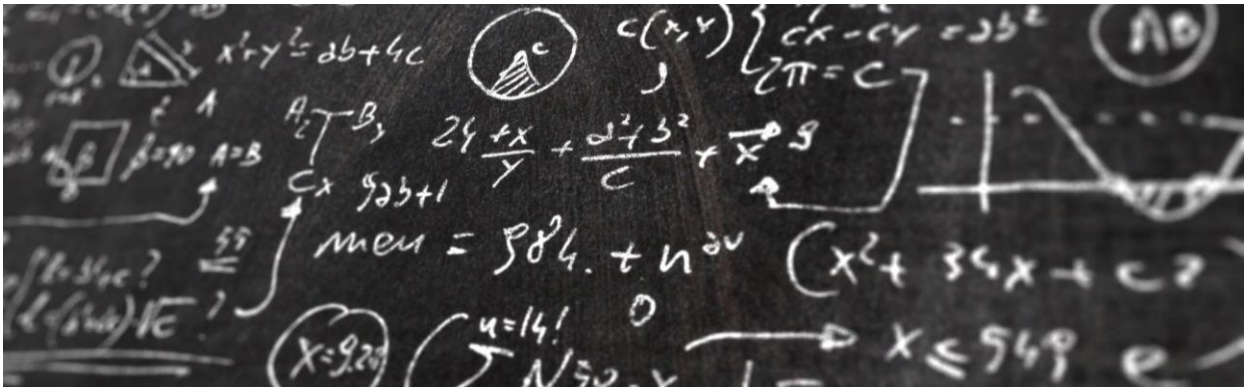

Semester Project - 2025

IDATT2105 Full-stack application development

Read this document carefully and be sure you take the details into account!

The project assignment is voluntary and can only be carried out by students who receive a C, and want to improve their grade to A or B. Note that all parts of the project (programmed solution, documentation, and presentation) are expected to maintain **very** good quality to achieve grade B or A.

The requirement is to implement a full-stack web application including

- A frontend using Vuejs (v3) related framework and libraries
- A backend using Java (v17,v21 or v24) and Spring boot framework (v3) / Spring framework (v6)
- **A database using MySQL (v8) and/or H2**

The project contains several parts and the more the group does - with good quality - the better! **So, every group should assess what they prioritize and complete the chosen functionalities as much as possible. Unfinished functionality will not be evaluated and should not be part of the delivery!** The evaluation will be based on good code and necessary documentation for testing and further development, also user interface and well-implemented functionality are important in assessment.

Note that plagiarism is not allowed. If the source code of two or more groups is too similar, the grade of all those groups may be adversely affected and can result in being banned from the study!

Note also that the total number of students per group is from **2 to 4**, and the expected effort per student is about 40 hours.

More on general and product requirements are described in the following.

General Requirements

1. The submission deadline is Friday **11th of April 2025, kl. 14:00.**
2. The required materials should be submitted in **Inspera**
3. One submission per group. Remember to write the names of all participants in the group.
4. Technical requirements

4.1. Clean CSS. No use of CSS frameworks like Tailwind
4.2. Spring Boot/REST based backend. Each endpoint should have proper authentication/authorization, for example based on JWT and Spring Security.
4.3. Database communication using Spring JDBC / JPA
4.4. The solution should contain tests. The code coverage shall be at least 50%
4.5. The group should make sure to use CI/CD during development.
4.6. OWASP and universal design principles/techniques should be used in the project
4.7. Session storage can be used to provide short-lived login session on the frontend

5. Documentations requirements (for each module)
    5.1. End points (API) must be documented, for example, using Swagger. Note, an explanation of what the endpoints do and what the different attributes are is required. In addition, code must be documented as usual (javadoc).
    5.2. System documentation is also a requirement., i.e., documentation that enables a new developer to quickly get the project up and running for testing and further development (architecture sketches/class diagram). Instructions for how to run the project can preferably be done as a README file, while other documentation should be as a PDF.
    5.3. Test data that can be used while testing the app, for example, test user credentials, database credentials, etc.
    5.4. The prerequisites must be documented if the project/module is dependent on other projects
6. Submission materials.
    6.1. A zip file including all the modules/files
    6.2. A runnable source code of each project/module. For example
        - Properly documented Source and Test files.
        - Configuration files like pom.xml (maven), package.json, Dockerfile, etc.
        - Flyway or normal DB schema scripts with test data
    6.3. **Description of how to run tests and get the system running. This should be made easy (for the user) - as a script, docker/maven command or the like.**

Product requirements / features

The product is a ‘finn.no´ like e-commerce marketplace – a web application where sellers can list their items and potential buyers can browse or search the items using various filters, can add items into their favorite lists, can communicate with sellers, and can also and buy items!

The layout and design of user-interface elements such as screens is open-ended and it´s up to each individual group's creativity. Students, however, can take inspiration from existing websites such as finn.no, prisguiden.no, prisjakt.no, and so on. UI screens should be designed such that they are mobile devices friendly in terms of size, pagination and scroll ability. In addition, the application shall have the following features.

1. The application shall have a main menu towards top of the screen with relevant menu items
    a. The main page also displays a list of main categories, and a list of recommended items based on user profile, browse history, etc.

2. Listing of items on the main (or in an inner) page can be done in scrollable, paginated, and thumbnail (default) view or map view.
    a. A frontend shall have support for multiple sizes of item thumbnails such as full width, half-width, quarter-width, double-height, etc.
    b. The frontend should retrieve items from the backend using APIs
    c. The backend shall provide APIs for adding, getting, updating, deleting such items.
    d. The backend APIs should be documented with Swagger.
3. Listed items can be filtered based on, for example, categories, locations etc.
4. Items can be searched based on item descriptions, locations, categories, published date, etc.
5. An item can have the following properties – id, brief description, full description, category, location (latitude, longitude), price, and so on based on the type of the item!
6. Upon clicking an icon thumbnail or map marker, an item details page shall be shown. In item details page,
    a. Shall include information about the item such as item description, price, location, contact information of the seller and so on.
    b. Shall include pictures gallery of the related item.
    c. Shall allow the logged-in user to bookmark the item, for example by clicking the bookmark logo displayed on the top-right corner of the items page.
    d. Shall allow the logged-in user able to contact the user who posted the item, for example via message inbox/dialogue.
    e. Shall allow the logged-in user to be able to either reserve the item or buy it instantly via VIPPS (based on configurations set by the seller). Note that test credentials for VIPPS shall be provided on the Blackboard.
7. The application (frontend) supports internationalization, i.e., supports multiple languages based on language settings on the browser
8. Users shall be
    a. able to register themselves,
    b. able to edit their profile including credentials
9. There will be two types of user roles
    a. Normal user - can list, buy/sell items, etc.
    b. Administrator – can add/update/modify categories, can do other administrative actions
10. A logged in user(s)
    a. can add new items, can update, delete, archive own listed items only
    b. can add items into their favorite or bookmarked list.
    c. can negotiate price on a listed item
    d. can buy an item listed by other users

Project presentation video (digital)

The project presentation video should be submitted digitally. Each group should create a 15–20-minute video to present their work comprehensively, with both the project itself and its key features presented plus the documentation, repo, issue-board and test-coverage discussed. More info will come closer to the presentation. The presentation video would be submitted together with the project files and repo link.

Questions about the assignment?

Use the forum on Blackboard. The idea, however, is that if the desired functionality is ambiguous, the group is largely free to interpret the task themselves. **The group should prioritize the most important functionality and finish that, before taking on more functionality. Therefore, make a list of your prioritizations and why.**