now you're moving to the **next level: LMS API + Android app (Kotlin frontend)**.

I'll give you a **step-by-step beginner-friendly guide** to convert your PHP LMS into an API + connect it with Android.

□ OVERALL ARCHITECTURE

```
MySQL (Database)
      ↓
PHP REST API (Backend)
      ↓
Android App (Kotlin UI)
```

□ STEP 1: CREATE API FOLDER IN PHP

Inside your LMS project:

```
/lms/api/
```

□ STEP 2: CREATE DATABASE CONNECTION FOR API

□ **api/db.php**

```
<?php
header("Content-Type: application/json");

$conn = new mysqli("localhost", "root", "", "lms");

if($conn->connect_error){
    die(json_encode(["status" => "error", "message" => "DB connection
failed"]));
}
?>
```

□ STEP 3: API – LOGIN STUDENT

□ **api/student_login.php**

```
<?php
include('db.php');

$email = $_POST['email'];
$password = md5($_POST['password']);

$q = $conn->query("SELECT * FROM students WHERE email='$email' AND
password='$password'");

if($q->num_rows > 0){
    $data = $q->fetch_assoc();

    echo json_encode([
        "status" => "success",
        "student_id" => $data['id'],
        "name" => $data['name']
    ]);
} else {
    echo json_encode([
        "status" => "error",
        "message" => "Invalid login"
    ]);
}
?>
```

□ STEP 4: API – GET COURSES (ENROLLED ONLY)

```
<?php
include('db.php');

$student_id = $_GET['student_id'];

$result = $conn->query("
SELECT courses.id, courses.title
FROM courses
INNER JOIN enrollments
ON courses.id = enrollments.course_id
WHERE enrollments.student_id = '$student_id'
");

$courses = [];

while($row = $result->fetch_assoc()){
    $courses[] = $row;
```

```
}

echo json_encode($courses);
?>
```

□ STEP 5: API – GET LECTURES

```
<?php
include('db.php');

$course_id = $_GET['course_id'];

$result = $conn->query("
SELECT * FROM lectures WHERE course_id='$course_id'
");

$lectures = [];

while($row = $result->fetch_assoc()){
    $lectures[] = $row;
}

echo json_encode($lectures);
?>
```

□ STEP 6: TEST API (IMPORTANT)

Open in browser:

```
http://localhost/lms/api/student_login.php
```

Use POST via Postman:

- email
- password

□ STEP 7: ANDROID APP (KOTLIN)

□ STEP 7.1 ADD INTERNET PERMISSION

AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET"/>
```

□ STEP 7.2 ADD RETROFIT

build.gradle

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

□ STEP 7.3 API INTERFACE (KOTLIN)

```
interface ApiService {

    @FormUrlEncoded
    @POST("student_login.php")
    fun login(
        @Field("email") email: String,
        @Field("password") password: String
    ): Call<LoginResponse>

    @GET("get_courses.php")
    fun getCourses(
        @Query("student_id") studentId: Int
    ): Call<List<Course>>

    @GET("get_lectures.php")
    fun getLectures(
        @Query("course_id") courseId: Int
    ): Call<List<Lecture>>
}
```

□ STEP 7.4 DATA MODELS

Login Response

```
data class LoginResponse(
    val status: String,
    val student_id: Int,
    val name: String
)
```

Course

```
data class Course(
    val id: Int,
    val title: String
)
```

Lecture

```
data class Lecture(
    val id: Int,
    val title: String,
    val youtube_link: String,
    val pdf_file: String
)
```

□ STEP 7.5 RETROFIT CLIENT

```
object RetrofitClient {

    private const val BASE_URL = "http://10.0.2.2/lms/api/"

    val instance: ApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(ApiService::class.java)
    }
}
```

□ STEP 8: LOGIN IN ANDROID

```
RetrofitClient.instance.login(email, password)
    .enqueue(object : Callback<LoginResponse> {

        override fun onResponse(call: Call<LoginResponse>, response:
Response<LoginResponse>) {

            if(response.body()?.status == "success"){
                val id = response.body()!!.student_id
                // go to dashboard
            }
        }

        override fun onFailure(call: Call<LoginResponse>, t: Throwable) {
        }
    })
```

□ STEP 9: SHOW COURSES IN ANDROID

Use RecyclerView:

- Show course title
- Click → open lectures

□ STEP 10: PLAY LECTURES IN ANDROID

Use WebView:

