Project
Putting it all together

CSCI 4131: Internet Programming (Fall 2022)

1 Essential Information

- **Deadline** Monday December 19th at 11:55PM.
  - Yes, this is in the finals week. I am sorry about that, this is an unfortunate necessity of the unique circumstances of the semester. As we do not have a final exam for our class the time you might otherwise have spent studying for our final exam can go towards this project.
  - We can afford to be a little flexible on this if needed in response to personal finals schedule. If you need flexibility on this assignment please email course staff and explain your circumstances (why an extension might be needed) and the deadline you are seeking.
- **Size** This project is meant to be a bit shorter than your average Homework assignment, but more open-ended in design, and more comprehensive in nature.
- **Individual Assignment** This is an individual assignment. You should not share code with any other student. Likewise, please do not search online for tutorials on how to build todo lists. The goal is for you to solve this problem independently – there will be no value to you in following a tutorial.

2 Change Log

- None yet.

3 FAQS

- None yet.

4 Introduction

Throughout this semester we've been making a series of relatively focused studies of the core languages and technologies of internet programming, HTML, CSS, JavaScript, the DOM, the HTTP protocol (both at a concept level, and at a more specific low-level), As well as common back-end technologies used in the modern era to build websites: routing, templatting, and SQL databases. The purpose of this assignment is to bring all these ideas together and build your first genuine website. In particular, you will build a relatively simple single-user TODO list management application.

5 Requirements

This assignment is, by intent, unlike our previous assignments. I will not be giving you specific and detailed requirements. Instead, I will list the requirements more broadly. This is by intent – part of this assignment is about YOU figuring out how to piece together the parts you have learned over the semester to create a final working whole.

5.1 Technical requirements

From a technical standpoint, we have the following requirements:

- Your website must use the Express framework for it's server
- Your website's server must be able to run on CSELabs machines
- Your website should store data persistently in your allocated mysql database. We should be able to restart the server (in a completely new machine) and not lose any uploaded data.
- All files needed to run your todo list should be submitted in your final submission. We will not dictate a folder structure.
- You **MUST** include a file "readme.txt" in plain text format, or "readme.md" in markdown format which explains how to launch your software, and any information we need to test it. Include any important URLs we will need. This file should be easy to find upon opening the zip file you submit.
- Your website must not use any third-party CSS or JS libraries without prior authorization
- Your back-end server must not use any nodeJS libraries (other than those discussed previously in lecture) without prior permission.

5.2 Required behaviors

Your server should implement basic todo list software:

- Existing todo list items can be viewed
- Adding todo items
- Deleting todo items
- Marking a todo item as "done" (note – this is separate from deleting) or not-done.
- Filtering todo items by completion status (show only done, show only not-done)

You are not required to implement a user-authentication or user-authorization scheme – that is to say, it is OK if any user, without logging in, can access and modify all todo list items.

It is OK if adding a todo list item causes the browser page to reload, however making an item as "done" or "not done" should not cause the browser page to reload. Those actions must have their data stored using asynchronous web requests.

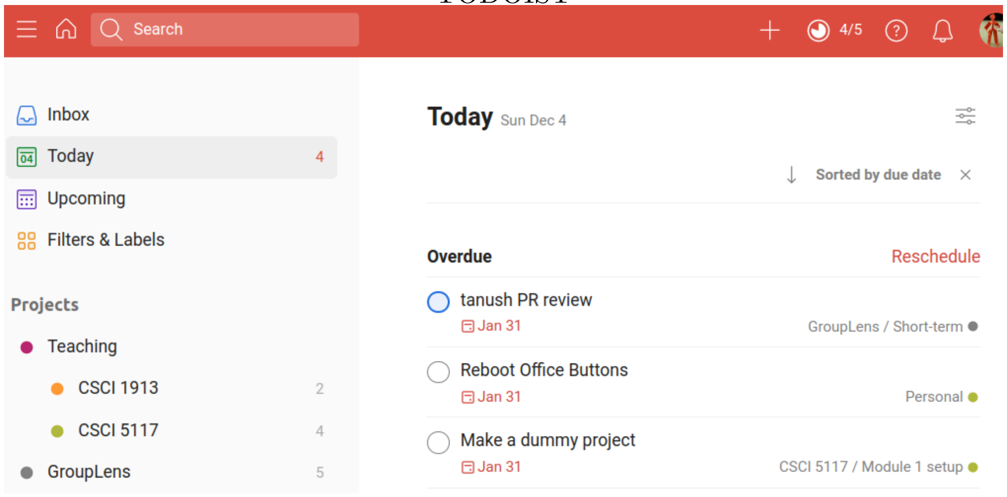
5.3 Visual Requirements

There are MANY possible designs and layouts for a todo list. You are being given freedom here. We do, however, expect effort be placed into making a reasonable visual appearance for your application. Your page should not be black-and-white, we don't want to see lots of plain buttons, we would prefer not to see a table of todo items at all! Likewise we expect you to think deliberately about the layout of your website. Be a little creative here.

