Design Plan Document

Charles Bostwick, Jade Pearl, Ada Truong, Robyn Cohen, and Holland Brawner

Department of Computer Science, University of Maryland Global Campus

Professor Shanna Nevarez

April 9, 2024

Introduction

This document outlines the architectural and design specifics of the Web Game Devs project, an innovative educational platform crafted to enrich programming knowledge through interactive learning methods using a Hangman Game focus on programming terms embedded in a webpage hosted on GitHub. The team for this project consists of Charles Bostwick, Jade Pearl, Ada Truong, Robyn Cohen, and Holland Brawner at the University of Maryland Global Campus, under Professor Shanna Nevarez's guidance. This capstone project merges the theoretical principles learned during their studies with practical application. By deploying a "Programming Hangman" game within a responsive web environment, the project endeavors to create an immersive educational experience, leveraging the strengths of both web technologies and game development to captivate and educate users.

System Architecture

The system is architected using a microservices-based approach, separating the web front end from the game logic developed in Godot 4.2. The front end utilizes HTML, CSS, and JavaScript to design responsively. At the same time, the backend game logic is handled through GDScript in Godot.

- **Front-End**:
  - **Technologies**: HTML5, CSS3, and JavaScript for creating a responsive design.
  - **Frameworks/Libraries**: Bootstrap for responsive layouts, JQuery for simplified DOM manipulation.
- **Backend (Game Development)**:
  - **Engine**: Godot 4.2, a powerful open-source game engine that facilitates rapid development and prototyping.
  - **Language**: GDScript, a Python-like scripting language tailored for the Godot Engine, is used for game logic.
- **Hosting/Deployment**:
  - Deployed on GitHub Pages, allowing static site hosting directly from a GitHub repository.
  - CI/CD processes implemented via GitHub Actions to automate deployment workflows and ensure a consistent and error-free deployment process.

Part 1: Hangman Game Design

System Architecture

It utilizes Godot 4.2 for its robust, open-source capabilities, which allow for rapid prototyping and development. GDScript, a Python-like language, was chosen for its simplicity and effectiveness in implementing in-game logic. The game component of the Web Game Devs project, developed in the Godot Engine, serves as the centerpiece of our educational platform. This section delves into the game's

development specifics, from architectural patterns to the pseudocode that breathes life into the "Programming Hangman" experience.

System Architecture: Godot 4.2 is utilized for its robust, open-source capabilities, allowing rapid prototyping and development. GDScript, a Python-like language, was chosen for its simplicity and effectiveness in implementing in-game logic.

Design Patterns and Principles

- **MVC (Model-View-Controller)**: Adapted to structure game logic, UI, and input handling, facilitating code maintenance and scalability.
- **Singleton Pattern**: Ensures global access to crucial game states and configurations, enhancing consistency across the game experience.
- **Observer Pattern**: Critical for real-time event handling, responding dynamically to user interactions and game state changes.

Data Structures and Storage

Game vocabulary and definitions are dynamically managed within a JSON file or natively within the Godot script system, supporting seamless updates and expansions to the game's content.

Game Logic Implementation

We present the following pseudocode to demonstrate the application of the design above patterns and principles in developing the Hangman game. This code snippet encapsulates the essential aspects of game initialization, user interaction, and dynamic state updates. It serves as a blueprint for the game's architecture, highlighting the seamless integration of game mechanics and user interface management.