```
webView.loadUrl("https://www.youtube.com/embed/VIDEO_ID")
```

□ STEP 11: OPEN PDF IN ANDROID

Use:

```
https://docs.google.com/gview?embedded=true&url=YOUR_PDF_URL
```

□ FINAL FLOW

ANDROID APP FLOW:

```
Login
  ↓
Dashboard (Courses)
  ↓
Lectures List
  ↓
Watch YouTube Video
  ↓
View PDF
```

□ WHAT YOU JUST BUILT

✔ PHP REST API
✔ MySQL backend
✔ Android Kotlin app
✔ Login system
✔ Course system
✔ Lecture system
✔ Video + PDF support

□ NEXT LEVEL UPGRADES (VERY POWERFUL)

If you want next:

- □ JWT authentication (secure API)
- □ Progress tracking (watched videos)
- □ Certificate generation
- □ Push notifications (new lectures)
- □ Full Udemy-style Android app UI
- ☘□ Host API online (000webhost / VPS)

□ JUST TELL ME NEXT STEP

Say:

- “make full android project structure”
- “add JWT login”
- “add progress tracking API”
- “make UI like Udemy app”

I’ll build it step-by-step for you □

give me full kotlin project structure guide for this

Great □ now I’ll give you a **FULL Android (Kotlin) Project Structure for your LMS app** so you can build it like a real production app (Udemy-style architecture).

□ □ FULL ANDROID LMS PROJECT STRUCTURE (KOTLIN)

□ Architecture: MVVM (Recommended)

```
UI (Activities / Fragments)
        ↓
ViewModel
        ↓
Repository
        ↓
Retrofit API
        ↓
PHP Backend (Your LMS API)
```

□ 1. PROJECT STRUCTURE

□ App Package

```
com.lms.app
│
├── data/
│   ├── api/
│   │   ├── ApiService.kt
│   │   ├── RetrofitClient.kt
│   │
│   ├── model/
│   │   ├── LoginResponse.kt
│   │   ├── Course.kt
│   │   ├── Lecture.kt
│   │
│   ├── repository/
│       ├── AuthRepository.kt
│       ├── CourseRepository.kt
│       ├── LectureRepository.kt
│
├── ui/
│   ├── auth/
│   │   ├── LoginActivity.kt
│   │   ├── RegisterActivity.kt
│   │
│   ├── dashboard/
│   │   ├── DashboardActivity.kt
│   │
│   ├── course/
```

```
│   │   ├── CourseActivity.kt
│   │
│   ├── lecture/
│   │   ├── LectureActivity.kt
│   │   ├── VideoPlayerActivity.kt
│   │   ├── PdfViewerActivity.kt
│
├── viewmodel/
│   ├── AuthViewModel.kt
│   ├── CourseViewModel.kt
│   ├── LectureViewModel.kt
│
├── utils/
│   ├── SessionManager.kt
│
└── MainActivity.kt
```

---

2. API LAYER (Retrofit)

ApiService.kt

```
interface ApiService {

    @FormUrlEncoded
    @POST("student_login.php")
    fun login(
        @Field("email") email: String,
        @Field("password") password: String
    ): Call<LoginResponse>

    @GET("get_courses.php")
    fun getCourses(
        @Query("student_id") studentId: Int
    ): Call<List<Course>>

    @GET("get_lectures.php")
    fun getLectures(
        @Query("course_id") courseId: Int
    ): Call<List<Lecture>>
}
```

RetrofitClient.kt

```
object RetrofitClient {

    private const val BASE_URL = "http://10.0.2.2/lms/api/"

    val instance: ApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
```

```
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(ApiService::class.java)
    }
}
```

□ 3. MODELS (DATA CLASSES)

□ **LoginResponse.kt**

```
data class LoginResponse(
    val status: String,
    val student_id: Int,
    val name: String
)
```

□ **Course.kt**

```
data class Course(
    val id: Int,
    val title: String
)
```

□ **Lecture.kt**

```
data class Lecture(
    val id: Int,
    val course_id: Int,
    val title: String,
    val youtube_link: String,
    val pdf_file: String
)
```

□ 4. REPOSITORY LAYER

□ **AuthRepository.kt**

```
class AuthRepository {

    fun login(email: String, password: String, callback:
Callback<LoginResponse>) {
        RetrofitClient.instance.login(email, password).enqueue(callback)
```