One important note – try your website at a few different screen/window sizes. We are not formally requiring your website's layout to "work well" across all reasonable screen sizes, but (Especially if you have a particularly large monitor) you should check that it will work on the smaller laptop screens the professor owns. If you are particularly worried about this, feel free to document what screen size you tested your layout at in the README file and we will try to replicate that when evaluating your visual design/layout.

If you're looking for inspiration, consider looking at existing todo list software such as

TODOIST



Today Sun Dec 4

Sorted by due date

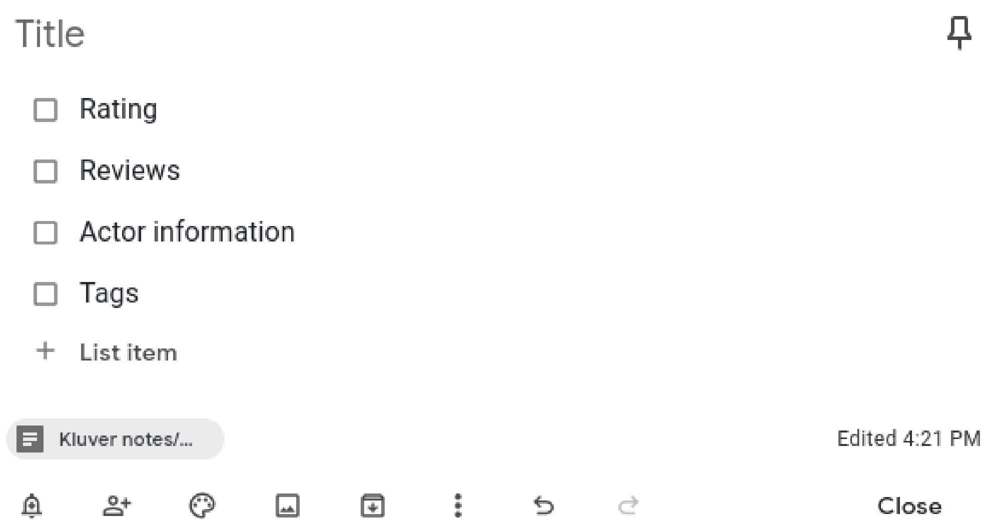
Overdue

tanush PR review
Jan 31 — GroupLens / Short-term

Reboot Office Buttons
Jan 31 — Personal

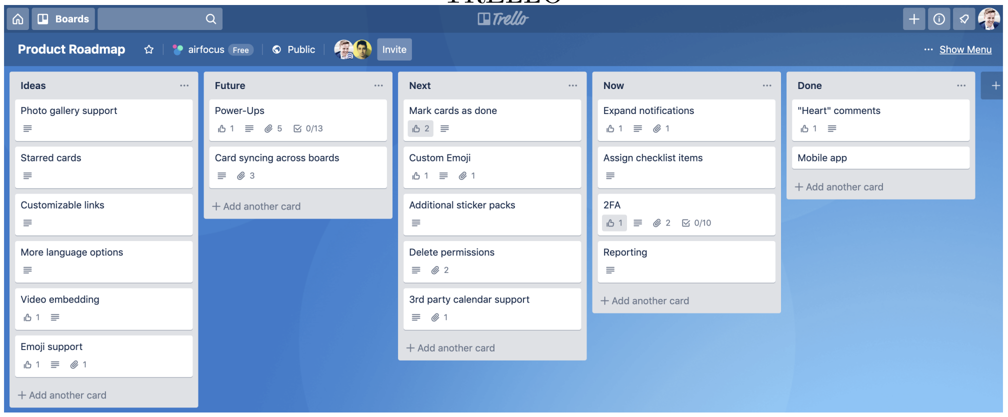
Make a dummy project
Jan 31 — CSCI 5117 / Module 1 setup

GOOGLE KEEP


Title

- ☐ Rating
- ☐ Reviews
- ☐ Actor information
- ☐ Tags

TRELLO



Ideas	Future	Next	Now	Done
Photo gallery support	Power-Ups	Mark cards as done	Expand notifications	"Heart" comments
Starred cards	Card syncing across boards	Custom Emoji	Assign checklist items	Mobile app
Customizable links		Additional sticker packs	2FA	
More language options		Delete permissions	Reporting	
Video embedding		3rd party calendar support		
Emoji support				

Notice how all of these make some use of color (OK, google keep only uses grey, but that's just a part of a larger interface that does use color), and while there are obvious interactive elements, rarely, if ever, are they formatted as buttons. Nothing is laid out in a classic "table" layout (As none of this is really tabular data).

