Kim Hogle’s Project Plan: Responsive Web Application for Movie Recommendations

I. Introduction

- Objective: Develop a fun interactive web app that lets users vote on their favorite movie from a pair, about 30 times. Based on their choices, it identifies their favorite genres and shows recommended from TheMovieDataBase (TMDB). The project uses TMDB API, JavaScript, HTML, CSS, and is designed to be responsive across different devices. The project demonstrates practical skills in API integration, data analysis, and modern web development best practices.
- Tools and Technologies: HTML, CSS (Grid, Flexbox, Media Queries), JavaScript, Chart.js, TBDM API.
- Goals:
    - Implement a responsive design.
    - Integrate various web development features.
    - Utilize a third-party API for data retrieval.
    - Ensure a visually appealing and user-friendly application.

II. Responsive Design

1. Implementation
    a. Media Queries: Implement media queries to adapt the layout for mobile and desktop screen sizes.
    b. CSS Grid and Flexbox: Use CSS Grid and Flexbox to create flexible and responsive layouts.
    c. Responsive Components: Ensure all components (graphs, forms, tables) adjust appropriately across different devices.

III. Feature Implementation

1. Selected Features
    a. Feature 1: Analyze data that is stored in arrays, objects, sets or maps and display information about it in your app.
    b. Feature 2: Create a function that accepts two or more input parameters and returns a value that is calculated or determined by the inputs.
    c. Feature 3: Visualize data in a user-friendly way. (e.g. graph, chart, etc)
    d. Feature 4: Persist data to an internal API or use localStorage.

2. Integration of Third-Party API
    a. TheMovieDataBase API: Used to fetch movie data and genres.

IV. Data Handling and Analysis

1. Data Storage and Retrieval
    a. Store user voting data and selected movie information in arrays or objects.
    b. Track selected genres using a Map or object to count genre frequency.
2. Data Visualization
    a. Use Chart.js to show genre vote counts in a chart.
    b. Alternatively, display a ranked list of top genres based on votes.

V. Advanced Features (Optional)

1. Form and Data Storage
    a. Add a form for username input and validate with regular expressions.
    b. Store results or preferences in localStorage to persist across refreshes
2. Data Persistence
    a. Use localStorage to store user genre preferences and recommendations.
    b. Persist data between page reloads to enhance user experience.
3. Interactive UI Features
    a. Display voting progress, responsive movie cards, and genre tags dynamically.
    b. Implement visual feedback (e.g., animations, transitions) to enhance interactivity.

VI. Project Development

1. Node.js Web Server
    a. (Optional) Set up a simple Node.js server using Express.js for future API handling.
    b. Project is primarily frontend-based using HTML, CSS, and JavaScript.
2. Database Interaction
    a. Not required. All data handled in-browser using arrays/objects and optionally localStorage.
3. JavaScript Framework
    a. (Optional) Future upgrade path could include migrating to React or Svelte for SPA behavior.

VII. Review Process

1. Internal Review

    a. Thoroughly test genre analysis and recommendation logic.
    b. Validate responsiveness across mobile and desktop devices.
2. External Feedback
    a. Demo the project to peers and mentors.
    b. Use feedback to polish UI and improve clarity of results.

VIII. Documentation and Final Submission

1. Code Annotation and Documentation
    a. Use comments in all JavaScript functions to explain logic.
    b. Write a comprehensive README.md that includes:
- Project overview and user story
- Setup and run instructions
- Feature list and implementation breakdown
- API usage details (TMDB endpoints)

2. Final Submission
    a. Ensure site is live on GitHub Pages and fully responsive.
    b. Link GitHub repository and live demo in your submission form.