Final Project Proposal

Due: Wednesday May 22th, 2024

Submission Requirements (per student):

- `proposal.pdf` (PDF of this completed proposal document)
- `wireframes.[png|pdf]` (1-2 wireframes, at least 1 UI flowchart, any other brainstorming/design phase notes)
- `index.html` (main landing page for your store)
- `product.html` (see HTML requirements for alternative view-switching implementation)
- `styles.css` (shared stylesheet for your store)
- `imgs` (folder containing at least one product image, may be template for all products)
- *Other files, such as page-specific CSS and starting JS may vary*

Website Overview

The objective of the project is to build an e-commerce store (or a similar theme). Your proposal should answer the following questions:

- What type of e-commerce store are you building?
- What is the overall theme of the website (sleek, retro, modern etc)?
- Who is the intended user(s)?
- What exactly does your e-commerce store hope to accomplish? (Many e-commerce stores already exist, why is yours different).

Remember that this is a chance to be creative while practicing putting together everything you've learned in CS132 in a comprehensive "beta-version" project. You won't be implementing any actual products/transactions :) This should be 3-4 sentences answered below.

Answer: The type of e-commerce store that I plan to build is a Pokemon Store that will help users build a team for Pokemon-related games. My plan for the overall theme of the website will be Pokemon-themed with a vibrant look, and the intended audience are the thousands of players of Pokemon games worldwide. The e-commerce store will list Pokemon on the store and allow users to add them to their cart. The store will adapt to recommend Pokemon for a player to choose from based on common team layouts from Pokemon Showdown data. Product pages will show a Pokemon's type, strengths, weaknesses, etc. and allow users to leave comments on what team the Pokemon works well in. Once a team of 6 is chosen, the user can checkout their Pokemon.

> ***Note on Alternative Theme:*** *This year, you may also choose an appropriate alternative theme besides an E-Commerce store. Two examples of appropriate alternative projects include a game manager or a message board.* ***You must get official approval from EI for an alternative theme.***
>
> *Alternatives must have reasonable equivalents for the requirements listed in this project, and you'll need to clearly specify your proposed Option Features in your proposal, matching given*

options as close as possible. An example for a game manager translation from an E-Commerce store could include:

- *Any product features (main store view, single-product view, product management, etc.) could be replaced with an entity like a user's game score or a specific game with distinguishing attributes such as difficulty, description, category, etc.*
- *Users vs. admins are pretty straightforward for translation.*
- *Your backend API could have endpoints for GET game data/score/player data and POST score/new player, etc.*
- *The cart requirement will require some thinking on your end, but a reasonable proposal that meets the learning objectives should be given.*
- *Games could be an existing game scoreboard (e.g. NYT Crossword scores) or a game that you may have implemented in JS (ok if the game is implemented in a scoreboard).*

Collaboration Interest

You will have the option to collaborate with a partner (chosen or random) for the back-end part of this assignment. This was added to the Final Project last year, and is strongly recommended based on feedback and overall learning outcomes of building a full-stack application with another student. All students are still required to implement their own front-end requirements, whether or not they use the same API built with a partner, but students are encouraged to collaborate (either with their partner or others) on their different front-ends and layout strategies.

Indicate whether you are interested in a chosen partner (if you know who right now, please provide name and Caltech email), randomly-assigned partner, or no partner (you may change your mind later).

Note: On Thursday, May 16th, there will be space in lecture to work on the proposal and find partners; you can also use #fp-partner-search to find a partner.

Answer: No partner

Chosen Features

For both the front- and back-end you are required to implement the "Required Features" listed below.

In addition, you will also need to choose and implement **at least 2** of the “Chosen Features” from the ones below in order for you to customize this project and make it your own.

You are welcome and encouraged to add more features beyond this - this makes for a really strong project and a quality portfolio piece to showcase. In order to meet the requirements of this assignment your 2 "Chosen Features" should be from the ones listed below. However, if you think of a feature you would like to implement and is not on the list, you may request permission from Melissa to have that feature count as one of your two chosen features. Before requesting, note that in order to be considered for approval the requested feature must be one that requires both front- and back-end support (similarly to the other

chosen features listed). The requested feature should not have a similar implementation to the other chosen features on the provided list.

The required features and "chosen features" from the spec are provided after the Required Features for reference.

Out of the “Chosen Features” for both the front-end and back-end, what features will you be implementing? **Note: You may change your mind later as you progress through the term and the Final Project.**

Answer: Option 3 (allow users to submit “reviews” on teams that a Pokemon works well in)

Option 5 (users can change the “color” of a Pokemon from “normal” to “shiny”, they can also select up to 4 moves from the Pokemon’s moveset)

Front End (Strictly Individual)

Note: The only collaboration allowed in this part is linking up to a collaborated API, and wireframes are also fine to work on together (as long as they are different). We expect clearly distinct front-ends for a shared API (which is a great learning experience to reinforce isolation of dependencies between clients and APIs), and submissions with each partner having a clearly distinct client program will be considered stronger submissions. Ask EI for any clarifying questions on these expectations.

