Logo

Description automatically generated

**Course Name:** Computing with Software Development

**Year/Group:** Year 4

**Semester:** 7

**Module Title:** Final Year Project

**Project Coordinator:** Margaret Finnegan

Learning Management System / Virtual Learning Environment

**Student Name:** Keneith June Atillo

**Student Number:** X00190944

**Student Email:** X00190944@mytudublin.ie

Table of Contents

[Section 1: Detailed Discussion 3](#_Toc179120115)

[System Functionality 3](#_Toc179120116)

[User Roles 4](#_Toc179120117)

[Section 2: Existing Applications in Domain 4](#_Toc179120118)

[Section 3: Platform, Technologies, and Libraries 6](#_Toc179120119)

[Frontend 6](#_Toc179120120)

[Backend 6](#_Toc179120121)

[Database 7](#_Toc179120122)

[Authentication and Security 7](#_Toc179120123)

[Development and Collaboration Tools 7](#_Toc179120124)

[Hosting and Deployment 7](#_Toc179120125)

[Section 4: Risks 8](#_Toc179120126)

# Section 1: Detailed Discussion

The proposed project idea is a Learning Management System (LMS) or Virtual Learning Environment that can be used for college/universities to manage and deliver educational content to learners/students.

## System Functionality

The application should have a number of key features which would include most if not all of the following:

* **User Management** – users are able to securely register and login. Each user will have roles (e.g., lecturers, student) determining their permissions. Users will also have profiles that they can set up and edit and is viewable by other users.
* **Course Management** – lecturers should be able to create materials for the modules that they teach such as generating quizzes and assignments. They should also be able to upload resources such as PDF, PowerPoint, etc. for lectures that students can download. Ideally the application should be able to support multiple content types that can be accessed by students. For students, they should be able to browse available modules and enrol. Alternatively, an admin user can assign students to a specific course/module as well.
* **Assessment and Evaluation** – as mentioned above, instructors should be able to generate quizzes and define scoring rules. Students can submit quizzes, lab work or CA uploads that lecturers can review grade, provide feedback. Grading can be down automatically for multiple choice quizzes for example but manually grading is of course available.
* **Progress Tracking and Reporting** – students should be able to see their progress in each module with some visuals. For lecturers and even administrators, they should be able to access reports on course/module performance, student engagement, and success rates.
* **Communication and Collaboration** – if time permits, a discussion forum or real-time chat would be an excellent addition so students and instructors can communicate privately or even among students as well. Having a real-time chat could eliminate the use of external applications students would use such as Discord.
* **Administration and System Management** – admins should be able to manage all users and assign roles. They can also monitor courses/modules and view analytics.

## User Roles

The application will have three types of users which are administrators, instructors/lecturers, and students with different permissions and access.

* **Administrator** – they have full control over the application and are able to manage users, oversee courses/modules, and access analytics. In the admin dashboard, it should display key metrics such as total users, active courses/modules, recent enrolments, and overall application performance.
* **Instructor/Lecturer** – they create and deliver course content, assess students, and facilitate communication within their modules. In their dashboard, it should show an overview of their module, enrolled students, pending submissions, and upcoming deadlines.
* **Student** – they the mainly the main users of the platform. Once they have enrolled to a course/module they can start learning by engaging with course/module content, completing tasks, and interacting with lecturers and peers. In their dashboard, it should show what course/module they are enrolled in, what their current progress is, upcoming deadlines, and notifications.

# Section 2: Existing Applications in Domain

Using **Brightspace** created by D2L (Desire2Learn) and was just recently rolled out in college for comparison.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Brightspace** | **Project** | **Comment** |
| **User role** | Admin, instructor, student, and other customisable roles | Admin, instructor, and student only | Brightspace offers more user roles |
| **Course management** | Full course creation with modules, content upload etc. | Similar course creation with modules and content upload | Core functionality is similar for both |
| **Assessment types** | Supports quizzes, surveys, and various assignments | Supports quizzes and assignments. Surveys can be included | Similar in providing assessments but Brightspace may offer more types |
| **Grading system** | Comprehensive grading with rubrics and grade book features | Manual and auto-grading only | Brightspace has a more extensive grading system |
| **Content delivery** | Supports videos, documents, SCROM packages, and interactive media | Supports documents and other standard media possibly even videos as well | Brightspace offers SCORM compliance |
| **Discussion forum** | Course-based forms and discussion threads | May include course-specific forums with instructor moderation | Brightspace have advanced discussion feature option |
| **Communication Toos** | Email, notifications, announcement, real-time chat | Email, notification, real-time chat (optional) | Similar communication features |
| **Progress tracking** | Detailed tracking with analytics, dashboards, and reports | Progress visualisation and basic reports for student and instructors | Brightspace offers more advanced analytic tools |
| **Analytics and reporting** | Detailed analytics and reporting for course and users | Basic reporting on student progress and engagement | Brightspace offers more in-depth and customisable analytics |
| **Integration with external tools** | Integrates with third-party tools like Google Workspace, Zoom | May be possible to include depending on feedback and feasibility | Brightspace has more out-of-the box integrations |
| **Accessibility features** | Strong accessibility features compliant with WCAG | Could be a great addition as well | Brightspace offers comprehensive accessibility support |
| **Scalability and performance** | Highly scalable and robust for large institution | Can be scalable with cloud support depending on implementation | Brightspace is designed for enterprise-level scalability |

