University of St Andrews School of Computer Science

CS5063 — Web Technologies — 2025/26

Assignment: P1 - GameShelf
Deadline: 2025-10-31 21:00:00
Credits: 30% of module grade

MMS is the definitive source for deadline and credit details

You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

1 Description

You have been hired by GameShelf, a company that builds bespoke single-page storefronts for independent board-game shops. Each shop curates its own catalogue (e.g., party games, heavy strategy, family titles), sets prices, and manages customer baskets.

Your task is to build an interactive single-page web application that simulates a board-game shop's online checkout experience. Users should be able to browse the shop's catalogue, view game details, configure options (such as, edition, add expansions), add items to a basket, and place a mock order.

You will be required to use the following technologies:

- HTML5 for the structure of the web page,
- CSS3 for presentation of the web page,
- JavaScript for client-side programming to handle interaction and functionality on the web page.

The coursework is divided into three stages: Primary, Secondary, Tertiary requirements. You are encouraged to complete them in order, ensuring that each stage is working before moving on to the next. The tasks build on each other and should be submitted as a single project.

You are not allowed to use 3rd party libraries, frameworks, or preprocessors (such as Vue, React, SCSS, or Bootstrap).

The suggested structure is that there is an index.html, a style.css, and one or more JavaScript files. However, you should feel free to diverge from this as you see fit and can justify in your report.

If you don't want to come up with your own boardgame shop and list of games, the catalogue for an example shop BigBoardBoutique has is provided in the file catalogue.js. This can be used as the basis of your submission.

2 Data Model

Each catalogue item should be represented as an object with the following properties:

- Name: The name of the game, a string up to 64 characters long.
- Price: The price of the game, a positive number representing the number pence (e.g. 1500 for £15.00).
- Genre: The type of game (e.g. Social deduction, deck-building, family), a string up to 32 characters long.
- Description: A short description of the game, up to 256 characters long.
- Options: An optional array of configuration options (e.g. expansions, editions, special dice) with additional prices.

You are free to expand on this model based on your shop's theme (e.g. adding an image of each game), but all entries in the catalogue must conform to the constraints listed above. You do not need to persist data between page reloads. The file catalogue.js contains a sample catalogue that you can use to test your application and as a basis for your catalogue. Because it is a JavaScript file, it can be included in your project with a script tag.

3 Requirements

3.1 Primary Requirements

Design and implement a basic web application using HTML, CSS, and JavaScript that:

- Displays the shops catalogue with the name, price, category, and description of each item.

- Allows the user to add items to their order. The order should be displayed in a separate list with the total cost.
- Provides a way to remove items from the order.
- In a seperate CSS file, add CSS rules to style the catalogue and the order. Make sure to set the font, background, and text color at a minimum.

3.2 Secondary Requirements

Build on the basic functionality by adding:

- Options: Allow users to select the various options for each game when adding an item to their order. The total cost should update based on the selected options.
- Discount Codes: Implement discount codes (e.g., “GAMESNOW” for 10% off). The user should be able to enter a discount code, and the total cost should be updated accordingly. You can come up with your own discount codes and discount rules but they must be documented in your report and the system must reject invalid codes.
- Use CSS to ensure that the layout is responsive, working well on both desktop and mobile devices.

3.3 Tertiary Requirements

Extend the system by implementing:

- Sorting: Implement sorting for the catalogue, allowing users to sort by category or price (ascending/descending).
- Search: Allow the user to search for catalogue items by name or category and filter the displayed catalogue based on the search results.

4 Report

The report should include the following sections:

- Overview: A brief description of the shop and the catalogue you have chosen to implement. This section must state which of the requirements you have completed and to what extent.

- Design: Description of your design of your web application and the intent behind the choices that you made when designing and implementing it.
- Testing: A description of how you tested your web application, including any issues you encountered and how you resolved them. You should include screenshots of your application in action and any error messages you encountered in the console.
- Evaluation: This section should include one or two paragraphs evaluating the success of your submission compared to what you were asked to do.
- Conclusion: This section should include one or two paragraphs addressing what you have achieved, what you found difficult, and what you would have done if you had more time.

Word Limit: The report should be no longer than 2,000 words. The word limit for this assignment is advisory, and excludes references and appendices. No automatic penalties will be applied based on report length, but your mark may still be affected if the report is short and lacking in detail or long and lacking focus or clarity of expression. A word count must be provided at the start of the report.

5 Hints

- We recommend an incremental approach: make sure that each part of your program works before moving on to the next part. Make a safe copy of whatever you have got working so far so that you always have a working solution to submit while you are developing the next part of your solution. This should allow you to experiment with different ways of organising your code and implementing functionality without worrying of breaking things.
- Use an HTML validator and a CSS validator to validate your code.
- Use strict mode for JavaScript and check for errors/warnings in the JavaScript console of your web browser.
- It is good practice to write your HTML code, CSS code, and JavaScript code in separate files and link to the CSS and JavaScript files from within the HTML file as appropriate.
- Use the Live Server plugin for Visual Studio Code (or something similar for your editor or IDE of choice) to run your web application when developing it.

- Any requirement requiring CSS is designed to assess your ability to code using CSS rather than your graphic or interactive design skills. You should aim for a clean, simple, and consistent design but we will be grading the CSS based on the quality of the code rather than the visual appearance.

Submission

A single file containing your code and PDF report in ZIP format must be submitted electronically via MMS by the deadline. Submissions in any other format will be rejected.

The specific mark descriptors for this piece of coursework are described below.

Marking

Submissions will be marked in accordance with the School's General Mark Descriptors, which can be found in the student handbook at: http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

The following examples illustrate the typical characteristics of submissions that would fall into each mark band.

- **1–3:** Little or no working code, and little or no understanding of web technologies demonstrated.
- **4–6:** Some attempt at implementing basic functionality, such as displaying part of the catalogue or handling a user action, but the code does not run reliably or fails to meet the Primary requirements. The report shows limited understanding of the technologies used.
- **7–10:** A partially working web page that demonstrates some understanding of the main concepts but is missing major functionality from the Primary requirements. Code may be inconsistent or poorly organised, and testing is minimal or absent. The report is brief and lacks technical depth.
- **11–13:** A working implementation that successfully fulfils the Primary requirements. The report demonstrates reasonable understanding of the implementation and some evidence of testing.
- **14–16:** A complete and well-structured implementation of the Primary and Secondary requirements. Functionality is correct with only minor issues or missing features. The report includes thoughtful discussion of de-

sign decisions and systematic testing with evidence such as screenshots and validator output.

- **17–18:** A complete and well-structured implementation of the Primary and Secondary requirements, with significant progress on the Tertiary requirements. Code demonstrates excellent quality. The report is well structured and written and includes thoughtful discussion of design decisions and systematic testing with evidence such as screenshots and validator output.
- **19–20:** All required functionaity implemented correctly. The work shows exceptional design and implementation, with comprehensive documented testing. The report is extremely well structured and written and includes thoughtful discussion of design decisions and systematic testing with evidence such as screenshots and validator output.

Policies and Guidelines

Marking
See the standard mark descriptors in the School Student Handbook: https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Lateness penalty
The standard penalty for late submission applies (Scheme A: 1 mark per 24-hour period, or part thereof): https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good academic practice
The University policy on Good Academic Practice applies: https://www.st-andrews.ac.uk/education/handbook/good-academic-practice/