Pseudocode

```
extends Node2D

# Global variables for the game.
var word_list = ["hangman", "apple", "banana", "orange"]  # The list of possible words to guess.
var current_word = ""  # The word currently being guessed.
var display_word = []  # The representation of the word being displayed to the player.
```

```
var attempts_remaining = 6  # The number of incorrect guesses before game over.
var max_attempts = 6  # Maximum attempts allowed.

# Initializes the random number generator to ensure different outcomes in each game session.
func _init():
    randomize()

# Called when the node enters the scene tree for the first time.
func _ready():
    start_game()

# Starts a new game by selecting a random word and resetting game variables.
func start_game():
    current_word = word_list[randi() % word_list.size()]  # Selects a random word.
    print("Word to guess: ", current_word)

    display_word = ["_"] * len(current_word)  # Initializes display_word with underscores.
    attempts_remaining = max_attempts  # Resets the attempts to maximum.
    update_display()  # Updates the game display to show the initial game state.

# Updates the game's display with the current state of the word and attempts left.
func update_display():
    $WordLabel.text = "Word: " + " ".join(display_word)  # Updates the displayed word.
    $AttemptsLabel.text = "Attempts left: " + str(attempts_remaining)  # Updates the attempts left.

    # Optionally update the hangman figure.
    update_hangman_figure(attempts_remaining)

# Processes the player's letter guess and updates the game state accordingly.
func guess_letter(letter):
    if letter in current_word:
        # Reveals the guessed letter in the displayed word if it exists.
        for i in range(len(current_word)):
            if current_word[i] == letter:
                display_word[i] = letter
        # Checks if the game has been won (no more underscores).
        if not "_" in display_word:
            game_won()
    else:
        # Decreases the attempt count on incorrect guess.
        attempts_remaining -= 1
        if attempts_remaining == 0:
            game_lost()

    # Updates the display after each guess.
    update_display()

# Placeholder for updating the visual representation of the hangman.
```

```
func update_hangman_figure(attempts):
    # This function should update the game's visual feedback based on the number of attempts left.
    pass

# Called when the player has correctly guessed the word.
func game_won():
    print("Congratulations! You've guessed the word: ", current_word)
    # Restarts the game for another round.
    restart_game()

# Called when the player has run out of attempts without guessing the word.
func game_lost():
    print("Game over! The word was: ", current_word)
    # Restarts the game for another round.
    restart_game()

# Resets the game variables and starts a new game.
func restart_game():
    # This function can be expanded to include a prompt for restarting or ending the game.
    start_game()

# Processes player input, converting keypresses into letter guesses.
func _input(event):
    if event is InputEventKey and event.pressed and not event.echo:
        # Converts the keypress event to a lowercase letter string.
        var letter = OS.get_scancode_string(event.scancode).to_lower()
        if 'a' <= letter and letter <= 'z':
            # Ensures the letter hasn't already been guessed before processing the guess.
            if letter not in display_word:
                guess_letter(letter)
```

This pseudocode exemplifies the MVC pattern in action, with clear distinctions/boundaries between game logic (model), display elements (view), and input handling (controller). The Singleton pattern is embodied in the global management of the game state, ensuring consistency and accessibility.

Interfaces:

- **IGameplay**: Facilitates core game functionalities, underpinning the game's interactive features.
- **IUserInterface**: Governs the updates and management of the game's UI in response to gameplay.
- Add additional functions and such from the pseudo code down here to expand.

Input/Output Formats:

- Engages users through intuitive input mechanisms and provides immediate, informative feedback, enriching the learning experience.

Part 2: Website Design

Beyond the game, the Web Game Devs project features a comprehensive website that supplements the "Programming Hangman" game with additional educational resources. This section outlines the website's technical approach and functionalities. The website's responsive and accessible platform uses HTML5, CSS3, and JavaScript.

System Architecture: The website's responsive and accessible platform uses HTML5 and CSS3, and *may* use JavaScript to enable simple animations.

Features and Functionality:

- Seamlessly integrates the Godot-compiled game, offering direct access within the web environment.
- Delivers a rich repository of tutorials, glossaries, and quizzes, enriching users' programming knowledge.

Web Front-End Design and User Interaction

It prioritizes navigation and accessibility, ensuring content is easily discoverable and usable for all users. It employs a responsive design strategy to accommodate various devices and screen sizes.

1. **homepage.html**

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Homepage</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <link rel="stylesheet" href="stylesheet.css">
    </head>
    <body>
        <div class="header" style="margin-left: 90%;">
            <p>UMGC CMSC 495</p>
        </div>
        <div class="homepage">
            <h1>Hangman</h1>
            <p>and other games</p>
            <form action="game.html">
                <input type="submit" value="Start New Game"/>
            </form>
            <h4><a href="team.html">Meet the Team</a></h4>
        </div>
    </body>
```

```
</html>
```

2. **game.html**

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Hangman</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <link rel="stylesheet" href="stylesheet.css">
    </head>
    <body>
        <div class="back button">
            <a href="homepage.html"><img src="backbutton.png" alt="Back
button"
                style="width:200px;height:50px;margin-right:75%;"></a>
        </div>
        <div class="header">
            <p>UMGC CMSC 495</p>
        </div>
        <h1>Game here</h1>
    </body>
</html>
```

3. **team.html**

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Meet the Team</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <link rel="stylesheet" href="stylesheet.css">
    </head>
    <body>
        <div class="back button">
            <a href="homepage.html"><img src="backbutton.png" alt="Back
button"
                style="width:200px;height:50px;margin-right:75%;"></a>
```

