(Note fra luca – teams versionen kan ændres tjek en ekstra gang)

Exam project

2025_2_web

Write a report showing your reflections and considerations with regards to the topic and the project

If you would like to write the report in a group:

1 student - min. 10 - max. 15 pages
2 students - min. 15 - max. 20 pages
3 students - min. 20 - max. 25 pages
4 students - min. 20 - max. 30 pages

The exam is still individual.

Questions only in the public channel for this course

Hand-in

On Wiseflow as PDF. You will get an email from the administration.

You must also hand-on the code for your application as a ZIP file.

All the points below are mandatory and must be part of your report. Failure to deliver it will not allow you to attend the exam and therefore get a 00.

The censor and the teacher already know the theory; we need to know how you use it in your system.

The report

You should cover various aspects of the development process, design decisions, user experience considerations, and technical details. Here's an outline with points to guide them, as well as suggestions on how they can reflect on each:

1. Introduction

What is the purpose of the web application?
Reflect on the problem the application is solving and why it is relevant.

Target audience

Who are the primary users of the application? How does the app cater to their needs?

Goals of the project

What did you aim to achieve? Were there specific challenges set to solve?

2. Requirements and Planning

Project Requirements
What were the functional and non-functional requirements? Reflect on how they determined these and how they prioritized them during development.

Research

What research did they conduct to understand the needs and design?

Planning and Timeline
Reflect on how you broke down the project into tasks, assigned responsibilities (if working in teams), and kept track of deadlines.

3. System Design

Architecture.
What architecture did you choose for the application (e.g., client-server, microservices, MVC)? Reflect on why they made this choice and any trade-offs.

Tech Stack
What technologies were used (front-end, back-end, database, frameworks)? Why were these technologies chosen, and how well did they work together?

Wireframes and Mockups
Discuss initial designs, wireframes, or mockups. What did you aim to achieve with the UI/UX, and how did they refine it over time?

4. Frontend Development

User Interface Design
Reflect on the user interface (UI) design. Was the design intuitive and user-friendly? How did they approach layout, color schemes, and typography?

Technologies Used
What front-end frameworks or libraries did they use (e.g., React, Angular, Vue.js)? Why were these chosen, and what challenges did they face with these technologies?

Responsive Design
Did you ensure the application is responsive (mobile-first design)? How did they test the application on different devices?

5. Backend Development

Server-Side Logic
What backend language or framework did you use (e.g., Node.js, Django, Ruby on Rails)? Reflect on how the backend was structured and how you handled server-side logic.

Database Design
What type of database did they choose (e.g., SQL, NoSQL)? Discuss how the database schema was designed, how relationships were modeled, and any challenges in data management.

API Integration
Did you integrate with any external APIs? If so, which ones, and how did they handle authentication, rate-limiting, and error handling?

6. Security Considerations

Data Security
How did you ensure the security of data? What steps did they take to protect sensitive information (e.g., encryption, HTTPS, placeholders for queries)?

Authentication and Authorization
What method did you use for user authentication (e.g., OAuth, JWT, sessions)? How did they ensure that only authorized users could access certain features?

Common Vulnerabilities
Did you address common web application vulnerabilities like XSS, CSRF, SQL Injection? How did you test for and mitigate these risks?

7. Testing and Quality Assurance

Testing Strategy
What types of testing did you implement (unit testing, integration testing, manual testing, user acceptance testing)? Reflect on how these helped ensure the app's functionality.

Bug Tracking and Fixes
How did you track bugs and issues during development? What steps were taken to resolve them?

Performance Testing
How did you test for performance issues, such as load times or server response times? What optimizations were made? Consider the use of the Google sheet for the languages, how did you make the translation faster?

8. Deployment

Deployment Process
How did they deploy the web application (e.g., cloud platforms like AWS, Heroku, or PythonAnywhere)? What challenges did you face in this process?

Environment Setup
What different environments (development, staging, production) did they set up, and how did they manage environment variables and configurations?

9. Conclusion and Reflections

What went well?
Reflect on the aspects of the project that went smoothly. Was there a particular feature or part of the process they felt confident in?

What challenges were encountered?
Discuss the major challenges, whether technical, design-related, or time-related. How did you overcome them or plan to overcome them in the future?

Learning Experience
What did you learn from this project? This could be technical skills, project management, teamwork, or personal growth.

Suggestions for improvement
Based on what you have learned, how would they approach the project differently if you had to start again?

Additional Tips for Reflection:

Critical Thinking: Why did you make certain choices, and what were the consequences of those choices?

Evidence-Based Reflection: You should back up your reflections with concrete examples (e.g., code snippets, user feedback, performance metrics).

Every topic discussed during the class in theory or practice is part of the exam. Every link, video, post on Teams, Github repos, and in general everything we have done in the class is part of the exam.

The product

During the class we have coded a minimal "Twitter/X" clone. You can continue working on it or create a whole new application.

The application must have:

Front-end validation

Back-end validation

Multi-language page

Every query with placeholders

Logical division of files and folders

Comments in the code when needed

Good naming conventions

A good UX, so the "home" page doesn't reload when an action takes place

The user

A normal user in the application can:

Sign up – an email is sent to verify the account

Sign in

Sign out

Forgot password – an email with a link to reset the password is sent

Edit profile – name, email, avatar, etc

Add avatar image to profile

Change avatar image in the profile

Create a post – can be only text, or also contain some media (image, video, file)

Update a post

Delete a post

Search for something – could be a post, user, or anything you would like

Like/Unlike a post

Follow/Unfollow any other user

Write a comment on a post

Delete the account

The Administrator

The system must have at least 1 administrator than can:

Sign in

Sign out

See all users

Block/Unblock a user – an email is sent letting the user know about it

Block/Unblock a post - an email is sent letting the user know about it

Get the languages in the system via the Google sheet

You own the system

Since you own the system, you can add anything else to it. If you decide to build a whole new system, a whole new idea, then make the previous requirements match your idea