Instructions

Overview

Date out: 14/11/2025

Date and time due: **14/12/25 at 23:59**

Value: **25%** of total module marks

Assignment format: student works continually, pushing their work to GitHub; a demo is video-recorded at the end

Submission format: a zip archive containing the repository (only the final version), uploaded in Brightspace

Graded: continuity of work, progress recording, video demonstration, functionality, code quality and style, user experience

Use of GenAI: not allowed for any portion of this assignment

Oral examination: some students may be called to an oral exam after submission

General format

This assignment requires you to build a client-side web application, with requirements given in stages that correspond to the remaining module topics (in week 10 on 19/11/25, week 11 on 26/11/25 and week 12 on 03/12/25). This will allow you to simultaneously work on the CA and towards achieving the practical learning outcomes of the remainder of the module. There will be no separate lab work.

Grading

The following elements are graded:

Continuity of work [20%]
You must work on your application and push your work to GitHub continually, ideally completing each of the four stages of the assignment before the next one is given out. The commit trail found in your repository must provide evidence of your work sessions (the assignment is such that you are not expected to complete any of the stages in one sitting, which means that there would typically be many more than four commits for this assignment). There will be no lab assignments in the second part of the semester, but you should 'try out' things covered in the lectures before applying them in the assignment. A trail of such activity in your GitHub repository (e.g. in directories week9, week10 etc.) would also be positive evidence of continual work.
Progress recording [20%]
It is important that all git commit messages contain authentic information about your progress. These can be multi-line messages that describe problems and how they were solved, tough concepts and how they were mastered, design or implementation decisions that were made. The descriptions must be specific, with references to file and construct names.
Video demonstration [20%]
Once you complete the work, you must record a screencast video with voiceover, up to 10 minutes long (any content beyond the 10th minute in a longer video will be ignored i.e. not viewed for grading), that explains the puprpose of your application and demonstrates all its use

cases, including any error handling. The video should also show the execution of a successful accessibility audit on your application and any completed testing. Finally, it should include a brief overview of three problems that you encountered while completing the assignment and how you solved them, including any relevant code. You might want to take screenshots of such problems as they manifested in your application, so as to be able to show them in the video at the end.
Functionality, code quality and style, user experience [40%]
In this section, the submission will be graded on

- whether the required functionality is present and working
- code quality in terms of structure, comments, construct naming, generalisation, reuse
- user experience (UX)

Functional and other requirements

Stage 1 (published: 14/11/25, to be done by around: 01/12/25)

- Create a sub-directory in your RWAT git repository called 'ca2' and keep all the work for this assignment inside it.
- Download and unzip into the *ca2* directory the starting code for this assingment, provided alongise these instructions. The code contains a custom element. You should examine it briefly and its interface, focusing mostly on the usage examples at the bottom of the JavaScript and HTML files.
- Your task in this part of the assignment is to write a custom element of your own that uses the flippable card elements in a memory card game. You must not modify the shape-card element in the process.
- Your element should take the dimensions of the 'board' as its only parameter in the format 'rows x columns' e.g. '3 x 4'.
- This video shows how the game should work when finished.

- Your final files should not contain the examples. The HTML file should include a single instance of your game element, apart from templates.

Stage 2 (published: 19/11/25, to be done by around 01/12/25)

- Turn your CA2 work into an npm project by running npm init in the *ca2* subdirectory in your RWAT GitHub repository.
- Install *vite* in the CA2 directory and make sure that you can use it to run your code.

Stage 3 (published: 26/11/25, to be done by around 08/12/25)

- Design and create a Firestore collection of documents to keep time-stamped game results, each consisting of the number of clicks it took to complete the game.
- Add code to your application that adds a document to the Firestore collection every time a game is completed.
- Add a button to your application that allows the user to display the average number of clicks to game completion. This should be calculated from the data found in Firestore.

Getting more information

- Treat the lecturer as your customer. Ask questions if the requirements are not clear or if you think that there is information missing.
- However, please do not send emails for this purpose. Come to the lectures and labs to ask any questions you have.

Attachments
ca2_start_files.zip
(3.15 KB)