# Exam for PG6301 Web Development and API design, 2021

This is a practical exam. This mean you will be evaluated on what working software you are able to produce and deliver, rather than what theoretical knowledge you have obtained.

The knowledge goals are front-end development, specifically with React and API development, specifically with Express. In the exam, you are asked to demonstrate that you are able to use these technologies to create a working web application with an API backend.

Specifically, you will be evaluated on the following:

- Does the program you deliver run "out of the box"? The examiner will run `npm install && npm test && npm start` on your project - this should start the backend and frontend with no issues
- Does the program you deliver contain the required functionality? The examiner will follow the instructions of the assignment or your readme-file and try to use the application
- Is the program structured and created in a sensible way? Is the frontend structured into meaningful components in a file structure that is easy to understand?
- Does the code follow good programming practices? Specifically, do you use appropriate technologies and naming conventions and does your code have good coverage?
- Are design choices, functionality and shortcomings explained well in the README.md-file?

Grading guideline:

- **Passing (E)**: `npm start` runs a functioning React application with components you can navigate between; there are some unit tests run with `npm test`
- **D**: The React application communicates with an Express API with both fetching (GET) and mutating (PUT, POST or DELETE) requests
- **C**: A substantial subset of the functional requirements are fulfilled; there is decent test coverage (> 50%). There should be some layout with CSS grid or flexbox
- **B**: (Almost) all of the functional requirements are fulfilled; there is good test coverage (> 60%); code is structured in a thoughtful way. There should be a structured (but not necessarily polished layout) for the pages
- **A**: The candidate provides non-required features such as logon with Active Directory or Google, Websockets or Typescript; test coverage > 70%

The delivery can use React functional components or class components. It can build with Webpack, Parcel or another build tool, as long as `npm start` works. There should be some CSS, but there are no requirements to how this is used. The code can be in JavaScript or Typescript. You can use npm or yarn as you prefer, and you can structure your code as you see fit, but be aware that conventions are there for a reason. You can have the express and React app in the same module (`package.json`-file) or in separate modules as you see fit. Just make sure that `npm start` works.

## TL;DR:

The behavior of your code and code coverage when the examiner runs `npm install && npm test && npm start` will be a substantial part of the evaluation. Make sure you test this many times as a program that doesn't run severely impacts your grade.

Some things to watch out for:

- Generated files (`node_modules`, `.parcel-cache`, `dist` and `coverage`) should not be part of the delivery
- Deliver a zip-file (not RAR!). The best way is to develop on Github and select "Code" > "Download ZIP" on your repo at github.com
- Make sure you fulfill the spirit of the functional requirements

## Cooperation during the examination

You are allowed to help and get help from other students, but you must mention any code you share in the README.md-file or it will be considered plagiarizing. You will be evaluated on your own code that you deliver and should disclose any code that is not your own.

# Functional requirements: Messaging system

You task is to create a messaging system where users can send messages to each other. You can use the messaging functionality in Canvas for inspiration. To qualify for a passing grade, you must be able to store messages on the Express server in memory and retrieve them. For C and higher, users must log in to see their own messages. For B and A, you should implement web sockets to notify users that they have received a new message.

1. A logged in use should be able to register more users in the system
    - Users should have properties first name, last name and email address
    - Optionally, users can have description and picture
2. A logged-in user should be able to create messages that are sent to one or more users
3. Users should be able to see messages where they are a recipient or sender
4. Users should be able to respond to messages

The system data should be stored as variable in the Express server and it's okay that the data disappears every time the application is restarted. You can not assume that the examiner has any specific database technology available to test the functionality.

For login, you can use hardcoded usernames and passwords, allow self registration or support the use of Google or Active Directory. You can assume that the examiner has access to two different Google accounts and two different Active Directory accounts. You have to use a localhost address for the redirect URI and you should supply a client id and client secret (if you are authenticating in the backend).