Project Grading Criteria

The purpose of this project is to demonstrate what you've learned in this course. You're not expected to implement everything perfectly, but you're expected to try and cover all bases. In other words, the aim of the project is for you to try many different technologies, and then you can master the specific tools you like later on.

Milestones

Project idea and team formation (5%)

The deadline to select the project idea and team members is by the end of Week 3. Write this information in the Google Sheet found on Moodle, and make sure you do not repeat the same idea taken by another team.

The project ideas must be approved by the course instructors.

Intermediary review (5%)

During the labs of Week 6 (lecture 8), you will demonstrate what you've done so far and receive feedback and code review.

Final submission (90%)

The final project presentation will be on the "final exam" date determined by the DoE. For the demonstration, you need to fulfill the criteria below.
You do not need PowerPoint slides, just show how you fulfilled the grading criteria.

Requirements

The app should have some kind of "items" that you manage, like a simple content management system (CMS):

- The user should be able to create, read/view, update/edit, and delete items from a database.

- “item” examples: blog post, restaurant menu, portfolio project, shop product, …
- Each item in the CMS could be accessed by a unique URL path. e.g., example.com/post/123

To meet most of these requirements, you will need to use a modern framework such as SvelteKit or Next.js.

Note: if you have an idea for a project that does not exactly meet these criteria but is still sophisticated enough in a different way, contact the course instructors for special approval.

Functionality (/35)

[15] At least these four pages exist:

[1] Homepage (static content)

[1] About page (static content)

[5] Items collection list page (from CMS), e.g., list of blog posts

[8] Single item details page (from CMS), e.g., a single blog post

- Must use a dynamic URL **path** parameter

[10] CMS items’ forms (to create new items) have appropriate controls and validation

At least 3 different types of input controls

Invalid input should not be submitted to the backend

Error messages are descriptive (relate to the mistake done by the user).

[10] Use proper tools setup

[5] Must use TypeScript, and follow best practices. e.g., not using `any`

[2] ESLint (should not conflict with Prettier when it comes to formatting rules)

- Must be configured to read Svelte files properly (if you’re using Svelte)

[2] Prettier (or ESLint formatting rules)

[1] Valid and semantic HTML (as per W3 Validator - some warnings are fine, but errors should be fixed)

Deployment (/10)

[5] Deploy the project on Cloudflare Pages, Vercel, Netlify, or any other platform that supports SSR (or edge rendering):

- Must be able to fulfill the SEO and dynamic path requirements
- Should have Continuous Deployment (CD)

[5] Continuous Integration (CI) via GitHub Actions. With linting and format checking at the minimum.

Editor Config (/5)

[2] Prettier and ESLint installed
- In other words, defined in `.vscode/extensions.json`

[2] Editor configuration (`.vscode/settings.json`) should include:
- *Format on save* enabled.
- Default formatter configured to Prettier
- ESLint configured to auto-fix all auto-fixable problems

[1] The editor configuration should be version controlled and used by all team members.
- i.e., the `.vscode` folder is not gitignored

If not all team members use VS Code, then the equivalent EditorConfig settings should be used.

SEO (/10)

[5] CMS item pages should have unique and descriptive titles and (meta) descriptions.

[5] Use Server-side Rendering (SSR) to serve the dynamic pages HTML.

CSS (/15)

[5] Aesthetics

[10] Responsiveness (mobile-friendliness)

Bonuses (+12)

Bonuses should be demonstrated during the presentation to get their points.

[3] Feature Sliced Design (FSD)

[4] Server-side validation:

    - **[2]** Authorization: Integrating the CRUD security rules with user authentication. e.g., Firebase security rules.
    - **[2]** Validating the data on the backend (security rules). e.g., Firebase security rules.
- **[5]** Testing (unit and/or integration), with a step in your CI pipeline.
    - Alternative: Husky pre-commit hooks with lint-staged

Team Size

2–5 members per team.

Each member must demonstrate their most significant contributions during the final presentation. This is to encourage a balanced distribution of work. Large differences in contribution would decrease your grade.

Pre-approved Project Ideas

These projects are pre-approved and you can start working on them right away. For custom ideas, you must verify with the course instructors first. A maximum of one team can claim each project.

You must have the same functionality and screens implemented in these template apps. You can clone these templates and inspect their screens (views) and data tables and fields.

1. CRM: A simple Customer Relationship Management app for managing contacts, deals and interactions.
2. IT Ticketing: A mobile app that powers an IT ticketing & issue tracking system.
3. Workplace Safety: Identify and resolve workplace safety issues.
4. Inventory Management: Manage inventory levels, stock value and vendor details.
5. Assignments: Manage Class Schedule & Assignments with this App.
6. Shift Scheduling: Schedule shifts and manage timesheets for frontline workers.

Other AppSheet templates (and templates from other sites as well) may also be fine, just confirm with the course instructors first.

Note: the backend is not a graded part of this course.
You may use simple solutions such as: Firebase, Supabase, or even Google Sheets.