```
    }
}
```

☐ CourseRepository.kt

```
class CourseRepository {

    fun getCourses(studentId: Int, callback: Callback<List<Course>>) {
        RetrofitClient.instance.getCourses(studentId).enqueue(callback)
    }
}
```

☐ LectureRepository.kt

```
class LectureRepository {

    fun getLectures(courseId: Int, callback: Callback<List<Lecture>>) {
        RetrofitClient.instance.getLectures(courseId).enqueue(callback)
    }
}
```

☐ 5. VIEWMODEL LAYER

☐ AuthViewModel.kt

```
class AuthViewModel : ViewModel() {

    private val repo = AuthRepository()

    fun login(email: String, password: String, callback:
Callback<LoginResponse>) {
        repo.login(email, password, callback)
    }
}
```

☐ CourseViewModel.kt

```
class CourseViewModel : ViewModel() {

    private val repo = CourseRepository()

    fun getCourses(studentId: Int, callback: Callback<List<Course>>) {
        repo.getCourses(studentId, callback)
    }
}
```

 LectureViewModel.kt

```
class LectureViewModel : ViewModel() {

    private val repo = LectureRepository()

    fun getLectures(courseId: Int, callback: Callback<List<Lecture>>) {
        repo.getLectures(courseId, callback)
    }
}
```

---

6. UI SCREENS

LoginActivity

Flow:

- Email + Password
- Call API
- Save student_id in session

DashboardActivity

✔ Show enrolled courses
✔ RecyclerView cards

CourseActivity

✔ Show lectures list
✔ Click lecture → open video/pdf

VideoPlayerActivity

```
webView.loadUrl("https://www.youtube.com/embed/VIDEO_ID")
```

□ PdfViewerActivity

```
webView.loadUrl("https://docs.google.com/gview?embedded=true&url=PDF_URL")
```

□ 7. SESSION MANAGER

□ SessionManager.kt

```
class SessionManager(context: Context) {

    private val pref = context.getSharedPreferences("lms",
Context.MODE_PRIVATE)

    fun saveStudentId(id: Int) {
        pref.edit().putInt("student_id", id).apply()
    }

    fun getStudentId(): Int {
        return pref.getInt("student_id", 0)
    }

    fun logout() {
        pref.edit().clear().apply()
    }
}
```

□ 8. ANDROID FLOW (REAL APP FLOW)

```
Login Screen
    ↓
Save student_id
    ↓
Dashboard (Courses)
    ↓
Course Click
    ↓
Lecture List
    ↓
Video Player / PDF Viewer
```

□ 9. BASE URL (IMPORTANT)

If using emulator:

```
http://10.0.2.2/lms/api/
```

If using mobile:

```
http://YOUR_PC_IP/lms/api/
```

□ 10. WHAT YOU HAVE NOW

✔ Full Kotlin architecture (MVVM)
✔ Retrofit API integration
✔ Login system
✔ Course system
✔ Lecture system
✔ YouTube player
✔ PDF viewer
✔ Session management

FULL WORKING ANDROID ACTIVITY CODE (KOTLIN + UI + LOGIC) for your LMS so you can directly run it.

I'll include:

- LoginActivity
- RegisterActivity
- DashboardActivity (courses)
- CourseActivity (lectures)
- VideoPlayerActivity
- PdfViewerActivity
- SessionManager

□ 1. LOGIN ACTIVITY

□ **LoginActivity.kt**

```
class LoginActivity : AppCompatActivity() {

    lateinit var email: EditText
    lateinit var password: EditText
    lateinit var loginBtn: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        email = findViewById(R.id.email)
        password = findViewById(R.id.password)
        loginBtn = findViewById(R.id.loginBtn)

        loginBtn.setOnClickListener {

            RetrofitClient.instance.login(
                email.text.toString(),
                password.text.toString()
            ).enqueue(object : Callback<LoginResponse> {

                override fun onResponse(call: Call<LoginResponse>, response:
Response<LoginResponse>) {

                    if (response.body()?.status == "success") {

                        val id = response.body()!!.student_id

                        SessionManager(this@LoginActivity).saveStudentId(id)

                        startActivity(Intent(this@LoginActivity,
DashboardActivity::class.java))
                        finish()
                    }
                }

                override fun onFailure(call: Call<LoginResponse>, t:
Throwable) {
                }
            })
        }
    }
}
```

□ 2. REGISTER ACTIVITY

□ **RegisterActivity.kt**