5.4 Above and Beyond

Meeting the basic listed requirements here will be enough to get a grade of 90%. To get a grade of 100% you will need to go above-and-beyond in some way. The exact nature of this is up to you, however a few ideas:

- Particularly impressive visual treatment or user interactivity – this might include relevant animations or interactions, or it could mean a responsive design (using CSS to make the interface work well at almost all screen sizes, including mobile)
- *integrated* extra data – for example a category system, with category navigation/filtering. or a deadline system with sorting, a task-nesting system, etc.

- A working (and well designed) user log in system, with data separated by user. (IF you do anything like this, make sure you give us login information or account creation details in your README file!)
- Other *todo list tracking* related features.

You can also implement some combination of the above.

It is **STRONGLY RECOMMENDED** that you explicitly list what you did that was, in your opinion, "above and beyond" in your required README file.

6 Evaluation Criteria

The rough evaluation criteria:

- (10 points) Preparation and submission
  - These points represent your submission itself. So long as you have a readable README file that explains everything we need to run your software, and it works on CSELabs machines without modification you should be fine.
  - A submission will lose points here for a missing or incomplete submission. Not having a README file, or making us "guess" something that needs to be clearly stated. If we have to read your actual code to use the todo list software, you're probably losing points here.
- (20 points) Style and approach
  - A fully functional, visually impressive, feature rich piece of software can still be built poorly – these 20 points reflect "was it built well" vs. "was it hacked together". As such, it's a bit of a catch all of many different evaluations.
  - You are expected to use good code style (we will be reading/reviewing your code and expect this to not be an annoying/difficult task. Indent your code where reasonable, and otherwise use good judgment about comments, and structure)
  - You are expected to make good use of the many different features available in the web-stack. You will lose points here for "hacky" solutions to problems that ignores, or shows lack of knowledge of, good ways to approach problems. (As an example here – using tables to get a 2-column layout instead of CSS. It works, but it's a hack, and a bad one at that)
  - You are expected to make good use of software engineering principals such as helper functions, separating code into multiple files, avoiding inline/embedded HTML/CSS, using template inheritance/including to avoid code re-use etc.
  - A full credit submission can have one or two *minor issues* – but *major issues* or major "bad choice" moments will lead to large point-loss.
- (50 points) Features – 10 points each for implementation.
  - Todo list view
  - Todo list filtering by done-status
  - Adding todo list items
  - removing todo list items
  - marking a todo item as done or undone without a page-reload.

- (10 points) Visual style
  - A 0 point submission has no CSS, and makes no use of appropriate HTML elements to create a usable visual experience
  - A 1-4 point submission makes bad use of HTML and CSS. The webpage has many things that look "unstyled". It looks like "make it look good" was an afterthought, and one you might have run out of time on, or put little-to-no effort into.
  - A 5-7 point submission has major visual issues, but clearly attention was payed to layout. There is maybe one element that looks unstyled or unconsidered, but an effort was made.
  - A 8-9 points submission has one or two "rough corners" where things could visually be improved, but otherwise has a clear layout. You clearly made a plan and executed it.
  - A 10 point submission uses reasonable HTML and CSS tools to create a deliberate-looking layout. The website has a clear visual identity. You clearly made a plan and executed it well.
- (10 points) Impress us
  - A 9-10 point submission makes us go "wow!, this was amazing!" you've added one particularly complicated feature, or a few well done features/interfaces. All new features are well-considered and bug-free.
  - A 5-8 point submission has added one or two new features of modest complexity/impressiveness. These features are (mostly) bug free and well implemented. Alternatively a score in this range may be given for impressive, but unrelated, new features (a tic-tac-toe game is interesting, but not really connected to todo lists!)
  - A 1-4 point submission has not finished adding new features, added "trivial or unrelated" new features only, or the new features are poorly implemented and do not work.
  - A 0 point submission meets the minimum listed technical requirements above, and goes no further.