# Responsive Web application for creating a Character for the Polaris RPG

### **I. Introduction**

* Objective: Create a web application that streamlines the creation of a Character in the Polaris RPG system
* Tools and Technologies: HTML, CSS (Grid, Flexbox, Media Queries), JavaScript, Unsplash API
* Goals:
  + responsive design
  + Streamline the character creation process in a concise manner
  + Utilize third-party APIs to create a visually distinct experience
  + user-friendly application.

### **II. Responsive Design**

1. Implementation
   * Media Queries: Implement media queries to adapt the layout for mobile and desktop screen sizes.
   * CSS Grid and Flexbox: Use CSS Grid and Flexbox to create flexible and responsive layouts.
   * Responsive Components: Ensure all components adjust appropriately across different devices.

### **III. Feature Implementation**

1. Selected Features
   * Feature 1: Use arrays, objects, sets, or maps to store and retrieve information displayed in the app.
   * Feature 2: Visualize data in a user-friendly way .
   * Feature 3: Create a function that accepts two or more input parameters and returns a value that is calculated or determined by the inputs.
   * Feature 4: Retrieve pictures from a third-party API and use it as a background display
   * Backup Feature: Use a regular expression to validate user input and either prevent the invalid input or inform the user about it (in all cases prevent invalid input from being stored or saved).
2. Integration of Third-Party API
   * Unsplash API: To create select different images for the page background

### **IV. Data Handling and Analysis**

1. Data Storage and Retrieval
   * Create JSON Files to store character data
   * Implement functionality to update and retrieve this data as needed.
2. Data Visualization
   * Allow character data to update on screen as choices are made
3. Data Handling
   * Store various skill data in arrays to be referenced as needed

.

### **V. Advanced Features (Optional)**

1. Interactive UI Features
   * Implement features to allow different list of skills to be selected , to help condense shown information
   * Have relevant art be displayed on particular choices
2. Advanced character display
   * Have information displayed in a way that is easy to read and printable (possibly using official character sheets)

### **VI. Project Development**

1. Node.js Web Server
   * Set up a Node.js server using Express.js to serve the application.
   * Implement at least one route that the app uses (e.g., /weather, /emissions).
2. Database Interaction
   * (Optional) Use MongoDB to store and retrieve weather and CO2 emissions data.
3. JavaScript Framework
   * (Optional) Develop the project using React for enhanced interactivity and state management.

### **VII. Review Process**

1. Internal Review
   * Test calculations for accuracy
   * Ensure everything remains readable despite changing backgrounds
   * Make sure data is successfully saved at each step of creation and can be reloaded as necessary
2. External Feedback
   * Present the project to peers or mentors for feedback.
   * Incorporate feedback to refine and polish the application.

### **VIII. Documentation and Final Submission**

1. Code Annotation and Documentation
   * Annotate code with clear comments and explanations.
   * Write a comprehensive README.md file covering:
     + Project overview and objectives.
     + Setup and installation instructions.
     + Usage guidelines and feature descriptions.
     + Data sources and API integration details.
2. Final Submission
   * Ensure the project is fully functional and well-documented.
   * Prepare the project for final presentation or submission to evaluators.