Logo

Description automatically generated

**Course Name:** Computing with Software Development

**Year/Group:** Year 4

**Module Title:** Final Year Project

**Project Title:** Virtual Learning Environment (VLE)

**Project Coordinator:** Margaret Finnegan

**Project Supervisor:** Ben Walsh

Use Case and Logical Architecture

**Student Name:** Keneith June Atillo

**Student Number:** X00190944

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

Table of Contents

[Section 1: Use Cases 3](#_Toc180329496)

[User Registration 3](#_Toc180329497)

[User Login 3](#_Toc180329498)

[Module Creation 4](#_Toc180329499)

[Module Enrolment 4](#_Toc180329500)

[Assignment Creation 5](#_Toc180329501)

[Assignment Submission 6](#_Toc180329502)

[Section 2: Logical Architecture 7](#_Toc180329503)

[Logical Architecture Diagram 7](#_Toc180329504)

[Component Details 8](#_Toc180329505)

[Angular Application 8](#_Toc180329506)

[ASP.NET Core API 8](#_Toc180329507)

[Azure SQL Database 8](#_Toc180329508)

[Azure Blob Service 8](#_Toc180329509)

[Hangfire Service 9](#_Toc180329510)

[SignalR Service 9](#_Toc180329511)

# Section 1: Use Cases

## User Registration

|  |  |
| --- | --- |
| **Title (Goal)** | User Registration |
| **Primary Actor** | Admin, Instructor, Learner |
| **Story (Successful Scenario)** | 1. The user navigates to the registration page. 2. The user fills in the required fields. 3. The system checks if the email is already registered. 4. The system creates a new account. 5. The user is able to login. |
| **Alternative Scenario** | **Step 2**  The user entered an invalid value for one or more of the input fields (e.g., a field is left blank or the password provided is too weak). An error message is displayed on the screen to inform the user.  **Step 3**  The email provided is already in use. An error message is displayed on the screen to inform the user. |

## User Login

|  |  |
| --- | --- |
| **Title (Goal)** | User Login |
| **Primary Actor** | Admin, Instructor, Learner |
| **Story (Successful Scenario)** | 1. The user navigates to the login page. 2. The user inputs their email/username and password. 3. The system verifies the credentials. 4. The user if logged in and redirected to their dashboard based on their assigned role. |
| **Alternative Scenario** | **Step 2**  The user entered the wrong email/username and/or password. An error message is displayed on the screen to inform the user.  **Step 3**  The account associated with the email is not found in the database. An error message is displayed and suggests the user to navigate to the registration page first and try logging in again after registering successfully. |

## Module Creation

|  |  |
| --- | --- |
| **Title (Goal)** | Module Creation |
| **Primary Actor** | Instructor |
| **Story (Successful Scenario)** | 1. The instructor logs into the system. 2. The instructor navigates to the module creation page. 3. The instructor provides the module details. 4. The instructor uploads any relevant materials. 5. The system saves the module information and is published making it available to the learners. |
| **Alternative Scenario** | **Step 3**  The instructor does not provide a value for a required field (e.g., module name). An error is displayed on the screen and informs the user. |

## Module Enrolment

|  |  |
| --- | --- |
| **Title (Goal)** | Module Enrolment |
| **Primary Actor** | Learner |
| **Story (Successful Scenario)** | 1. The learner logs into the system. 2. The learner navigates to the discover modules page. 3. The learner searches for a module using filters. 4. After clicking/selecting a module from the results, the learner can then click the “Enrol” button. 5. The system adds the module to the learner’s module list. 6. The learner has now access to module materials. 7. The background process will send an email notification to confirm enrolment. |
| **Alternative Scenario** | **Step 4**  The learner is already enrolled in the module. An information message is displayed to inform the user that they are already enrolled and/or the “Enrol” button is disabled. |

