ADVANCED SERVER WEB:
A MULTIPLAYER MINESWEEPER

We will be building a multiplayer minesweeper using NodeJS, express, EJS, socket.io, JQuery, and some good old HTML+CSS. If you do not know what *multiplayer* or *minesweeper* are then please ask your friends, they will explain it better than any document can. All the other words are technologies studied in this course (or in previous courses).
The goal of this exercise is to ***use all*** *those technologies in one big project.*

MULTIPLAYER MINESWEEPER

The difference between classic minesweeper and multiplayer minesweeper is that there are more than one player. But there is more: each player has a colour, and when new land is discovered (the classic minesweeper gameplay) then the new land is *owned* by the player who discovered it first. The winner, if any, is the player that after the complete area has been discovered has the most land. Players that cause a mine to explode are immediately excluded from the game.

TECHNOLOGIES

Players have to register and login before they can play. The login data has to be stored in a database. A summary of each played game is stored on file (define your own format, maybe even keep a replay). The game is to be played through a webinterface. All HTML, CSS, Javascript, ... has to be served using an express-application. Use the more advanced features such as EJS where applicable. Test your code. NodeJS and JS have a lot of unit-test-libraries. Pick one of these libraries, try it. A multiplayer game is almost by definition asynchronous. This means you will be using socket.io for the communication between client and server (e.g., player's moves and game updates).

TEAM WORK

Client-server communication in socket.io happens trough messages. Pair up (this means at least per two), and discuss *before you start coding* what these messages will be and what data they send over the network. The idea is that if Alice writes ``Alice’s client`` and ``Alice’s server`` and Bob writes ``Bobs client`` and ``Bobs server``, then ``Alice’s client`` and ``Bobs server`` can also work together and ``Bobs client`` and ``Alice’s server`` can also work together*. Ideally you all clients can connect to al all servers and vice versa*, but maybe it is a bit too cumbersome to organise the discussion about the protocol in an orderly fashion.

CHECKLIST

☐ LIST OF COMMANDS/EVENTS IN THE CLIENT-SERVER COMMUNICATION

☐ NodeJS

☐ using methods with callbacks

☐ writing methods with callbacks

☐ Databank

☐ File

☐ HTTP-server and Express (sessions, …)

☐ EJS

☐ socket.io

☐ Javascript test library