Final Project Proposal

💡 Project names..

Ashes & Hallows – Rebirth through death. Phoenix + Deathly Hallows fusion.

Cloak & Ember – Invisibility Cloak + Phoenix rebirth. Fire and hidden wisdom.

Purpose:

I've always loved Harry Potter—not just the magic, but the deeper themes about love, resilience, loss, loyalty, and friendship. These stories have stuck with me since I was a kid and still mean a lot to me now as an adult. As a data nerd and a lifelong fan, I saw this project as a fun way to blend both passions.

I have been a software engineer for around 6 years but I am currently transitioning from front-end specifically into data science. I've been working as a Business Data Analyst for the last 9 months and have started looking at everything—including this JSON—through a more analytical lens. This project is my way of stretching both sides of my skillset: clean, accessible front-end UI *and* meaningful data organization.

My goal is to make something whimsical, accessible, and technically robust using data from multiple sources. I want this to be fun for users (especially other HP nerds), but also thoughtful—designed with sound structure, animation, interactivity, and deep exploration of the relationships in the data itself.

Audience

Harry potter nerds like me of course... I dont know if there is more to say here. Honestly? I want to build something I'd be proud to share with my HP-nerd family. (We literally have a family group chat and share HP memes and gifs articles etc.. its probably a bit much but its fun.) I want them to play with it and light up when the cards glow, the spells sparkle, and the wand colors shimmer on hover. I want to use Icons that I might be able to animate and give life to the page.

Project Scope & Features

Here's how I plan to structure the app, making sure each page is purposeful and grounded in data.

Pages

- **Home** – Overview of category options (characters, spells, elixirs, etc.) with navigation
- **Category Browser** – Filterable view by category (e.g., all students in Ravenclaw)
- **Item Detail Modals** – Dynamic pop-ups for each item with theme-based animations

API's

- `hp-api.onrender.com` for Characters, Wands, Patronuses, and Houses
- `wizard-world-api.herokuapp.com` for Spells, Elixirs, and Ingredients
- `rapidapi.com` for Quotes

Categories (Built from API data):

- Characters

- Spells
- Elixirs
- Ingredients
- Houses
- Wands (derived from wand objects in character JSON)
- Patronuses (derived from character data)
- Quotes (with optional filters by speaker)

Highlighted Features

- Color and icon-coded category cards (e.g., light color of spell or elixir matches card animation)
- LocalStorage for recent items viewed and favorites
- Fully responsive grid & card layout with animation
- Accessibility-aware markup (e.g., keyboard navigation, ARIA where needed)
- Audio effects on interactions (e.g., a bubble sound on elixir card open)

Data Exploration Plans

I want to go deeper than surface-level data. Some examples:

- Use `wand.core` and `wand.wood` from character JSON to group characters by wand type.
- Show house distributions across students vs staff, with percentages and charts.
    - We all know how popular Gryffindor is.. but I want to see the others too.
- Highlight which spells are verbal vs non-verbal (`canBeVerbal`) and color-code their `light`.
- Pull ingredients from elixir data and allow filtering by ingredient (e.g., "show me all elixirs that use Mandrake").
- Display quotes by speaker and connect them to character pages.
- Group Patronus types and show how rare they are across the dataset.

Anticipated Challenges (Real Ones!)

- Some API endpoints return inconsistent or incomplete data. (That is what happens when you are working with copywrited material. I wanted to include magical beasts and herbology but I guess elixirs will have to be good enough.)
    - I'll need fallback strategies and clear empty states (e.g., "No Patronus listed for this character").
- Connecting data across APIs that don't share common IDs (e.g., linking a quote to the character without fuzzy matching).
    - Maybe the quotes wont connect to the characters and just be a random generation on the bottom of the dialogue / modal.. then I wont have to make a connection at all..
- Ensuring responsive is taken in consideration for everything made.
- Keeping the scope tight enough to finish everything without compromising quality.
    - We only have 3 weeks left and I am taking 4 other classes - 3 online and 1 institute in person

Major Functions:

Category Navigation Cards (Home Page)

What it does: Displays a grid of clickable category cards (e.g., Characters, Spells, Elixirs, Ingredients, Wands, Houses, Patronuses, Quotes).

Filter Characters by House, Role, Wand, Patronus, or Ancestry

What it does: Allows users to filter characters across several traits.

Details: Users can toggle between filters like:

- House (Gryffindor, Slytherin, etc.)
- Role (Student, Staff)
- Wand core or wood
- Patronus type
- Ancestry (pure-blood, half-blood, etc.)

  This function dynamically updates the character grid to reflect selected filters.

Detail Modals with Contextual Styling

What it does: When a user clicks on a card, a modal opens showing detailed information.