# Section 3: Platform, Technologies, and Libraries

A list of platforms, technologies, and libraries that is can be used for the project. Some may not be used, and some may be added or switched out instead. The main technologies that are definitely going be used are Angular and ASP.NET Core which I have some familiarity with as these were the technologies used in the company that I worked with during work placement.

## Frontend

* **Angular** – Typescript-based framework used for developing Single Page Applications (SPA).
* **Angular Material** – A UI component library based on Material Design principles to create responsive and modern user interfaces.
* **RxJS** – library for reactive programming allowing to handle asynchronous data streams like API calls and event handling.
* **NgRx** – a library for state management in Angular used for maintaining a predictable state within the LMS.
* **HTML5, CSS3, SCSS** – standard web technologies for structuring and styling the user interface.
* **Jasmine/Karma** – for unit testing Angular components, services, and modules.

## Backend

* **ASP.NET Core** – a cross-platform, high-performance framework for building RESTful APIs and backend services.
* **Entity Framework Core (EF Core)** – an Object-Relational Mapping (ORM) framework to handle database interactions.
* **ASP.NET Identity** – a membership system to manage user authentication, authorization, and roles.
* **SignalR** – a library for adding real-time functionality such as notifications or chat features.
* **Hangfire** – a background job scheduling and processing library that will handle tasks such as sending scheduled emails or processing background jobs such as automated reminders.
* **Swagger (Swashbuckle)** – auto-generating API documentation and testing endpoints.
* **xUnit/NUnit** – for unit testing backend logic, services, and controllers.

## Database

* **Microsoft SQL Server** – a powerful relational database (RDBMS) used for storing course/module data, user data, quiz, results, and other data for the platform.
* **Entity Framework Core (EF Core)** – as mentioned before, EF core will manage database interactions.
* **SQL Server Management Studio (SSMS)** – a comprehensive tool for managing SQL server databases. I have not used it before but was mentioned during work placement.

## Authentication and Security

* **OAuth2 and OpenID Connect** – for securing user authentication and support for external login providers such as Goole and Microsoft.
* **JWT (JSON Web Token)** – to secure API endpoints.
* **SSL/TLS** – secure data transmission.
* **Data Encryption** – using hashing and encryption mechanisms to secure store data such as passwords.

## Development and Collaboration Tools

* **Visual Studio 2022** – for C# .NET development.
* **Visual Studio Code** – for developing the frontend with Angular.
* **Azure DevOps** – preferably I would like to use Azure DevOps for hosting repositories although GitHub will also work.
* **Postman** – for API client testing.

## Hosting and Deployment

Preferably, I would like to use **Microsoft Azure** services for hosting both frontend and backend including managing CI/CD pipelines, source control, and project management.

# Section 4: Risks

The project uses reliable and well known technologies which very unlikely to pose any risk in its development although the possibility is very low, there is still a small chance for something wrong to happen. For example, the project heavily relies on SignalR for real-time features, Hangfire for background processing, and Entity Framework Core for database interactions. If any of these libraries face compatibility issues, bugs, or deprecated, it could cause massive impact on core features such as real-time notifications, background task management, and especially data access.

The chosen cloud provider for deployment and services for the project is Microsoft Azure. Any downtime and service limitations could affect the availability and reliability of the platform but again considering the service is owned by Microsoft this is extremely unlikely to happen.

There are also some risks with Angular, .NET Core and any library that is associated with it that are regularly updated. Keeping up with these updates could pose challenges especially if breaking changes are introduced. There’s also the risk of data loss with the project relying on SQL server for storing critical data but all of these can easily be mitigated.