- **Required Features**:
    - A “main view” page: a way to see all of your service's products
    - A “ single view” page: a way to see one product (should contain at least 3 pieces of information such as the price/rating/name/colors/sizes it's available in etc.). You may choose to implement each "view" as a separate HTML page (e.g. index.html, contact.html, cart.html, etc.) or dynamically using JS/DOM manipulation.
        - The single view should be applicable and functioning for all products on your main view page.
    - A customer service view (e.g. contact page)
        - Must have a form for users to submit complaints/questions.
        - Must include some level of input validation (HTML5 or JS) to prevent invalid messages (such as empty messages) from being sent
            - This includes indications to the user (such as a message displayed on the screen) indicating that there is an issue with the form if all parts of the form are not filled out
            - You must also indicate to user that their message has been received
    - A way to filter through the products to only display ones with certain properties
        - This can either be accomplished with a search bar *or* a dropdown/other UI elements that, once selected, only display products matching the chosen query
    - A "cart" view which displays a list of items that the user is interested in
        - A user should be able to add/remove items from the cart
    - Must have at least 10 CSS rule sets in your .css file and 10 unique rules **at minimum**
        - Must change at least 2 box model properties (border, padding, margin, height, width)

        - Must import and use at least one font from Google Fonts
        - Must use at least 2 flex properties (this excludes setting display: flex)
    - Must use fetch to communicate with the back-end (your web service)
        - Errors should be gracefully handled and displayed to the user (you may not use alert, console.log or console.error)

Back End (Recommended But Optional Partner)

- **Required Features**:
    - Must have at least one JSON response (including a plain text response is optional)
    - Must have a GET endpoint to request and return all of your service's products
    - Must have a GET endpoint to request and return information about a single item
        - This endpoint should be functional for each item in your e-commerce store
    - Must have a GET endpoint to request and return some of your service's products based on a filter (price, category, type etc.); **this may be implemented with optional query parameters for the first GET endpoint above**
    - Must format the data from each request appropriately to send to the front-end (you should be able to explain why the way you designed your response schema are effective)
    - Must have a POST endpoint to accept and store all feedback received via customer service form (see front-end requirements)
    - Must have at least 2 files to store your data as txt or JSON (SQL tables are also sufficient if you have SQL background)
    - Each file/table must contain at least 2 different attributes that are used to store or retrieve useful data
    - There should be at least 15 items in your E-commerce store (although we encourage you to add more to create a more interesting storefront).
        - (If you choose to use SQL, these items should be inserted into the table in a setup.sql file using INSERT statements)

Chosen Features

OPTION 1: An additional view indicating when/what promotions are available at your e-commerce store. There should be at least 5 promotions/sales available. All the information for the sales/promotions should be retrievable from a GET endpoint.

OPTION 2: A way for a user to buy a product/item from your e-commerce store. You should update the page indicating to the user the item has been sold and/or the order is on its way. The updates to the stock of the item should be supported with a POST endpoint. Each time an item is bought the overall stock should be decremented.

OPTION 3: A way for a user to leave a review on an item. This review should be displayed on the page for other users to see and should be persistent across page reloads (using the appropriate storage technology). This feature should be supported with a POST endpoint so that each review left will be added to your file system or database.

OPTION 4: An additional view in your e-commerce store so that users have a way of becoming "loyal customers." This should be achieved with a form and should require at least three pieces of information from the customer (name, email, address, phone number etc.). Once a user joins successfully there should be a message displayed on the page indicating this. This should be supported with a POST endpoint so that all information on "loyal users" is added to your file system or database.

OPTION 5: A way for a user to customize a product. The user should be able to customize at least two features such as color of the product, material of the product etc. These changes should be supported by a POST endpoint which updates the item description with the new customizations.

OPTION 6: An additional view for a FAQ page. This should have at least 5 frequently asked questions and the answers displayed on the page for customers to read. This should be supported by a GET endpoint from which all the frequently asked questions and answers can be retrieved.

OPTION 7: An additional admin view allowing an administrator to add and/or remove items from the store as well as update current inventory. This should be supported with a POST endpoint which appropriately updates the file system or database with the correct information based on whether the admin has added/removed an item from the store.

OPTION 8: An additional admin view for managing the accounts that the e-commerce store users have created. An admin should be able to edit a user account and modify information. This should be supported with a POST endpoint which will appropriately update the user information based on the administrator's actions.

Potential Showstoppers

- List any problems you think could come up in your project.
- This section should be present even if you have no questions so we can see you're thinking about this.