```
class RegisterActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_register)

        val name = findViewById<EditText>(R.id.name)
        val email = findViewById<EditText>(R.id.email)
        val password = findViewById<EditText>(R.id.password)
        val btn = findViewById<Button>(R.id.registerBtn)

        btn.setOnClickListener {

            val url = "http://10.0.2.2/lms/api/register.php"

            val request = object : StringRequest(Method.POST, url,
                Response.Listener {
                    Toast.makeText(this, "Registered",
Toast.LENGTH_SHORT).show()
                    finish()
                },
                Response.ErrorListener {}) {

                override fun getParams(): MutableMap<String, String> {
                    return hashMapOf(
                        "name" to name.text.toString(),
                        "email" to email.text.toString(),
                        "password" to password.text.toString()
                    )
                }
            }

            Volley.newRequestQueue(this).add(request)
        }
    }
}
```

□ 3. DASHBOARD ACTIVITY (COURSES)

□ **DashboardActivity.kt**

```
class DashboardActivity : AppCompatActivity() {

    lateinit var recycler: RecyclerView
    val courses = ArrayList<Course>()
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_dashboard)

        recycler = findViewById(R.id.courseRecycler)

        val studentId = SessionManager(this).getStudentId()

        RetrofitClient.instance.getCourses(studentId)
            .enqueue(object : Callback<List<Course>> {

                override fun onResponse(call: Call<List<Course>>, response:
Response<List<Course>>) {

                    recycler.layoutManager =
LinearLayoutManager(this@DashboardActivity)

                    recycler.adapter = CourseAdapter(response.body()!!) {
course ->

                        val i = Intent(this@DashboardActivity,
CourseActivity::class.java)
                        i.putExtra("course_id", course.id)
                        startActivity(i)
                    }
                }

                override fun onFailure(call: Call<List<Course>>, t:
Throwable) {}
            })
    }
}
```

□ 4. COURSE ACTIVITY (LECTURES)

□ **CourseActivity.kt**

```
class CourseActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_course)

        val courseId = intent.getIntExtra("course_id", 0)
        val recycler = findViewById<RecyclerView>(R.id.lectureRecycler)

        RetrofitClient.instance.getLectures(courseId)
            .enqueue(object : Callback<List<Lecture>> {

                override fun onResponse(call: Call<List<Lecture>>, response:
Response<List<Lecture>>) {

                    recycler.layoutManager =
```

```
LinearLayoutManager(this@CourseActivity)

                    recycler.adapter = LectureAdapter(response.body()!!) {
lecture ->

                        val i = Intent(this@CourseActivity,
VideoPlayerActivity::class.java)
                        i.putExtra("video", lecture.youtube_link)
                        startActivity(i)
                    }
                }

                override fun onFailure(call: Call<List<Lecture>>, t:
Throwable) {}
            })
    }
}
```

□ 5. VIDEO PLAYER ACTIVITY

□ **VideoPlayerActivity.kt**

```
class VideoPlayerActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_video)

        val url = intent.getStringExtra("video")

        val webView = findViewById<WebView>(R.id.webView)

        webView.settings.javaScriptEnabled = true
        webView.loadUrl(url!!)
    }
}
```

□ 6. PDF VIEWER ACTIVITY

□ **PdfViewerActivity.kt**

```
class PdfViewerActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_pdf)

        val file = intent.getStringExtra("file")

        val url = "https://docs.google.com/gview?embedded=true&url=$file"

        val webView = findViewById<WebView>(R.id.pdfView)

        webView.settings.javaScriptEnabled = true
        webView.loadUrl(url)
    }
}
```

□ 7. SESSION MANAGER

□ **SessionManager.kt**

```
class SessionManager(context: Context) {

    private val pref = context.getSharedPreferences("lms",
Context.MODE_PRIVATE)

    fun saveStudentId(id: Int) {
        pref.edit().putInt("student_id", id).apply()
    }

    fun getStudentId(): Int {
        return pref.getInt("student_id", 0)
    }

    fun logout() {
        pref.edit().clear().apply()
    }
}
```

□ 8. WHAT YOU NOW HAVE

✔ Login system (API)
✔ Register system
✔ Dashboard (courses)
✔ Course → lectures
✔ Video player (YouTube)
✔ PDF viewer
✔ Session system
✔ Full navigation flow

□ 9. APP FLOW

```
Login
 ↓
Dashboard (Courses)
 ↓
Course Click
 ↓
Lecture List
 ↓
Video Player / PDF Viewer
```