CP1295 — Mini-Project (20%): "QuickNotes" – Browser-Based Sticky-Note Board

Individual Project – Due August 10, 2025 at 11:59 p.m. (D2L submission)

Students must be doing this individually.

1. Purpose

QuickNotes is a compact single-page web app that lets users create, move, edit, and delete virtual sticky notes directly in the browser. The workload is sized for 20% of the course grade while exercising every major topic in CP1295:

- DOM manipulation & events
- Custom objects & classes
- Browser storage & DevTools
- Asynchronous JavaScript (fetch, async/await)
- ES-module organisation
- JSON parsing & generation
- Core Git / GitHub practices

2. Learning Outcomes Demonstrated

- Build and manipulate DOM elements with JavaScript events.
- Encapsulate state in a custom class (Note).
- Persist client-side data with localStorage and inspect it in DevTools.
- Write and consume asynchronous code that fetches external data.
- Structure a small codebase with ES modules (import/export).
- Serialize and deserialize data to/from JSON.
- Apply disciplined version control in a public GitHub repository.

3. Mandatory Features (Minimum Viable Product)

#	Feature	Specification
1	Add note	Typing text (or double-clicking the board) creates a coloured note at the cursor.
2	Edit note	Click note inline textarea or modal; blur / Esc saves.
3	Drag & drop	Click-drag moves a note; position persists.
4	Delete note	The “ ” button removes it with a fade-out.
5	Persistence	Notes (text, colour, x, y) auto-save to localStorage; restored on refresh.
6	Random-quote enhancer	The “ ” icon on a note fetches a productivity quote from an open API (async/await, error-handled).
7	ES-module layout	Minimum files: main.js, ui.js, notes.js, storage.js.
8	JSON export	The “Export” button downloads my-notes.json, containing all notes.

4. GitHub Workflow Requirement

- Create a public repository before writing any code. Begin with a clear and concise README that clearly states the project's goal.
- Use at least one feature branch (e.g., feature/drag) and merge through Pull Requests.
- Record 8 meaningful commits (no single “final” dump).
- Comment on each PR and tag the final commit v1.0.0 before submission.
- A weak commit history will cap your GitHub workflow credit.

5. D2L Submission Requirements

- Working GitHub link — the tagged release URL (https://github.com/…/releases/tag/v1.0.0).
- Demo video (at least five minutes) — screen recording with your narration that shows:
- User workflow (add, edit, drag, delete, export, quote fetch)
- Overall code structure (modules, class, storage).

- No additional written report is required.

6. Marking Rubric (20%)

Category	Criteria	Marks
Functionality	All eight MVP features operate as specified	10
Code quality	Clear naming, modular structure, comments	4
Demo video	At least five minutes; audible narration; shows UI flow and key code; clear resolution	6

7. Recommended Timeline

Week	Milestone
1	Repo + README; basic HTML/CSS layout; add-note feature; start drag-and-drop
2	Complete drag-and-drop & delete; implement Note class; localStorage save/load
3	Refactor to ES modules; quote fetch with async/await; error handling
4	JSON export; UI polish; record demo video; tag release; submit on D2L

8. Technical Constraints & Advice

- Plain HTML, CSS, and JS only (no frameworks).
- Target evergreen browsers (Chrome, Edge, Firefox).
- Keep styles light; dark mode optional.
- Follow the Murach JavaScript style guide.
- Cite any significant external code inspirations in the README.

9. Academic Integrity

Discuss ideas freely, but write your code. Uncredited copying will trigger disciplinary procedures at the college level.