**Final Course Project [Human Security Course]**

# Full Secured Application Using Keycloak for Digital Identity Management

## 1. Introduction

In this project, students are required to design and implement a **fully secured web application** that integrates:

- A modern frontend framework
- A secured backend framework
- A database management system
- **Keycloak as a centralized Digital Identity and Access Management (IAM) server**

The main focus of this project is **NOT the system idea itself**, but rather the **correct implementation of digital identity, authentication, authorization, role-based access control, and token-based security using Keycloak**.

## 2. General System Architecture

The proposed system must be designed according to a multi-layer secured architecture consisting of four main components:

1. Frontend application.
2. Backend (API).
3. Database management system.
4. Centralized Identity and Access Management (IAM) server powered by Keycloak.

The frontend application is responsible only for user interaction and must never handle or store user passwords; instead, it redirects users to Keycloak for authentication using the Authorization Code Flow. After successful authentication, Keycloak issues an authorization code that is exchanged by the backend for a signed JSON Web Token (JWT) access token. The backend acts as the resource server and is fully responsible for validating token integrity, verifying token expiration, enforcing role-based access control, and processing business logic operations. Only validated requests are allowed to access the database layer, which contains the system data and must never be accessed directly by the frontend. All communication between the frontend and backend must be performed securely using the issued JWT access token in the Authorization HTTP header, ensuring full separation between authentication, authorization, business logic, and data storage.

## 3. Technology Stack (Choose One from Each Category)

### 3.1 Frontend (Choose One)

- Any modern frontend framework (React, Angular, Vue, etc.)

**3.2 Backend (Choose One)**

- Any backend framework(ASP.NET Core, Spring Boot (Java), Django (Python), Flask (Python), Node.js (Express, NestJS, etc.))

**3.3 Database (Choose One)**

- Any relational or non-relational database system (PostgreSQL, SQL Server, etc….)

**3.4 Identity and Access Management**

- **Keycloak is mandatory for all teams**

**4. Digital Identity and Security Requirements (Mandatory)**

Each team must strictly implement all of the following:

**4.1 Keycloak Integration**

- Create a full Keycloak Realm
- Create frontend and backend Clients
- Create Users and Roles
- Use **Authorization Code Flow** for frontend login
- Backend must verify: Token signature, Token expiration, User roles

**4.2 Excel-Based User Import (Mandatory)**

Users **must NOT be added manually** to Keycloak.

Each team must:

- Prepare an Excel file containing: Username, Email, Password, Assigned Role
- Import users: Using Keycloak Admin REST API or Keycloak bulk import tools

**4.3 Custom Keycloak Login Page (Mandatory)**

Each team must:

- Design a **custom Keycloak login theme**
- Add: System logo, System name, Custom colors
- The design must reflect the selected system (hospital, library, university, etc.)

**4.4 Role-Based Access Control (RBAC) – Mandatory**

Each project must contain **at least three different roles**, for example:

| Role | Allowed Operations |
|---|---|
| **Role 1** | Read Only |
| **Role 2** | Create, Read, Update |
| **Role 3** | Full CRUD |

Roles must be:

- Created in Keycloak
- Assigned to users through Keycloak
- Enforced in the backend APIs

**5. System Type (Choose One Only)**

Each team must choose one system type:

- Library Management System
- University Management System
- Hospital Management System
- Any similar management system

Important:
The grading focuses on **security, identity, and authorization**, not system complexity.

**6. Database Requirements**

Each project must include:

- At least **three related database tables**
- Full CRUD operations
- Role-based access enforcement on database operations
- Access to the database must be fully protected using JWT tokens

**7. Minimum Functional Requirements**

Each project must implement all the following:

- Secure login using Keycloak

- Token-based API protection
- Role-based API authorization
- Database CRUD operations
- Excel-based user import
- Custom Keycloak login page
- Secure frontend-backend communication using access tokens
- Correct HTTP status codes:
    - 200 OK
    - 201 Created
    - 401 Unauthorized
    - 403 Forbidden

**8. Team Requirements (Strict Policy)**

- Minimum team size: **5 students**
- Maximum team size: **7 students**
- No exceptions are allowed

**9. Project Discussion and Presentation**

- Teams will be divided into **two discussion groups**:
    - Group 1: Week 14
    - Group 2: Week 15
- Each team must select their discussion group using the **attached Google Form link**
- During the discussion, each team must present:
    - System architecture
    - Keycloak configuration
    - Login flow
    - User and role management
    - Excel user import process
    - Role-based authorization

- Live system demonstration

**10. Project Submission Requirements (Mandatory)**

Each team must submit **one public GitHub repository** containing:

- Frontend source code
- Backend source code
- Database scripts
- Keycloak configuration details
- Excel file used to import users
- Custom Keycloak theme files
- README.md file including:
  - Team members
  - System architecture
  - Setup and run instructions
  - Test user credentials
  - Role descriptions
  - Screenshots of the system

**11. Evaluation Focus Summary**

Grading will focus primarily on:

- Keycloak integration quality
- Excel user import correctness
- Role-based access control implementation
- Secure token verification in the backend
- Database access protection
- GitHub organization and documentation
- Team collaboration

User interface design is not the main grading factor.