Exam in IDATA2301 Web Technologies and IDATA2306 Application Development

Versions

v1.0 – Initial revision, 2025-01-27.

Introduction

This document describes the examination in two courses:

- IDATA2301 Web technologies
- IDATA2306 Application development

The examination format of both courses is portfolio, which means that you submit a set of "assets" (files, links). The hand-in material in both courses is based on group projects developed throughout the semester. Therefore, part of the evaluation is common for both courses. However, each course has also separate elements which are basis for the examination. The following sections describe both the expected mandatory elements, optional features, deliverables to hand in as well as guidelines for grading.

Mandatory requirements – common

A complete website must be implemented, both the frontend and backend must support the whole process of product filtering:

- Landing page
- Product search (search form and search result list/grid)
- Choosing one specific product
- Product detail page (for the chosen product):
    - Product image
    - Product description
    - Product price for the different providers
- Placing an order (this action may be different for the different project themes, you need to come up with a solution on what this means for your project, present it to the customer (teachers).
- Login form
- For logged in users:
    - Regular users:
        - Can add a product to their favorite list.
        - Can see their favorite products.
    - Admin users:
        - Can show and hide a product. A hidden product does not show up on for the users.

Non-functional requirements

1. Students have worked in Sprints, with planning and reflection.
2. Version control used for source code management.
3. Regular work throughout the semester.

4. Code quality according to the best practice.
5. README file(s) for the project.

Mandatory requirements for frontend (IDATA2301)

1. Design guideline developed and consistently followed. The guideline can be updated during the semester, but the resulting website must be in sync with the final guideline. Also, the style must be consistent across the whole website. For example, it is not good to have different styles for different buttons.
2. Wireframes of page sections sketched.
3. Semantic elements used (*div* and *span* are last resort).
4. Responsive website, looking good on both desktop and mobile device.
5. Accessibility and usability considered. Lighthouse tests passing (for usability, not necessarily for SEO, Performance and PWA).
6. *Some* data loaded from a backend, using REST API calls.
7. *Some* interaction with JavaScript – DOM element modification.
8. The website looks aesthetically pleasing. This is hard to judge and mainly is seen together with usability. For example, are the text sizes according to the best practice, are colors having good contrast, do they make sense for the theme?
9. The style is fitting the given theme of the business.

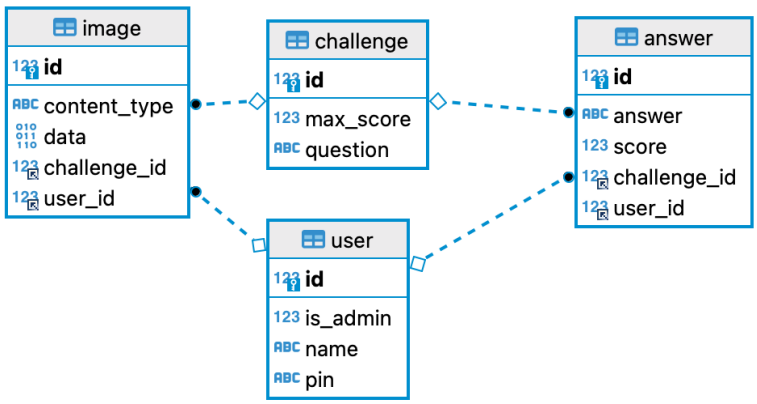
Mandatory requirements for backend (IDATA2306)

1. Necessary business logic implemented, supporting all the necessary user actions.
2. REST API endpoints for fetching necessary data: products, etc.
3. REST API endpoints supporting all four CRUD operations (Create, Read, Update, Delete) for at least one entity (even if all of them are not really used by the frontend)
4. Users with (at least) three different roles (groups): visitors (unauthenticated), authenticated users and administrators. To make the testing and exam grading easier, create the following predefined users:
    a. Regular user: username: dave, password: Dangerous2024
    b. Administrator user: username: chuck, password: Nunchucks2024
5. Authentication implemented with the following endpoints:
    a. An endpoint where users can authenticate (log in).
    b. 1+ endpoint accessible only to admins.
    c. 1+ endpoint accessible to both admins and regular users.
    d. 1+ endpoint accessible to everyone, including unauthenticated visitors.
6. Both backend and frontend secured with HTTPS.
7. Re-entrant test set – either Postman test collection or Java tests with MockMvc.
8. API endpoint documentation – either with Swagger, or in another form.
    a. All endpoints listed.
    b. Short description of the endpoint: the URL, expected request parameters, response code and response data.
9. Database password(s) and JWT secret key must be stored in environment (.env file or other files) outside of GIT. It is OK to store test-database user and password in GIT files (for automated testing), but not for production. This will be evaluated as part of good coding practice. Note: if you use GitHub actions for deployment (optional), store the usernames and passwords as GitHub secrets.

