Tasks Affecting Grade

Grading

The tasks in the list below shall be solved by all group members together. All group members get the same grade, provided everyone contributes to the development to about the same extent. If different group members want different grades, everyone shall contribute equally to the lowest of the group member's grades, and then only those who want a higher grade solve extra higher grade tasks. In this case, clearly state in both oral and written reports who has done which extra higher grade tasks.

- **Grade E**, Accepted final individual written report (task 37 below) describing the solution to all tasks marked X in the Mandatory column.
- **Grade D**, Accepted final individual written report (task 37 below), submitted no later than March 15, 2026, describing and motivating the solution to all tasks marked X in the Mandatory column.
- **Grade C**
  - Meet requirements for grade D
  - Final individual written report (task 37) describing and motivating solutions to tasks with a total higher grade score of at least 7 (of 26). Scores are given only for tasks presented latest at the final group report (task 36 below), March 10, 2026.
- **Grade B**
  - Meet requirements for grade D
  - Final individual written report (task 37) describing and motivating solutions to tasks with a total higher grade score of at least 13 (of 26). Scores are given only for tasks presented latest at the final group report (task 36 below), March 10, 2026.
- **Grade A**
  - Meet requirements for grade D
  - Final individual written report (task 37) describing and motivating solutions to tasks with a total higher grade score of at least 17 (of 26). Scores are given only for tasks presented latest at the final group report (task 36 below), March 10, 2026.

Task	**Manda-tory**	**Higher Grade Score**
Functionality		
1. The following use cases must be implemented. • Use case 5.2 must be fully implemented, in order to show authorization and authentication. • Use case 5.3 must be fully implemented, in order to show how data entered by the user is persisted. • Use case 5.4 must be fully implemented, to show how data is fetched from the database and presented to the user. You're allowed to implement also use cases 5.1 and 5.5, but that alone will not improve your grade. Some higher grade tasks might, however, require implementing additional use cases, which in that case is clearly specified. Note that you're *not allowed to invent any new functionality*, besides what's mentioned in the use cases, without asking the teacher. New functionality might, however, provided it includes something conceptually new, and not just more code, give extra higher grade points, as described in task 31.	X	
Robustness		
2. Flexible, easily understood code, this *must* be motivated. It *must* also be motivated which patterns, code conventions and frameworks are used.	X	
3. Appropriately **layered architecture**. *Motivate*!	X	
4. Javadoc comments (or corresponding in other programming language) for all public definitions.	X	
Security		
5. Authentication	X	
6. Authorization	X	
7. Password hashing Passwords must be hashed in the database.		1
8. CORS *This task can only be solved if client-side rendering is used.* Use CORS for all requests from frontend to backend, and motivate how CORS is configured.		1
9. Logging Motivate *what* is logged, *how* it's logged and *where* the log is. *There must be logging by the application itself.* Access logs for HTTP requests or logs generated by frameworks are also important, *but not sufficient.* The logging must meet the requirement mentioned in section 4.1 in the document *application-description.pdf.*		1

Task	**Manda-tory**	**Higher Grade Score**
Transactions		
10. Motivate when and how transactions begin and end. To illustrate the need for transactions, consider for example registering an application. This will require saving both availability periods and competences, which requires more than one insert statement in the database. Either all these inserts must be performed, or none of them. Make sure to watch and understand the recorded transaction lecture in Canvas (lecture 9) before you solve this task. **Note that transactions must be used for all use cases.**	X	
11. Implement **alternative flow 2a in use case 5.5**, *Show Application*. The scenario is that two users list the same application simultaneously, then user one updates the status (for example to *accepted*), and finally user two also updates the status. Now user two will overwrite user one's update, without even knowing that user one made an update. In this case, the application must instead abort the update and inform user two why it was aborted.		1
Error Handling		
12. You're required to **handle all kinds of errors**. *Always* present an appropriate error message to the user. Describe where different errors are handled; what is presented to the user, and how; and, if you implement logging, whether and how exceptions are logged. **Remember to handle exceptions caused by malfunctioning**. You can test this by for example making the database unreachable, or hardcoding an exception somewhere in your code.	X	
Internationalization and Localization		
13. Internationalization and localization of web pages. Displaying the UI in a new language shall only require adding text in that language, no coding shall be required.		1
14. Internationalization and localization of database. It shall be possible to add a new language without changing database schema. The only thing affected by this task in the existing database is the competence names.		1