Answer:

- **Since there are so many Pokemon, there are bound to be some Pokemon that are never used in a team. For these Pokemon, the recommendation will ignore the Pokemon and wait for a Pokemon featured in a team, and provide a team based on that.**
- **The PokeAPI does not feature all the available Pokemon for some reason, which I will have to look deeper into.**

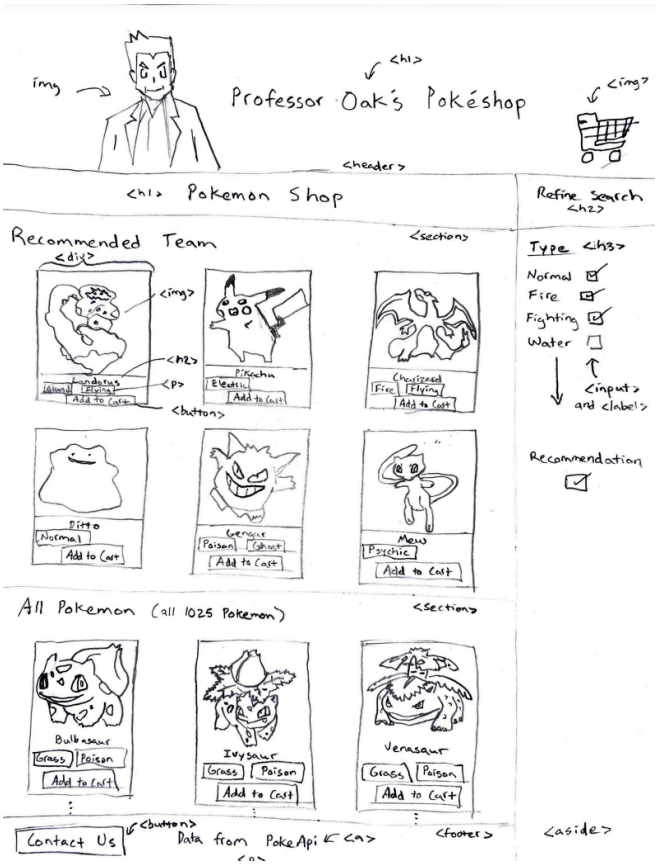
Wireframes and UI Flowcharts

Upload 1-2 wireframes for your proposed website here (max 1.5MB, png or pdf). It doesn't need to be a work of art, but we encourage you to consider both layout, design, and UI annotations/workflow diagrams.

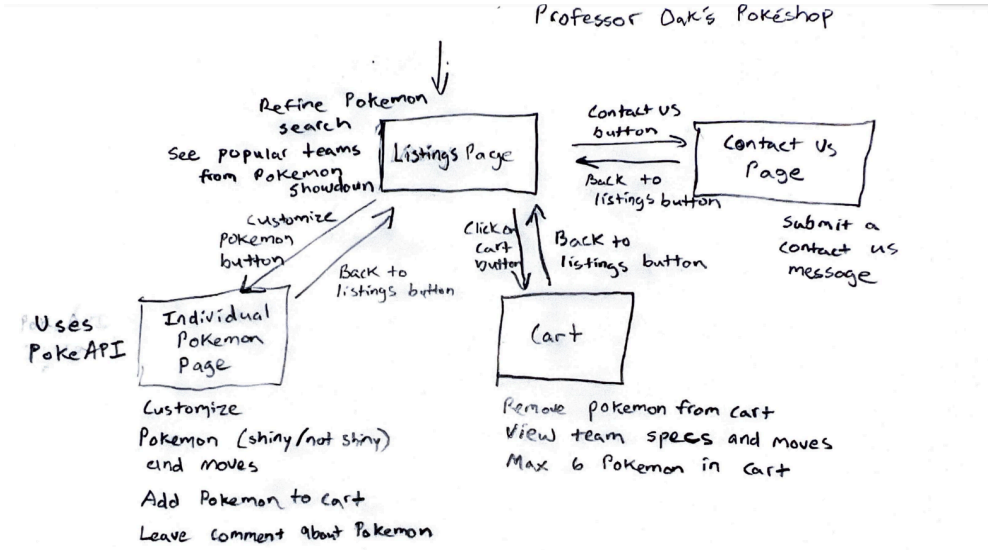
In addition, include at least one visualization (e.g. flow-chart) of a runthrough of your proposed website program from start to finish. The following is an example (a text-based version, though a visual flow chart should be used in whatever format you'd like, ideally more than one):

- User visits home page of website (index.html) -> is presented a list of options for the main menu view -> selects ('menu') for show all cafe products listed in '<n>: <product name> <price>' format, where <n> starts with 1 -> clicks the Black Coffee "+" button 3 times to add to their cart, and a cart icon at the top-right corner turns green for 2 seconds (per click) to indicate something was added - > user selects cart icon and is directed to cart.html -> cart.html shows a list item with the Black Coffee image, 3x, and the total price in $X.XX format ("Cart is empty" if no cart yet) -> user selects checkout -> POST request is made to our API to submit cart to simulate a purchase (with basic user information and list of items with quantity) -> API returns with a success message (in JSON) and estimated wait time ('simulated')

Wireframe of the index.html page:



Website wireframe:



Required Proposal Code

Last year, students gave feedback that it would have been helpful to incorporate some code in the proposal to offload work later in the term.