Details:

- Spells include incantation, type, light color, and whether it can be cast nonverbally.
- Elixirs show ingredients, difficulty, and visual color cues.
- Characters display biography-style cards (name, house, wand, Patronus, role, etc.)

  Modals are styled to match the category—for example, spells glow based on their `light` value from the JSON (e.g., red, green, etc.).

Wand Explorer

What it does: Groups characters by their wand wood or core material.

Details: Clicking on a wand core (e.g., “phoenix feather”) shows a list of all characters who use that wand core.

Elixir Ingredient Tracker

What it does: Allows users to explore potions and elixirs by the ingredients they use.

Details: Each elixir includes a list of ingredients. Users can click an ingredient (e.g., “Mandrake”) to see all elixirs that use it. This enables a cross-referenced view between categories.

Random Quote

What it does: Displays quotes filtered by character displayed on the modal of a card (using the RapidAPI Harry Potter Quotes API.)

Details: From the Character Detail modal, users can view famous quotes attributed to that character. Possibly a “Random Quote” button for general fun on the main menue. This adds emotional and literary depth to the app.

Card Animations and Thematic Interactions

What it does: Adds polish and delight via hover/entry animations, glows, and subtle sound effects.

Details:

- Spell cards glow with their spell's color
- Potion Cards glow based on potion characteristic color and has a bubble sound
- Quotes fade in with a soft animation
- Cards flip or pulse on hover or modal open

  (This makes the entire experience feel interactive and magical without being overwhelming.)

Paralax Scrolling..

- I have not even begun to imagine how or what I want to do this to but I think it would be a good way to show advanced CSS

Favorites (via LocalStorage)

What it does: Tracks user interactions with cards and stores them locally.

Details:

- Favorites are saved and displayed on the home page for easy access
- Users can "favorite" a character, spell, or elixir

  Everything is persisted in LocalStorage so data isn't lost on refresh.

File Structure for ReadME.md

```
/project-root
│
├── /css
│   └── style.css
├── /images
│   └── [icons, logos, backgrounds]
├── /js
│   ├── main.js
│   ├── api/
│   │   ├── hpApi.mjs
│   │   ├── wizardWorldApi.mjs
│   │   └── quotesApi.mjs
│   ├── components/
│   │   ├── CardRenderer.mjs
│   │   ├── Modal.mjs
│   │   └── Filters.mjs
│   ├── utils/
│   │   └── helpers.mjs
│   └── router.mjs
│
├── /data (optional, for caching)
├── /partials
│   ├── header.html
│   └── footer.html
```

```
├── index.html
├── category.html
├── detail.html
├── .gitignore
└── README.md
```

Module List

```
# Harry Potter Project Module List

This is a breakdown of the major JavaScript modules that make up the application.

## 1. router.js
Handles basic page routing and shared template loading.
- Injects shared HTML templates (like header and footer)
- Determines which page is being viewed and loads correct module logic
- Helps mimic a single-page app structure even with static files

## 2. api.js
Centralizes all fetch logic for each API.
- fetchCharacters(), fetchSpells(), fetchElixirs(), fetchQuotes()
- Accepts optional query params (e.g., house, light color, quote author)
- Includes error handling and parsing logic

## 3. characterList.js
Renders a full list of characters from the API.
- Dynamically loads and displays cards for each character
- Includes filters for students, staff, or specific houses
- Adds click listeners to open a detail modal per character

## 4. houseDetails.js
Displays details for each individual Hogwarts house.
- Renders characters in the selected house
- Calculates and displays percentage breakdown of students vs staff
- Links to navigate between house and character pages

## 5. spells.js
Displays spells in a color-coded format based on the "light" key.
- Fetches spell data and creates animated cards
- Opens spell modals with incantation, effect, and type
- Includes filtering or sorting logic if time allows

## 6. elixirs.js
Displays elixir/potion data with color accents based on “characteristics.”

- Lists elixirs with difficulty, effects, and key ingredients
- Ingredient list shown in modal view
- Supports filtering by difficulty or searching by keyword
```

```
## 7. quotes.js
Displays Harry Potter quotes.

- “Random Quote” button fetches a quote and displays it
- Can filter by character name if supported by the API
- May allow cycle-through or refresh of quotes

## 8. utils.js
Utility functions used across the application.

- DOM helpers like qs(), setClick(), renderWithTemplate()
- LocalStorage helpers
- Reusable scroll, animation triggers, or fallback handlers

## 9. storage.js
Handles localStorage-based features.
- Add/remove favorites
- Track recently viewed characters or spells
- Includes duplicate checks and timestamping if needed

## 10. modal.js
Controls how modals are rendered and behave.

- Dynamic content rendering based on data object type
- Accessibility features (focus trap, ESC key to close)
- Animation hooks for open and close

## 11. animations.js
Centralizes animation logic.
- Trigger glow on spell card hover
- Animate elixir cards based on characteristic color
- Works with the "light" and "characteristics" keys

## 12. stats.js (Optional/Bonus)
Handles house or data-driven stats.
- Student/staff breakdowns per house
- Common wand materials or frequent elixir ingredients
```

I am passionate about accessibility and while I think this project is already kind of massive and I may need to condense some of this, I want to be sure what I create meets WCAG 2.2. I know its going to do the “accessibility scan” in light house but thats superficial testing so .. yeah.

