Project Report

Student Registration Application Development with .NET Core and Angular

Group: 33

Members: Arif Ozden, Tekin Bal, Ali Riza Keles

Table of Contents

1. **Introduction:**

    - A brief overview of the project's objectives and scope
    - Used technologies in the project and their versions

2. **Project Overview:**

    - The application's purpose and functionality
    - The MVP (Minimum Viable Product) approach
    - Models-Controllers-Components
    - The CRUD operations and the chosen entity for implementation

3. **Architecture and Design:**

    - The architecture of the application (backend and frontend)
    - UML diagram that represents the system's structure

4. **Implementation Details:**

    - **Backend (.NET Core):**
        - The backend structure, modules, and CRUD operations
        - Error handling and logging
    - **Frontend (Angular):**
        - The frontend structure and components
        - Dynamic content, form validations, and error handling

5. **Integration and Functionality:**

    - **I**ntegration of the frontend and backend components
    - Content filtering

6. **Quality Assurance:**

7. **Deployment and Access:**

    - GIT

8. **Code Attribution and Documentation:**

    - Code snippets or inspirations used from other sources
    - Architecture, functionality, and coding practices in the project

9. **Conclusion:**

    - The achievements and challenges during the project
    - Overall outcome and future enhancements areas

10. **References:**

    - External sources, frameworks, or resources used during the project.

1. Introduction:

A brief overview of the project's objectives and scope.

We created a web-based student registration system in the project. For this purpose, we created a student model and created variables regarding the information that should be received in the student registration system. Our expectation from this system is to keep student records in the database with a simple and user-friendly interface.

Used technologies in the project and their versions.

We have backend, frontend and database components in our project.

In the backend, we used the net.core 6.0 framework, which is widely used around the world and especially in Norway. We used Visual Studio to make this part.

EFC.SqlServer
EFC.Tools

• On the front end, we used version 16 of Angular Framework, which is widely used in Norway. We used Visual Studio Code to make this part.

Angular.Forms
Angular.Router
Angular.OnInit
Angular.Core

• We created the database infrastructure by using Microsoft SQL Server Management Studio program.

2. Project Overview:

The essence of the application is to enter student records into the database. MVP aims to provide users with the ability to use CRUD and manage assets. We worked on a simple and useful project to meet a need we identified. We created our project by solving a technical problem with a simple method.

Model: StudentModel
Controller: StudentController
Components: HomeComponent, StudentComponent, Add-StudentComponent, Edit-StudentComponent

CRUD operations

The application performs CRUD operations using various methods in StudentController and StudentService. It also gives feedback such as error messages when there is incomplete input.

All elements of CRUD are used here. That is, Create (Create), Read (Details), Update (Edit), Delete (Delete).

Students are listed on them and a user-friendly interface is presented.
With the filter method, you can filter the students according to their departments or entry years, or you can search for students by entering the names of the students you are looking for with the Search method.

To add a new student, a new student can be added to the database by using the addStudent method. If the student wants to make changes to their information, they can edit the student using the edit method or delete the student record completely.

3. Architecture and Design:

• Backend Architecture (Using .NET Core)

The backend architecture was designed with scalability and maintainability in mind. First, we tried to prepare UML diagrams.

We showed the interactions between modules, endpoints and database in the system architecture.
We have prepared the back end equivalents of the methods we will use in the front end of our project.

We thought of ensuring that the information to be entered in the front end form is transferred to the database by providing a back end - database connection.

• Front End Structure (Angular)

Angular's component-based structure made it easier for us to create a simple and user-friendly interface. design, tidy and focused on user interactions, emphasizing a seamless experience across devices and screen sizes.

• Controller and Methods

In the studentcontroller we wrote our methods on the backend. We have following methods:

- GelAllStudents: By using this method all students in the database shows on the StudentList page.

- AddStudent: By using this method, user can add a new student to the database and the new added student comes to the studentlist.

- GetStudent: For the editstudent and deletestudent methods system has to bring the student with the given id, and for this operation user can use this method.

- EditStudent: By using this method user can update information for the chosen student.