Deliverables to hand in

The following deliverables must be delivered in Inspera:

- Link to **source code GIT repository**. If you used separate repositories for the frontend and backend, provide link to both of those. If during the semester several students had "their own project repository" (which is a relatively bad practice – you should have collaborated on the same repo instead), provide all of those.
- **README.md file(s)** – at the root of the project folder. No need to hand in anything specifically, just make sure that you GIT repository contains a README file. You can have two separate READMEs (one for the backend, one for frontend) in the respective folders. The target audience for the README are developers and next students. Include the following in the README:
  - A short description of the project
  - "Getting started" – how to run the project, what is necessary (environment variables, database(s))?
  - One or a couple of screenshots of your website (for the frontend part).
- **Video presentation** (video file) of the project, see content requirements below.
- **URL of the website in production**. It is expected that the website is deployed on a server. The website must be accessible during the whole examination period (and eventual complaints (*klagesensur)*).
- **Sprint reports**. No formal definition of the format. This can be, for example a markdown file stored in the same Git repository. You need to show:
  - When did you work with the project – which weeks?
  - What was the goal in each sprint?
  - How was the work distributed among group members? Who got assigned to which tasks?
  - What was accomplished in each sprint?
- For IDATA2301 the following additional deliverables are required either as links to online-files or PDF files:
  - **Design guideline.**
  - **Wireframes.**
- For IDATA2306 the following additional deliverables are required:
  - **Database schema**. Show the table and relations between them (PDF or PNG file). You can generate the schema with different tools. For example, take screenshot of "ER Diagram" in DBeaver. An example of what is expected:



- **Postman test collection(s)** – either a link to the collections online or a .JSON file containing the test collection. If you wrote other types of API endpoint tests (for example, Java tests with MockMvc), write a comment about it instead of providing a link to the tests.

Content in the Video Presentation

In general – you need to show that you have worked professionally, have learned what was taught in the course and have implemented the necessary elements for the project.

You need to present that in a video with the length between 12-15 minutes.

The presentation can be in English or Norwegian. If the presentation is in Norwegian, please, provide an English transcript. The transcript must not be 100% accurate, but it should represent the same things you mention in your video. (i.e., the meaning must be the same)

You can choose whether you create two separate videos (one for IDATA2301, one for IDATA2306), or one longer video for both courses. Just make sure you cover all the expected elements in your video(s).

See the suggested presentation structure below. All the described parts must be in the presentation. However, you can freely adapt the sequence of explanation so that it best fits within your story.

Suggested Video Presentation Structure for IDATA2301 Web Technologies

- **Problem introduction**. What task did you get assigned, what is your theme? What kind of shop is this? (0.5 - 1 minute)
- **Work methodology**. How did you use the typical software developer techniques to develop your project (Sprints, version control)? (0.5 – 1min)
- **Architecture** of the solution. Explain the overall architecture – is it monolith or decoupled? On one server or on two different servers? What authentication do you use? What database is used? (0.5 – 1min)
- **Design decisions and agreements.** Have you made any specific design decisions? What have you discussed with the customer (teachers) during the semester? For example, what was decided regarding the search options, or the currency? Clarify these decisions here. (0.5-1min)
- **Design guideline**. Show the design guideline. Mention how much of it is used in the result. In case there are deviations, reflect on the reasons why the result differs from the plan. (1 min)
- **Wireframes**. Show the initial wireframes you have sketched. Shortly explain how you were thinking when sketching the whole page structure. (0.5 – 1min)
- **Demo of the functionality**. Start with showing the main workflow – from opening the page, to searching for products and placing an order. Afterwards you can show the extra functionality you have got. (3-6 min)

