CSci 4131 Internet Programming Spring 25 Project: Putting it All Together

Deadline:

Sunday May 11th at 11:59pm

NOTE: Late submissions will **NOT** be accepted.

Change Log

- None yet.

FAQs

- None yet.

Individual Assignment

This is an **individual assignment**. You should not share your code with any other student. Likewise, please do not search online for tutorials on how to build to-do lists, use code from such tutorials, or use Large Language Models (LLMs) such as ChatGPT to assist in designing or implementing your solution. The goal is for you to solve this problem independently. You may (and should), however, research and if possible, try out existing to-do list applications and use them to inspire your solution. Some of the most popular to-do list applications are listed below (just search for them using your favorite browser).

Microsoft ToDo
Structured
Superlist
Things 3
Todoist
TickTick

Important: If we suspect you have used code that you have not developed, you will receive a **ZERO** for the project and an **F** for the course.

There are 5 pages in this assignment specification and requirements document.

Project Size

This project will require more effort than the most work-intensive individual homework this semester. You will build a new website rather than constructing it piece by piece. The project is meant to be open-ended in design and more comprehensive than the individual homework assignments. We suggest starting early to avoid rushing at the end.

Introduction

Throughout this semester, we've studied the core languages and technologies of Internet programming, including:

- HTML, CSS, JavaScript, and the DOM
- HTTP protocol (conceptual and low-level)
- Back-end routing, templating (using pug), and SQL databases.

The purpose of this assignment is to bring all these ideas together and build your first genuine website: a **To-Do List**.

Key Principles:

1. Read the rubric: Understand the grading criteria.
2. Accept no bugs: Fix all issues independently.
3. Consider the layperson: Make your website intuitive and user-friendly.
4. Consider aesthetics: Focus on layout, colors, and spacing.
5. Test carefully: We'll be taking a close look at both your code and your website, so test your code thoroughly and ensure it works as expected.

Requirements

This assignment is intentionally open-ended. Below are the broad requirements:

Technical Requirements:

1. Use Express for your server.
2. Ensure the server runs on CSELabs machines.
3. Store data persistently in your allocated MySQL database.
    - Restarting the server on a new machine should not lose data.
    - **NOTE**: You must provide your database credentials in the `README`, especially if you change your password.
4. Submit all files needed to run your website.
    - Include `package.json` but **do not** include `node_modules`.
    - Test your submission on another computer.
    -

5. Include a plain text `README` file with:
    - If sign in / sign up is implemented: Username and password for users
    - Instructions to launch your software.
    - URLs and features implemented.
6. Do not use third-party CSS or JS (including NodeJS / express) libraries without emailing course staff or posting on Piazza staff asking for approval first.

Visual Requirements:

- Use colors (not just black and white).
- Don't use plainly styled elements (e.g., `button`, `input`, `table`).
- Use flexbox, **not tables**, for to-do list items' layout. This may require you to get a bit more creative in how you present your to-do items.
- Test your website on different screen sizes. Principally, on a reasonably sized laptop screen (e.g., 14 – 15 inch diagonal)if you are using a large monitor. If you are concerned about this aspect, include the screen size you used in your `README` file. We'll try to test on a similar screen size.

Required Behaviors:

Minimum Required Behaviors:

- View existing to-do list items.
- Filter to-do list items (e.g., "done" or "in progress").
- Create, delete, and mark items as "done" or "in progress."
- Use `fetch` requests for marking items without reloading the page.

Advanced Features (Choose One):

1. **Comments and Description:**

    - Dedicated page for each to-do item.
    - Editable descriptions.
    - Add/delete comments.

2. **Categorization:**

    - Assign categories to items.
    - Menu to filter by category.
    - Create/delete categories.

3. **Deadlines:**

    - Assign deadlines to items.
    - Sort items by deadline.
    - View overdue items.

4. **Templating**

    - Use Pug to generate the HTML pages whenever possible, generating HTML with any initial data on the server side.

Grading Scale:

Here are the following grading ranges you can earn for this project:

- **0%-75%**: tables allowed, user accounts not required, minimum required behaviors implemented and working well
- **75%-85%**: Minimum Required behaviors, tables not allowed, one of the advanced features working well
- **86% or higher**:
- Minimum Required behaviors, tables not allowed, one of the advanced features working well
- ***And must implement ALL of the following behaviors to support USER ACCOUNTS:***
    - Account creation interface (Webpage or web pages to enable account creation)
    - Account deletion interface (Webpage or Webpages to enable account deletion)
    - Login/logout
    - Password hashing
    - User-specific to-do lists

Evaluation Criteria

1. Preparation and Submission (10 points)

- Include a readable `README` file with instructions on running your project, any credentials needed to use your project and database credentials.
- Ensure the project works on CSELabs machines.
- Failure to properly do this may result in us not being able to run your project, resulting in a **0** for the project.

2. Style and Approach (10 points)

- Use good code style and structure.
- Avoid hacky solutions.
- Follow software engineering principles.

3. Visual Style (10 points)

- Use CSS and HTML effectively.
- Create a deliberate and visually appealing layout.

4. Core Features (35 points)

- Creating, deleting, viewing, filtering, and marking to-do list items.

5. Advanced Features (10 points)

- Fully implement at least one advanced feature.

6. Flexbox (10 points)

- Uses flexbox for layout --- no tables.

7. User Support (15 points)

- Implement user accounts and related features.

Submission Instructions

1. Submit all of your files to the project submission on Gradescope.
    - This includes your `README`, HTML, CSS, and JS files, and any other files needed to run your project.
    - Do not include `node_modules` or any other unnecessary files.
    - Do not submit a parent folder, only the files **in** the project folder.

2. **Reminder:** Late submissions will not be accepted.