## Assignment Creation

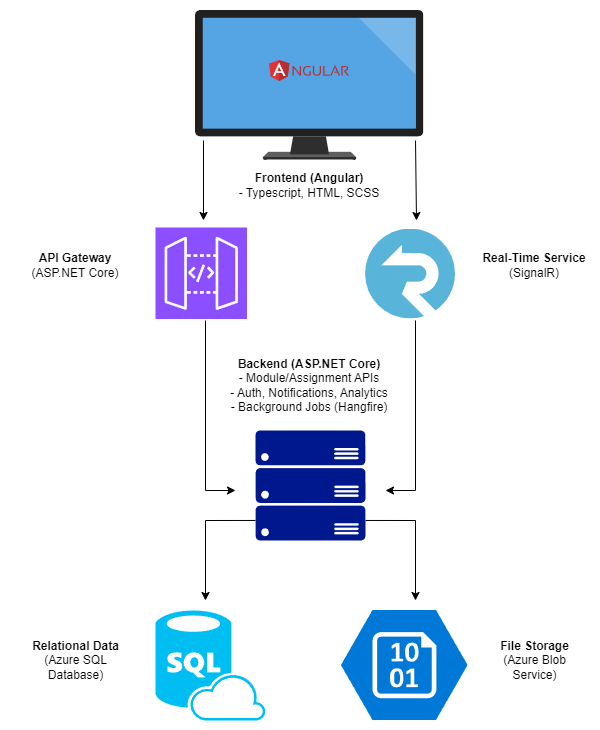
|  |  |
| --- | --- |
| **Title (Goal)** | Assignment Creation |
| **Primary Actor** | Instructor |
| **Story (Successful Scenario)** | 1. The instructor logs in and navigates to the assignments section. 2. The instructor selects/clicks the “Create Assignment” and fills out the details. 3. The instructor can upload reference materials or assignment templates. 4. The system saves the assignment and displays it in the module page for learners to view. 5. The learners receive notifications about the new assignment via email and/or push notification. |
| **Alternative Scenario** | **Step 2**  The instructor fails to provide one or more required fields. An error message is displayed on the screen to inform the user.  **Step 2**  The date for the assignment deadline is invalid. An error message is displayed on the screen to inform the user. |

## Assignment Submission

|  |  |
| --- | --- |
| **Title (Goal)** | Assignment Submission |
| **Primary Actor** | Learner |
| **Story (Successful Scenario)** | 1. The learner logs in and navigates to the assignments section. 2. The learner views the assignment details and uploads their submission. 3. The system stores the submission and records the submission timestamp. 4. The learner can see the submission status that the submission was successful. 5. The background process will send an email or push notification to confirm that a submission has been made to the learner and maybe to the instructor as well. |
| **Alternative Scenario** | **Step 2**  An error occurs when uploading a file during submission. An error message is displayed on the screen to inform the user.  **Step 3**  The learner has submitted their material late. The system displays the submission status as “Late”. |

# Section 2: Logical Architecture

## Logical Architecture Diagram



1. The user interacts through the Angular application.
2. The Angular application sends a request to the ASP.NET Core Web API.
3. The data is fetched or stored in the Azure SQL database.
4. The result is sent back to the API then eventually to the Angular application.
5. For real-time updates, SignalR is used to push updates to the Angular application.
6. For background processes, Hangfire schedules and executes the tasks.
7. The API also interacts with Azure Blob to store and retrieve large files.

## Component Details

### Angular Application

The frontend application where the user interacts with.

* **Languages:** TypeScript, HTML, SCSS
* **Deployment:** RESTful API
* **Security:** HTTPS

### ASP.NET Core API

Handles the requests from the Angular application to the database and vice versa.

* **Languages:** C#, .NET Core
* **Deployment:** RESTful API
* **Security:** HTTPS, OAuth

### Azure SQL Database

Stores structured data for the application.

* **Languages:** SQL
* **Deployment:** JDBC
* **Security:** Managed by Azure

### Azure Blob Service

Store large files such as course materials.

* **Languages:** Binary Data
* **Deployment:** RESTful API
* **Security:** HTTPS, Azure Storage Account Keys

### Hangfire Service

Background process that executes scheduled tasks and jobs.

* **Function:** Background Jobs
* **Deployment:** Sessionless
* **Security:** HTTPS

### SignalR Service

Provides real-time updates and chat functionality.

* **Protocol:** WebSocket
* **Deployment:** Sessionless
* **Security:** HTTPS