Task	**Manda-tory**	**Higher Grade Score**
Process		
15. Frequent integrations Code developed by different group members should *often* be tested together and integrated in the group's shared repository. **The commit history must show that all group members have contributed more or less equally to the codebase**.	X	
16. Use a **build tool**, like Maven. Ant isn't accepted.	X	
17. A **CI/CD pipeline** is used for unit testing, static analysing, and deployment. Note that it must be used for all those three purposes. It must also be possible to evaluate the results of tests and static analysers. You are encouraged to implement this pipeline early in the project, to benefit from it, but you are allowed to implement it at any time.		1
Handover		
18. The **application is live on a cloud platform**, for instance Heroku. Note that this requirement is mentioned in section 4.3 in *application-description.pdf.*	X	
19. The **source code is available in a public Git repository**, for handover to other developers, who did not take part in the development. *You must write sufficient documentation to make it possible for those to continue development and deploy new versions.* Note that this requirement is mentioned in section 4.3 in *application-description.pdf.* **The commit history must show that all group members have contributed more or less equally to the codebase**.	X	
Testing		
20. Backend unit testing, possible test frameworks for Spring are for example *Spring TestContext* (in-container) or *org.springframework.mock.env* (mocked container), and for Node.js you may choose for example *Jest. The tests must be extensive, and cover all layers in the backend.*		2
21. Frontend unit testing, *the tests must be extensive, and cover all layers in the frontend.*		2
22. Acceptance testing (end-user testing), using for example *Selenium.* The tests must access the UI, like a user, and assert that execution is correct; a recorded interaction with the application isn't enough. *The tests must be extensive, it's not enough just to get them going.*		2

Task	**Manda-tory**	**Higher Grade Score**
Requirements on the View		
23. All server-side functionality used in solutions to tasks in this list **must have a GUI**. Even if you choose to develop a REST API and use client-side rendering, you still have to implement a GUI.	X	
24. All web pages must work well in all browsers specified in section 4.2 in application-description.pdf. Apart from this, it doesn't matter how simple or advanced the view is.	X	
25. Server-side validation of data entered in HTML forms. The validation must be done in the topmost server layer.		1
26. Client-side validation of data entered in HTML forms.		1
27. Views consist of different parts, e.g. header, footer, menu and main content. Neither the fragments (header, footer, etc), nor the layout may be duplicated. All those parts must be reused by all views.		1
Persistence		
28. Data Migration This task refers to section three in *application-description.pdf.* To **pass**, all data in the existing database must exist in your new database. You must also explain how you handle lacking data, for example a user without password, but you don't have to implement that. For the **higher grade** point, you must implement handling of lacking data. All existing data is included in the existing database, you do *not* have to imagine any kind of lacking data that is not present in the existing database. There is a SQL script on the *Project* page in Canvas that generates the existing database.	X	1
29. Validation in the integration layer on the server, with the purpose to verify that all data sent to the database has correct format.		1

Task	**Manda-tory**	**Higher Grade Score**
Other Non-Functional Requirements		
30. Meet non-functional requirements from the following list, explanations are found in lecture two. **You get one point for each of the four requirements below that is met.** • Response time • Capacity • Scalability • Non-repudiation To get this point(s) you must clearly state what your requirement is, and also present a measurement and an analysis of the measurement which together show that you have reached the requirement. Discuss with the teacher how you intend to show that you have met a requirement.		1-4
Your Own Choice		
31. You can define another requirement, not mentioned in this list. You must discuss the requirement with the teacher, and agree on how many higher grade points it's worth.		
Reporting		
32. Add at least three resources to this course's Canvas page *Links*. The resources must have been of help to the group's development. Also, they must be unique, you're not allowed to add a resource that's already on the *Links* page. The max score is one, you won't get two points for six resources.		1
33. Since this is a course concerning architectural decisions, **you're required to maintain a log of all architectural decisions** you take. This log shall be added as an appendix to the final report. The log is per group, not individual, it's expected to be identical for all group members. You might want to use the GitHub wiki for this log (if you're using GitHub for your repository), but that's not a requirement.	X	

Task	Manda-tory	Higher Grade Score
Reporting, Cont'd		
34. Individual Written Report All group members shall submit an individual report *before* each of the four group report meetings described in tasks 35-38. These reports are submitted by answering quizzes in Canvas. Note that these quizzes aren't tests, they have questions where you write free text answers describing your project work. **One of the questions is to specify how much time you have spent on the course. That means you must keep track of your work hours.**	X	
35. First Group Report No later than January 27, 2026 (book reporting time in Canvas). You shall present your architectural decisions regarding project organization, how to use git, and which frameworks and layers you are going to use in both frontend and backend. **The log of architectural decisions (task 33) must be in order.** No code is required. **The higher grade point is given if the report is accepted before due date.**	X	1
36. Second Group Report No later than February 6, 2026 (book reporting time in Canvas). You shall have code connecting all layers. An HTTP request shall result in execution of presentation layer, business logic and database. It shall be production quality code that can be left in the final product. You shall have a clear picture of what is done and what problems there are. The log of architectural decisions (task 33) must be in order. **The higher grade point is given if the report is accepted before due date.**	X	1
37. Third Group Report No later than February 20, 2026 (book reporting time in Canvas). You shall have substantially more code than at the second report (task 36). It shall be production quality code that can be left in the final product. You shall have a clear picture of what is done and what problems there are. The log of architectural decisions (task 33) must be in order. **The higher grade point is given if the report is accepted before due date.**	X	1

Task	**Manda-tory**	**Higher Grade Score**
Reporting, Cont'd		
38. Final Group Report Performed no later than March 10, 2026 (book reporting time in Canvas). The group gives a detailed report to the supervisor. The log of architectural decisions (task 33) must be in order. **The higher grade point is given if the report is accepted before due date.**	X	1
39. Final Individual Written Report Submitted in Canvas no later than March 15, 2026. The report must follow the template provided in Canvas. *The report is individual, you must not write it together with other students.*	X	