Project

The rules:

- Can be done in groups of 4 students (2 minimum).
- Use Vue.js, Angular or React. You are free to use any library you want.
- Use API over HTTP or HTTPS (a public API, a mock server or your own backend).
- Build various views (pages, modals…). At least 5 pages with valuable logic.
- Set up various routes using vue-router or such a lib (ex. /products/:productId).
- Your project deposit will be on Github : Your git repository is your main delivery. Just share read-only access with your teacher if the repository is private (make it available to your teacher until the end of the year). The code must be available at least 2 hours before the oral presentation, which occurs during the last class session.
- You should follow this git workflow. The minimum:
  - master branch is for production code only
  - develop branch is for WIP
- I will check who commit, and when. #NeverForget
- You should at least comment tricky parts of your code, but using JSDoc comments project-wide is strongly encouraged.
- Document installation steps in README.md.

Graduation:

Criteria	**Points**
Git and workflow	2 points
User experience (UX), user interface (UI) *Graded based on your oral presentation.*	5 points
Code quality, maintenability, bugs, challenges… *Graded based on the source code on git master/main branch*	13 points

Bonus:

- Write a fully featured backend (higher bonus if built on node.js). Adding a backend does not reduce my expectation on the frontend part.
- Use high quality JSdoc comments, with tsconfig.json setup to check js all files.
- Host your static website online (ex. Firebase Hosting, render.com, aws S3…).

Project Example

- **Budget management**: An application such as Budget Cool that allows users to create and manage budgets that are updated automatically after each expense entry. Users can also define and create spending categories to better understand where their money is going and see statistics by period.

- **A train ticket office**: Using public API, it may find routes between 2 train stations at some date and time. Then, the customer can book required tickets.

- **Facial recognition**: Create a system that can detect and recognize one or more faces appearing on a camera video stream in real time by comparing them with faces it already knows, and display the person's name on the capture.

- **A clearing booking app**. Allows customers to find available cleaners near their home at some date and time. Then, the customer picks a proposal, fills its address…