## 🔧 ****Technology Stack****

**Backend**: ASP.NET 8 Web API

**Frontend**: (Your choice – React, Angular, Blazor, or Razor Pages)

**Database**: MySQL (managed via phpMyAdmin)

**Authentication**: JWT Bearer Authentication

**ORM**: Entity Framework Core with Pomelo.EntityFrameworkCore.MySql

## 🧱 ****System Architecture Overview****

### 1. ****User Roles****

**Student**

Registers and logs in

Views and downloads their own progress reports

**Admin**

Uploads reports for students

**Super Admin**

Manages admins

Monitors report uploads/downloads

## 📁 ****Backend API Project Structure****

pgsql

CopyEdit

NjalaUniversity.API/

│

├── Controllers/

│ ├── AuthController.cs

│ ├── StudentsController.cs

│ ├── ReportsController.cs

│ ├── AdminController.cs

│ └── SuperAdminController.cs

│

├── Models/

│ ├── User.cs

│ ├── Report.cs

│ └── Department.cs

│

├── DTOs/

│ ├── RegisterDto.cs

│ ├── LoginDto.cs

│ ├── ReportUploadDto.cs

│

├── Services/

│ ├── AuthService.cs

│ ├── ReportService.cs

│ └── AdminService.cs

│

├── Data/

│ ├── ApplicationDbContext.cs

│

├── Program.cs

├── appsettings.json

## 🗃️ ****Database Tables****

Use MySQL and configure via EF Core:

### 1. ****Users****

| Id | FullName | Email | MatricNumber | PasswordHash | Role (Student/Admin/SuperAdmin) |

### 2. ****Departments****

| Id | Name |

### 3. ****Courses****

| Id | CourseCode | CourseName | DepartmentId |

### 4. ****Reports****

| Id | StudentId | DepartmentId | Semester | AcademicYear | FilePath | UploadDate |

### 5. ****AuditLogs****

| Id | Action | UserId | Timestamp |

## 🔑 ****Authentication & Security****

JWT Authentication with role-based authorization

Secure endpoints:

Only Students can view/download their reports

Only Admins can upload reports

Super Admin has access to monitoring dashboard

Optional: OTP/email verification before download (for extra security)

## 📋 ****Key Features Breakdown****

### ✅ ****Student Features****

Register with full name, email, and matric number

Login and get JWT token

View dashboard with uploaded reports

Filter reports by:

Department

Name

Matric Number

Download own report (verification required)

### ✅ ****Admin Features****

Upload reports for students by:

Department

Semester

Academic year

View list of uploaded reports

### ✅ ****Super Admin Features****

View upload/download statistics (charts, logs)

Manage (add/remove) Admins

Full access to audit logs

## 🌐 ****Frontend Suggestions****

Use **React.js** or **Blazor** for a rich UI

Implement download verification modal (e.g., require email OTP)

Frontend calls protected API endpoints with JWT

## 🔐 ****Security Notes****

Use HTTPS

Hash passwords with BCrypt

Apply [Authorize] attributes on sensitive endpoints

Filter access by UserId == HttpContext.User.Id in student endpoints

## 🗂️ ****Phase 1: Project Setup****

### ✅ Step 1.1 – Create ASP.NET 8 Web API Project

### ✅ Step 1.2 – Configure MySQL & phpMyAdmin Connection

### ✅ Step 1.3 – Install Required NuGet Packages

### ✅ Step 1.4 – Setup Entity Framework & ApplicationDbContext

## 🔐 Phase 2: Authentication System

### ✅ Step 2.1 – User Model (Student/Admin/SuperAdmin)

### ✅ Step 2.2 – JWT Authentication Setup

### ✅ Step 2.3 – Register & Login Endpoints

### ✅ Step 2.4 – Role-Based Access (Authorize Attributes)

## 📄 Phase 3: Database Design & Migrations

### ✅ Step 3.1 – Define Models: User, Department, Report, Course

### ✅ Step 3.2 – Setup Relationships

### ✅ Step 3.3 – Run EF Core Migrations to MySQL

## 📤 Phase 4: API Functionality

### ✅ Step 4.1 – Report Upload API (Admin only)

### ✅ Step 4.2 – Report Search/Filter API (Student only)

### ✅ Step 4.3 – Report Download API with Verification

### ✅ Step 4.4 – Super Admin Monitoring APIs

## 📈 Phase 5: Frontend Development (React or Blazor)

### ✅ Step 5.1 – User Registration & Login Form

### ✅ Step 5.2 – Student Dashboard: Search, Filter, Download

### ✅ Step 5.3 – Admin Dashboard: Upload Reports

### ✅ Step 5.4 – Super Admin Dashboard: Analytics

## 🔒 Phase 6: Security and Finalization

### ✅ Step 6.1 – HTTPS & Authentication Enforcement

### ✅ Step 6.2 – Prevent Unauthorized Access

### ✅ Step 6.3 – Testing All Features

### ✅ Step 6.4 – Deployment (Optional: localhost demo)