- DeleteStudent: By using this method user can delete the student with the chosen id.

In the students.service we wrote the equivalents methods on the frontend with same names.

• Model

Id: The unique identifier of the topic
Student name
Student surname

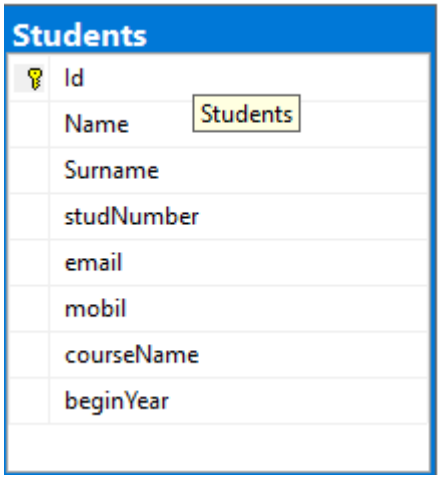
Student number

Student email

Student mobil

Students course name

Students beginning year



• Components

- HomeComponent
- StudentComponent
- Add-StudentComponent
- Edit-StudentComponent

On this app we used Bootstrap to create a responsive and user-friendly design. On the app we also used Font-Awesome for icons. The app has a simple and straightforward layout with a navigation bar at the top and a main content area in the middle. The app has a white background and scheme with black text.

4. Implementation Details:

• Backend (.NET Core):

we tried to configure the backend code base, adhering to coding rules and design patterns, trying to implement good practices. CRUD operations were systematically integrated to ensure data consistency and reliability. We thought of trying to embed error handling mechanisms into our project and warn the user in case of possible user login errors so that they can correct their errors.

• Frontend (Angular):

Frontend components were prepared with a focus on modularity and reusability. Form validations and real-time updates have been implemented to improve data integrity and user experience. Error handling strategies have been meticulously designed to provide a seamless and error-free interface.

5. Integration and Functionality:

The functionality of the application is evident by the harmony between the front-end and back-end. We did our best to ensure more effective participation of users and to solve the identified needs. While doing all this, we tried to follow the developments in Net.core and Angular technologies and benefit from their current and innovative features.

6. Quality Assurance:

Quality assurance was a phase of our project that we attached great importance to. At this stage, we did some tests to verify the reliability of the application. We encountered some difficulties in our tests, but to resolve these difficulties, we did some research on the internet and tried to increase the performance of the application by making the necessary improvements.

7. Deployment and Access:

By uploading our application to the git platform, we thought of providing easier access to those who will control our project.

8. Code Attribution and Documentation:

We carried out our work in accordance with functionality and coding standards. We presented external sources or inspired codes used during the project with transparent code attribution.

9. Conclusion:

Ultimately, the project demonstrated our team's perseverance and determination in leveraging .NET Core and Angular to design an application. While working on the project, we experienced some problems with the update of visual studio, and we did research to fix these problems. We had a very useful learning experience in terms of capturing future innovation and improvement opportunities while delivering a tangible product.

10. Contribution List:

Arif Özden, Tekin Bal and Ali Riza Keles did the coding in all documents. Its design and front end were made by Arif Özden, and its documentation was done by Tekin Bal and Ali Riza Keles.

11. References:

List of references external sources, frameworks, or resources employed throughout the project.

Chatgpt was used to write descriptions of some items.
https://www.w3schools.com/
https://stackoverflow.com/

https://angular.io/

https://learn.microsoft.com/en-us/dotnet/core/tutorials/

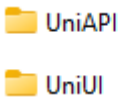
https://learn.microsoft.com/en-us/sql/sql-server/tutorials-for-sql-server-2016?view=sql-server-ver16

https://www.youtube.com/

https://getbootstrap.com/

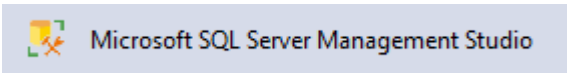
https://jquery.com/

How to open project?



UniAPI with Visual Studio

UniUI with Visual Studio Code



Microsoft SQL Server Management Studio for database