- **Responsiveness**. Show the responsiveness of your site. How does it look on a desktop and on a mobile? (0.5 min)
- **Accessibility and usability**. Reflect on the accessibility of your site. Have you paid attention to semantic elements? What techniques have you used? Perhaps show the result of Lighthouse validator test. In case something is missing there, reflect on the reasons. How about Don Norman's principles. Do you have any examples on where these principles are expressed in your user interface? (0.5 – 1min)
- **Extras**. Summarize what extras you have implemented in the project. What brings it up from a C-grade to an A-grade? (0.5 – 1min)
- **Reflection**. What have you learned in the course? (1 min)

Suggested Video Presentation Structure for IDATA2306 Application Development

The first part can be the same as in IDATA2301 Web technologies (you can even reuse the same part of the video, if you work with video editing):

- **Problem introduction**. What task did you get assigned, what is your theme? What kind of shop is this? (0.5 - 1 minute)
- **Work methodology**. How did you use the typical software developer techniques to develop your project (Sprints, version control)? (0.5 – 1min)
- **Architecture** of the solution. Explain the overall architecture – is it monolith or decoupled? On one server or on two different servers? What authentication do you use? What database is used? (0.5 – 1min)
- **Design decisions and agreements.** Have you made any specific design decisions? What have you discussed with the customer (teachers) during the semester? For example, what was decided regarding the search options, or the currency? Clarify these decisions here. (0.5-1min)

The second part is specific for IDATA2306 Application development:

- **Database schema**. Show the table and relations between them. You can generate the schema with different tools. For example, show "ER Diagram" in DBeaver. (1 min)
- **API endpoint documentation**. List all the API endpoints, describe them shortly – the URL, request structure, response structure, return codes. The easiest, probably, is to use Swagger-generated documentation. In case you don't use Swagger, describe the endpoints in another way. Don't use much time by discussing the details of every endpoint, just list them. (0.5-1 min)
- **User roles**. Describe the user roles shortly – what user groups (roles) do you have? What is accessible to each group? (0.5 min)
- **Demo of the functionality**. Walk through the main workflow – from opening the page, to searching products and performing a checkout (placing an order). Which endpoints are involved where? You don't need to spend very much time explaining specific code lines. Source code will be checked by examiners separately. But you can mention the main principles here. Have you separated the code in layers according to best practices? (Service, Repository, Controller). What is responsibility for each layer? You can take some use case, for example, one request. Then walk through that request handling from receiving the request in the controller (perhaps mention security settings as well), to the SQL request in repository. Here you have some room for improvisation. (3-6 min)

- **Tests**. Demonstrate the tests you have. Explain what you are testing, show the tests in action. This should be Postman test collection (or several collections) by default, but you may have also other tests: Mock MVC tests, for example. (1-3 min)
- **Extras**. Summarize what extras you have implemented in the project. What brings it up from a C-grade to an A-grade? (0.5 – 1min)
- **Reflection**. What have you learned in the course? (1 min)

Deadline

The deadline to hand in all the necessary deliverables on Inspera in both courses is TBD, 2024, 12:00 (lunch time, not midnight!)

Grading guideline (for examiners)

This section describes the grading guideline for examiners (including complaint handling). This information is also presented to the students for transparency.

Examiners use the following grading procedure:

1. **Check all the mandatory (fundamental) requirements** of the project. If everything is covered, the project deserves a C-level grade as a starting point. If something is missing, the examiner must comment on that, and the starting grade is reduced accordingly.
    a. In case the project is having significant faults and is missing several important parts of the mandatory requirements (or the hand-in is missing mandatory information which makes the grading impossible), the examiner can decide that the project does not deserve a passing-grade and is hence evaluated with an F (failed).
2. **Check all the extra features** the students have implemented. These must be clearly visible both from the presentation video, from the deployed system on the servers and in the source code. The extra features can increase the grade from 0 to 2 levels. For example, if the starting grade was a C, the final grade can be C, B or A, depending on how highly the examiner evaluates the extra functionality presented by the students. This also means that a project which is missing some mandatory requirements and hence has D or E as a starting grade, can go up only two grades (to B or C respectively). A project which got an F as a starting grade (too many important elements missing) can't "be saved" by extra functionality.

Grading schema

Grading is done in two phases, as described above: Fundamental requirement check and Excellence check.

The fundamental requirements differ among the courses (backend and frontend). Therefore, the schemas differ as well.

The information written in the grading schemas should be presented to the students (the group) as feedback for the grade (*NOR: begrunnelse*).

