* Database create a lessons table with (id, title, content, course\_id, video\_url, file\_path, video\_path)

1. Admin manage courses

* Front end add forms for adding and updating courses
* Backend create API endpoint (/admin/manage\_courses)
* Database create a teacher\_course table include (id, teacher\_id, course\_id)

1. Admin assign teacher

* Front end add a dropdown to assign teacher to course
* Back end create API endpoint (/admin/assign\_teacher) to update the teacher\_course table

1. Teacher mange lessons

* Front end : provide forms for teachers to manage lessons
* Back end : create an API endpoint (/teacher/manage\_lessons/<int:course\_id>) to CRUD operations lessons and save it in lessons table

1. Teacher manage assignments

* Frontend : create a page for teachers to create, remove and view assignments and view submissions

Add options for assignment types (images, mp3, mp4, others and fill-in-blank and MCQs)

* Backend : create endpoints (/teacher/create\_assignment/<int:course\_id>) , (/teacher/remove\_assignment/<int:assignment\_id>), (/teacher/view\_student\_submissions/<int:assignment\_id>/<int:course\_id>)

1. Student join courses

* Frontend : add a “join course” button for students
* Backend : Create an API endpoint (/student/join\_course) and (/enroll/<int:course\_id>)to add entries to the

Enrollments table

1. Student view lessons

* Frontend : create a page to display lessons in enrolled courses
* Backend: fetch lessons using endpoint (/student/lessons/<int:course\_id>)

1. Student submit assignments

* Frontend add a form for students to upload assignments
* Backend create an API endpoints (/student/assignment/<int:assignment\_id>) to store submissions in the student\_answers table for automatic calculate grade
* Database : add a student\_answers table with (id, student\_id, assignment\_id, question\_id, , answer)

1. Student view grades and view correct answers

* Frontend display grade on a student dashboard and “View correct answers” button for grade
* Backend : create endpoints (/student/grades)( /student/assignment/<int:assignment\_id>/answers)

**System Architecture (English)**

**1. Overview of the Architecture**

• The system is based on a **Local-host sever**? Architecture:

• Client: Runs on the user’s browser, designed with Bootstrap, and communicates with the server through APIs.

• Server: Processes business logic and connects to the database.

• Database: Stores course data, lesson content, and related information.

**2. Key Components**

1. Frontend: • Built with Bootstrap to create a responsive user interface. • Features: • Displays courses and lectures. • Allows users to browse content, view lectures, and track their learning progress. • Communicates with the backend through APIs.
2. Backend: • Developed using **Flask** to handle business logic. • Features: (20 cai o tren)
3. Database: • **MySQL workbench** is used to store structured data. • Key tables: •

Assignment\_images (idassignment\_idimage\_path) table

Assignment\_mp3 (id,assignment\_id, mp3\_path)table

Assignments (id, title, description, created\_at, course\_id, created\_by) table

Blog (id, title, content, image\_url, author, created\_at, image\_urls, video\_urls) table

Comment (id, blog\_id, nametitle, content, created\_at, parent\_comment\_id, user\_id) table

Contact\_messages (id, name, email, phone, subject, message, submitted\_at) table

Courses (id, name, description, created\_by, start\_date, end\_date) table

Enrollments (id, user\_id, course\_id) table

Grades (id, student\_id, assignment\_id , grade, feedback, graded\_by, created\_at, updated\_at) table

Lessons (id, title, content, course\_id, video\_url, file\_path, video\_path) table

Profile (id, user\_id, name, account, phonenumber, dateofbirth, live, nationality, speciality, description, created\_at, updated\_at, profile\_picture ) table

Questions (id, assignment\_id, question\_type, question\_text, correct\_answer, option\_a, option\_b, option\_c, option\_d) table

Student\_answers( id, student\_id, assignment\_id, question\_id, answer) table

Teacher\_course (id, teacher\_id, course\_id ) table

Users (id, username, email, password, role) table

Total 15 tables

1. File Storage: • Lecture files/videos are stored on folder.

**3. Communication Between Components**

• API: • Flask API is used for communication between the frontend and backend.

• Key endpoints:

'/register' for register new users

'/login' for login users

‘/logout’ exit the account

'/profile' show the profile details of a user

'/edit\_profile' changing profile’s user

'/admin/dashboard' show board of functions for admins

('/admin/manage\_blogs') fetch the numbers of blogs

'/create\_blog' admin create blogs

'/delete\_blog/<int:blog\_id>' admin remove blogs

'/comment/<int:blog\_id>' users comment in blogs

'/blog/<int:blog\_id>' show details of blog and comments in that blog

'/contact' user can contact to admin

'/admin/manage\_messages' fetch message

'/admin/respond/<int:message\_id>' use for admin respond message

'/admin/manage\_teachers' admin can ( add teacher, remove teacher)

'/admin/manage\_enrollments' admin can (remove enrollments ”student”)

'/admin/manage\_lessons' admin can (add lessons, update lessons, remove lessons)

'/admin/manage\_courses' (add new course, update course name description and date , remove course ,update assigned teacher )

'/admin/assign\_teacher' use for assign teacher to course

'/teacher/dashboard' show functions in dashboard for teacher

'/teacher/manage\_lessons/<int:course\_id>' teacher can ( create, update, remove lessons)

'/teacher/create\_assignment/<int:course\_id>' ( create(2 types images or mp3), remove assignments [2 types fill in blank and mcqs], view submissions)

'/teacher/remove\_assignment/<int:assignment\_id>' remove assignment

'/teacher/view\_student\_submissions/<int:assignment\_id>/<int:course\_id>' view student score and submission time

'/student/dashboard' show functions for students

'/student/join\_course'

'/enroll/<int:course\_id>' use for student join available course

'/student/lessons/<int:course\_id>' use for student view lessons (include video pdfs link ytb images and others)

'/student/assignment/<int:assignment\_id>' use for student submit assignment and calculate grade automatically

'/student/grades' help student see their grade when done assignment after submission

'/student/assignment/<int:assignment\_id>/answers' help student check their answers compare to the correct answers

• Data Flow: • Users send requests from the frontend. • The backend processes the requests, retrieves data from the database, or saves new data. • Data is returned to the frontend for display.