Graphic Identity

This color combo will try to have high contrast dark theme feels..

- Background: Dark Royal purple if flat color.. image of a dark starry background or even the great hall ceiling.. “bewitched to look like the night sky”

- **Primary Color**: Gold `#D4AF37` (Think golden snitch vibes…) or if too harsh then `#846B3A` — a little duller but still nice like butterbeer..
  - accents navigation, highlights important buttons, or frames interactive elements.
- **Secondary Color:** Dark Blue `#0B163B` or Royal Purple `#5758ff`
  - May not need because the cards will have a color based on the category etc.
- **Accent Colors (House Themes)**:
  - **Gryffindor**: `#7F0909` (Deep Red)
  - **Slytherin**: `#0D6217` (Emerald Green)
  - **Ravenclaw**: `#0E1A40` (Navy Blue)
  - **Hufflepuff**: `#EEE117` (Soft Yellow)
- Each Category card type will have its own color so need to review API data for others like teal or something.
- **Text**: `#FFFFFF` (White) for clean high-contrast body text.

Project Timeline (Weeks 5–7)

Week 5: Foundation, Data Setup & API Integration (Planning + Scaffolding Phase)

Primary Goal: Set up your app's structure, test APIs, and start rendering basic data from JSON responses.

Deliverables:

- [ ] Set up GitHub repo and connect to VS Code
- [ ] Create project structure (`index.html` , `js/` , `css/` , `partials/` , `assets/`)
- [ ] Design base HTML layout and navigation UI (cards for each category)
- [ ] Build reusable header/footer using template injection
- [ ] Set up JavaScript module system (ES Modules, utility functions, router, etc.)
- [ ] Test API calls from:
  - `hp-api` (Characters, Students, Staff)
  - `wizard-world` (Spells, Elixirs)
  - `RapidAPI` Quotes (with key)
- [ ] Create sample fetch for **Characters API** and render a few basic cards
- [ ] Set up Notion or Trello for progress tracking
- [ ] Draft your style system: color palette, category icons, animations

---

Week 6: Dynamic UI, Data Filtering & Modal Integration (MAJOR Build Phase)

Primary Goal: Make your site interactive and dynamic. Focus on functionality + polish.

Deliverables:

- [ ] Implement **category filtering UI** (e.g., buttons or nav tabs to sort by Characters, Spells, etc.)

- [ ] Loop through API data and render responsive **grid of cards** for:
  - Characters (basic view)
  - Spells (color-coded by `light`)
  - Elixirs (color-coded by `characteristics`)
- [ ] Implement **modal system**:
  - Display full details of each card
  - Style modals differently by category
- [ ] Add visual feedback (loading spinners, “no data” messages, basic error handling)
- [ ] Add simple animations (e.g., card hover, modal transitions)
- [ ] Enable quote API integration with a **"Get Random Quote"** feature or filter by character
- [ ] Add LocalStorage setup for:
  - Recently viewed items
  - Favorites
- [ ] Basic mobile responsiveness pass (flex/grid, card stacking)

Week 7: Polish, Accessibility, Animations, and Final Testing (Refine & Deploy Phase)

Primary Goal: Refactor, test, polish, and submit. Focus on user experience and performance.

Deliverables:

- [ ] Add **category-specific animations and glows** (e.g., spell light animations, potion color pulses)
- [ ] Finalize **favorites** and **recently viewed** interactions via LocalStorage
- [ ] Ensure **a11y best practices**: proper alt tags, focus states, semantic elements
- [ ] Run **Lighthouse** audit and address any major issues in:
  - Performance
  - Accessibility
- [ ] Add sounds for magic/hover effects
- [ ] Final review of error handling (e.g., failed fetch, empty states, 404 page)
- [ ] Deploy to GitHub Pages
- [ ] Final code cleanup: comments, ESLint fixes, modular cleanup
- [ ] Record walkthrough video and complete submission form

Project Links

Tello:

https://trello.com/invite/b/67e7d088993eee56de75f123/ATTId103096a3c0a584cfcafeff2d1117b81D44B6F84/wdd330-final-project

Github Repository:

https://github.com/CaitlinMEvans/wdd330

Github Pages (for deployed site?)*

Not sure..