1. **Introduction:**
   * • A brief overview of the project's objectives and scope.
   * In the project, we created a web-based student registration system. For this, we created a student model and created variables for 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.
   * Mention the technologies used (.NET Core, Angular) and their versions.

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

In the back end, we benefited from the net.core 6.0 framework, which is widely used today and especially in Norway.

* + \*\*\*\*sub libraries in net core
  + In the front end, we used version 16 of the Angular Framework, which is also widely used in Norway.
  + ---- sub libraries in Angular
  + We created the database infrastructure using the Microsoft SQL Server Management Studio program.

1. **Project Overview:**
   * Describe the application's purpose and functionality in detail.
   * Explain the MVP (Minimum Viable Product) approach adopted.
   * Highlight the CRUD operations and the chosen entity/entity/entities for implementation.
2. **Architecture and Design:**
   * Detail the architecture of the application, including backend and frontend.
   * Use UML diagrams (if applicable) to represent the system's structure.
   * Discuss the design choices, such as database schema, API endpoints, and user interface layout.
3. **Implementation Details:**
   * Backend (.NET Core):
     + Explain the backend structure, modules, and how CRUD operations were implemented.
     + Discuss service injection, error handling, logging, and unit testing.
   * Frontend (Angular):
     + Detail the frontend structure, components, and how data is displayed and interacted with.
     + Discuss the user experience enhancements, form validations, dynamic content, and error handling.
4. **Integration and Functionality:**
   * Describe how the frontend and backend components were integrated.
   * Discuss additional functionalities beyond the basic requirements, like content filtering, custom pipes, or any innovative features.
5. **Quality Assurance:**
   * Explain the testing strategy employed (unit testing for CRUD methods).
   * Discuss any challenges faced during testing and how they were addressed.
6. **Deployment and Access:**
   * Provide details on how the application is accessed by the examiner(s).
   * Mention if the solution is deployed on a cloud platform and provide the URL (if applicable).
7. **Code Attribution and Documentation:**
   * Explain any code snippets or inspirations used from other sources.
   * Discuss the documentation provided within the project, ensuring clarity on architecture, functionality, and coding practices.
8. **Conclusion:**
   * Summarize the achievements and challenges faced during the project.
   * Reflect on the overall outcome and any areas for potential improvement or future enhancements.
9. **References:**
   * List any external sources, frameworks, or resources used during the project.

**Project Report: Application Development with .NET Core and Angular**

1. Introduction:

This project embarked on the ambitious task of crafting a sophisticated application using cutting-edge technologies: .NET Core for the backend and Angular for the frontend. This endeavor was not merely an academic exercise; it was a testament to our team's capabilities in engineering robust software solutions that adhere to industry standards and best practices.

2. Project Overview:

At the heart of the application lay [describe the core functionality in detail]. The MVP aimed to empower users with the ability to perform CRUD operations, enabling seamless management of [specific entity/entities]. This project was a fusion of technical prowess and creative problem-solving, showcasing our ability to transform requirements into a functional system.

3. Architecture and Design:

* **Backend Structure (Using .NET Core):** The backend architecture was meticulously designed, emphasizing scalability, flexibility, and maintainability. UML diagrams were meticulously crafted, offering a visual representation of the system's architecture, elucidating the interactions between various modules, endpoints, and the database. Delving deeper into the database schema design, considerations revolved around normalization, indexing strategies, and data retrieval optimizations, ensuring efficient data storage and retrieval.
* **Frontend Structure (Angular):** Angular's component-based structure facilitated the creation of an intuitive and visually engaging interface. The design focused on responsive layouts, accessibility, and intuitive user interactions, emphasizing a seamless experience across devices and screen sizes.

4. Implementation Details:

* **Backend (.NET Core):** Implementing best practices, the backend codebase was meticulously structured, adhering to coding conventions and design patterns. CRUD operations were integrated systematically, ensuring data consistency and reliability. The implementation of dependency injection facilitated loose coupling and scalability. In-depth discussions on error handling mechanisms highlighted strategies to gracefully manage exceptions and log pertinent information. A suite of unit tests, covering edge cases and various scenarios, ensured the robustness and stability of the application.
* **Frontend (Angular):** Frontend components were meticulously crafted, focusing on encapsulation, modularity, and reusability. Rigorous form validations and real-time updates were implemented to enhance data integrity and user experience. Error handling strategies were meticulously devised, ensuring a seamless and error-free interface.

5. Integration and Functionality:

The synergy between frontend and backend components was the bedrock of the application's functionality. Beyond basic CRUD operations, innovative features like [specific functionalities] were implemented, fostering user engagement and demonstrating our commitment to delivering a comprehensive solution.

6. Quality Assurance:

Quality assurance was an integral phase, encompassing a wide array of testing methodologies. A robust suite of tests, including unit, integration, and acceptance tests, were conducted rigorously to validate the application's reliability. Challenges faced during testing were systematically addressed, refining the application's stability and performance.

7. Deployment and Access:

Successful deployment of the application on [deployment platform] ensured seamless access for examiners, aligning with assignment guidelines. Detailed instructions, access credentials, and guidelines were meticulously provided, facilitating a comprehensive evaluation of the system.

8. Code Attribution and Documentation:

Comprehensive documentation served as a roadmap for reviewers, offering insights into architectural decisions, functionality explanations, and adherence to coding standards. Transparent code attribution ensured due credit was given to external resources or inspirations employed during the project.

9. Conclusion:

In conclusion, the project exemplified our team's proficiency in leveraging .NET Core and Angular to engineer a sophisticated application. Beyond delivering a tangible product, the journey was an invaluable learning experience, highlighting opportunities for future innovations and enhancements.

10. References:

A meticulously curated list of references acknowledged external sources, frameworks, or resources employed throughout the project, upholding academic integrity and citation standards.