Grading of fundamental requirements for IDATA2301 Web technologies

Requirement	Evaluation	Comments
Here the examiners list each fundamental requirement as a separate row in the table	In general, each requirement should be evaluated with one of the following categories: • Good – everything is good, no comments here. • OK – some minor drawbacks, but OK in general. • Partial – some significant mistakes (or something missing), but partly OK • Poor – significant mistakes • Missing – the requirement is not satisfied at all. For example, no sprint reports available.	Here the examiners can write comments for each requirement. If a requirement is considered "Good", no comments are necessary. A comment is mandatory if a requirement is evaluated as "OK", "partial", "poor" or "missing"
Design guideline – is the design guideline complete? Do students follow it when implementing the site? Is the style consistently implemented?		
Wireframes – do students have wireframes (or mockups) of the site? How complete are they (Not necessarily advanced and very detailed, but covering all the pages of the site)		
Semantic elements – do students use semantic elements when building the HTML?		
Accessibility, usability, validation – is accessibility considered? Do Lighthouse tests pass?		
Responsiveness – is the website responsive? Does it look OK on both desktop and mobile?		
Interaction with JavaScript – does the site include JavaScript which modifies the DOM?		

REST API calls – is some data loaded from the backend (or external API) using REST API calls?		
Visual theme – does the site look aesthetically OK? Is the general theme of the *customer* followed (serious VS playful, etc.)?		
Video presentation – is the presentation clear? Does it present all the necessary elements? Note: the video does not need to be top-influencer quality, but the group should make effort to make the video perceivable and clear. For example, the sound should be loud and clear enough, texts shown in the video should be readable.		
Version control – is version control used during the semester? Is source code and documentation there?		
Regular work – is the work through the semester documented (using Sprint reports and GIT commits)?		
A complete application – is the application working? Are all necessary steps supported (landing page, product search, selecting products, placing an order?)		
Agile work methodology – are sprints and issues used? Are sprints documented?		
Clean code – does the code have good structure? Is it readable? Does it follow best practices?		
README. Is Readme provided? Does it contain short description of the project? Does it contain "Getting started" - instructions on how to run the project?		

Identified extra features implemented by the group in IDATA2301 Web technologies:

- List the identified features here as bullet points. The features should be presented by the students in the video presentation. The teacher is allowed but not required to identify other features as well (something that the students did not present themselves)

Final grade in IDATA2301 Web technologies: here the examiner writes the final letter grade for the project (A-F). Unless there is a very clear reason to do otherwise, all the members of the group get the same grade.

Comments: here the examiner can write summarizing comments for the project – something that explains why exactly that grade is chosen as the final evaluation.

Grading of fundamental requirements for IDATA2306 Application development

Requirement	Evaluation	Comments
A complete application – is the application working? Are all necessary steps supported (product search, selecting products, placing an order?)	Same rules apply here, see IDATA2301 grading schema	Same rules apply here, see IDATA2301 grading schema
Database schema – is the schema included in the hand-in? Does it describe the entities and relations clearly?		
REST API - endpoints for fetching necessary data, at least one of them with CRUD operations?		
Authentication – login, permission check implemented?		
3+ user roles – visitor, user, admin, with different access rules on different endpoints?		
HTTPS – are both backend and frontend secured?		
Test set – Postman collection (or MockMvc tests), reentrant?		
API documentation – endpoints listed and described?		
Video presentation – is the presentation clear? Does it present all the necessary elements?		
Version control – is version control used during the semester? Is source code and documentation there?		
Regular work – is the work through the semester documented (using Sprint reports and GIT commits)?		

Agile work methodology – are sprints and issues used? Are sprints documented?		
Clean code – does the code have good structure? Is it readable? Does it follow best practices? Are passwords stored in environment file(s)?		
README. Is Readme provided? Does it contain short description of the project? Does it contain "Getting started" - instructions on how to run the project?		

Identified extra features implemented by the group in IDATA2306 Application Development:

- List the identified features here as bullet points. The features should be presented by the students in the video presentation. The examiner is allowed to (but not required to) identify other features as well (something that the students did not present themselves)

Final grade in IDATA2306 Application Development: here the examiner writes the final letter grade for the project (A-F). Unless there is a very clear reason to do otherwise, all the members of the group get the same grade.

Comments: here the examiner can write summarizing comments for the project – something that explains why exactly that grade is chosen as the final evaluation.