With your proposal, submit at minimum, **2 HTML files linked to 1 shared CSS file** that is related to your proposed project. Based on prior feedback, we recommend starting JS as well though, and you can hard-code anything you'd like as proof-of-concept.

Clearly **indicate in your source code as comments** areas that you plan to extend/possibly modify (we won't give in-depth feedback at this stage, but at least 1.5 hours spent on the implementation of these files is expected for full credit).

You can refer to the Coffee Shop case study for an example of a simple store gallery view/single product page and the amount of effort we're expecting at this stage, but you may not copy/paste any code from it. **You *may* use some of your prior CP code (e.g. themes or basic interactivity that make sense in your store), but no more than 75% of each file may be from a CP, and you are expected to have implemented any CP feedback in this submission.** Comments that identify trade-offs/questions/"stretch goals" will be targeted for feedback by your grader (including if you add any optional JS). For material of popular interest in identified stretch goals that may not be core to the course content, EI may try to incorporate where appropriate in upcoming lectures.

Proposal-Stage Code Requirements:

- **At least 2 HTML file with 5+ distinct tags each**, 2 of which are semantic HTML5 tags
    - 1 landing page (index.html)
    - 1 single-view product page (e.g. product.html)

        - Exception: You may use JS to implement a single view in index.html (e.g. hiding an "all products" view and showing a single product view when a product is clicked)
        - You may hard-code HTML for the single product or hard-code JSON data (which would be populated from a backend API) for this stage, which is a good strategy for mocking up a front-end before hooking it up to a backend
- **At least one styles.css file** for shared styles across your store's pages (you may include other CSS files for page-specific styles)
    - At least one flex container must be used
    - At least 4 different rulesets
    - At least 4 different CSS properties
    - No redundancy in CSS (2-3 repeated box model/layout properties are ok, but encouraged to factor out with a grouped selector)
    - Bootstrap or other CSS libraries are not allowed for the proposal (we want you to get a basic layout/theme without using a library to start)
- Additional files such as JS are optional, but encouraged

Additional notes/questions for graders may be provided here:

The front-end is about 40% completed at the moment. No JS code is implemented yet.

The index.html page needs a lot more work, specifically in organization of the refine search, implementation of the cart, inclusion of all Pokemon, and other fine details. The front-end is very rough at the moment and likely to change significantly to improve the website's look and feel.

The product page's information is all there, but the front-end will change significantly as well.

Contact Us related information is not yet added.

Review

This year, we are adding a small component to get students feedback from their peers and to practice giving feedback on proposed projects. You do not need to spend a lot of time on feedback (you may earn up to 3 *additional* points for above-and-beyond effort, such as reviewing multiple student posts with thoughtful feedback), but for for full credit you must:

- Post your overview followed by one part of your proposal (e.g. wireframes or UI diagrams) on #final-project-review and include at least one question you have you'd like feedback on
- Review one other posted project and provide 1-3 sentences of feedback that demonstrates at least 5 minutes of reflection of trade-offs and concepts you've learned so far. Feel free to bring in your own user experiences! For example, you may have experience inputting a long last name that a website has a size limit on when creating an account, and could note this when suggesting a modification to another student's approach for representing names.

Link to your Discord post:
https://discord.com/channels/1224522707698192487/1239382960453718089/1242924263200260217

Link(s) to your Discord feedback:
https://discord.com/channels/1224522707698192487/1239382960453718089/1242932156947366049

Collaboration

For projects with partners, this section is required (if not, you can leave it out, or note that you are decided yet). Both students should provide a brief summary of their planned workload distribution, method(s) of collaboration, and 1-2 points you are most interested to do in the project. You may also clarify any concerns/confidence for your partner work here. Feel free to provide a paragraph or bullet points; we're looking for you to have thought this out and discussed your plan for collaboration at this step.

[Student 1 Name] Answer:

[Student 2 Name] Answer:

Open Questions

- Any other questions or concerns? If so, is there anything the course staff can do to help accommodate these concerns?
- This section should be present even if you have no questions so we can see you're thinking about this.

Answer:

- **Are significant changes to the back-end of the website allowed if they still fulfill the requirements?**
    - **My backup if the recommendation system using Pokemon Showdown data doesn't work is to use well-defined team layouts based on type instead of specific Pokemon.**
- **How are the back-end and front-end parts weighted?**