```
        </div>
        <div class="header">
            <p>UMGC CMSC 495</p>
        </div>
        <h1 style="text-align:center;">Meet the Team</h1>
        <img src="profile.jpeg" alt="Profile picture"
style="width:200px;height:200px;">
        <div class="profile">
            <h2>Charles Boswick | Title</h2>
            <p>bio here...</p>
        </div>
        <img src="profile.jpeg" alt="Profile picture"
style="width:200px;height:200px;">
        <div class="profile">
            <h2>Jade Pearl | Title</h2>
            <p>bio here...</p>
        </div>
        <img src="profile.jpeg" alt="Profile picture"
style="width:200px;height:200px;">
        <div class="profile">
            <h2>Ada Truong | Title</h2>
            <p>bio here...</p>
        </div>
        <img src="profile.jpeg" alt="Profile picture"
style="width:200px;height:200px;">
        <div class="profile">
            <h2>Holland Brawner | Title</h2>
            <p>bio here...</p>
        </div>
        <img src="profile.jpeg" alt="Profile picture"
style="width:200px;height:200px;">
        <div class="profile">
            <h2>Robyn Cohen | Title</h2>
            <p>bio here...</p>
        </div>
    </body>
</html>
```

4. **stylesheet.css**

```
.header {
    position: relative;
```

```
    float: right;

}

.homepage {
    position: relative;
    top: 200px;
    text-align: center;
}

.back button {
    position: relative;
    float: left;
}

img {
    float: left;
    margin-right: 10px;
}

.profile {
    height: 300px;
}
```

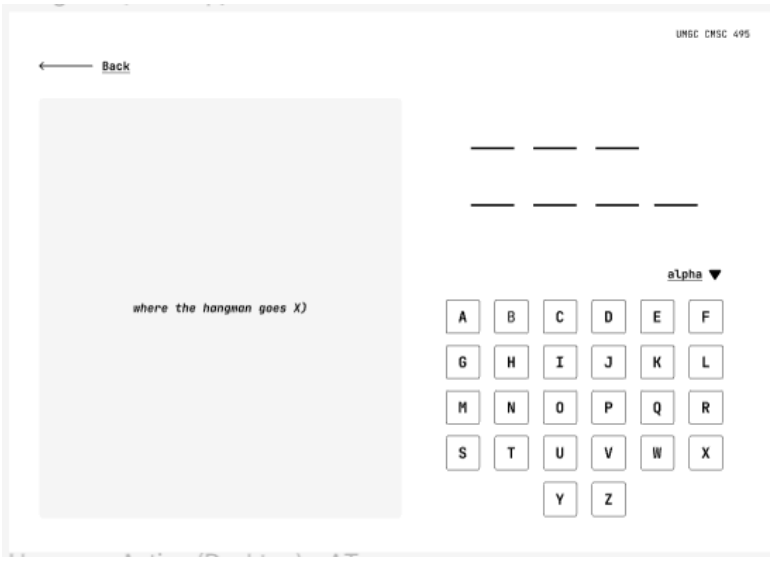
Design Mockups and UI Illustrations

Our vision extends to the website design, aiming to create a cohesive experience that complements the educational game. Below are mockups of the website, with frames for desktop, tablet, and mobile. Note that all frames are currently in greyscale, and will eventually be converted to a proposed UMGC color scheme (described below). In addition, gray content boxes and lorem ipsum text are placeholders for various visuals/text, including the actual hangman and team member profile pictures. Finally, not all frames are pictured, such as frames with components that require no/minimal positional readjustments regardless of the screen size.

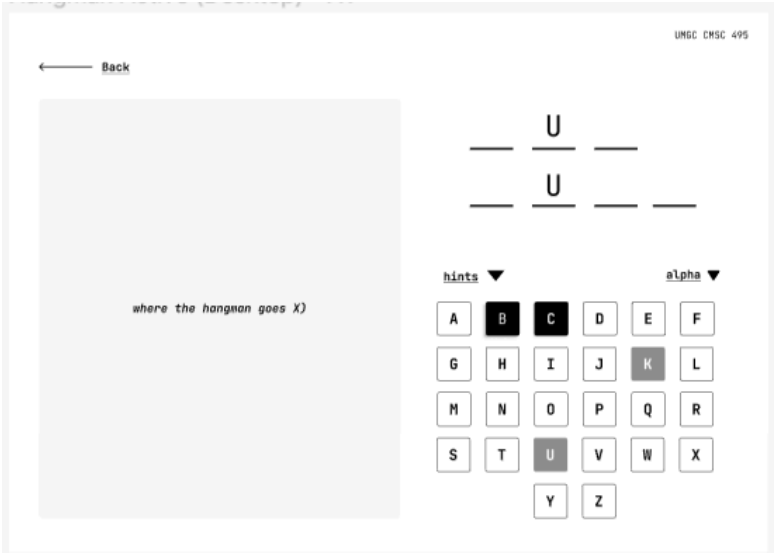
Desktop Frames

- Home: 

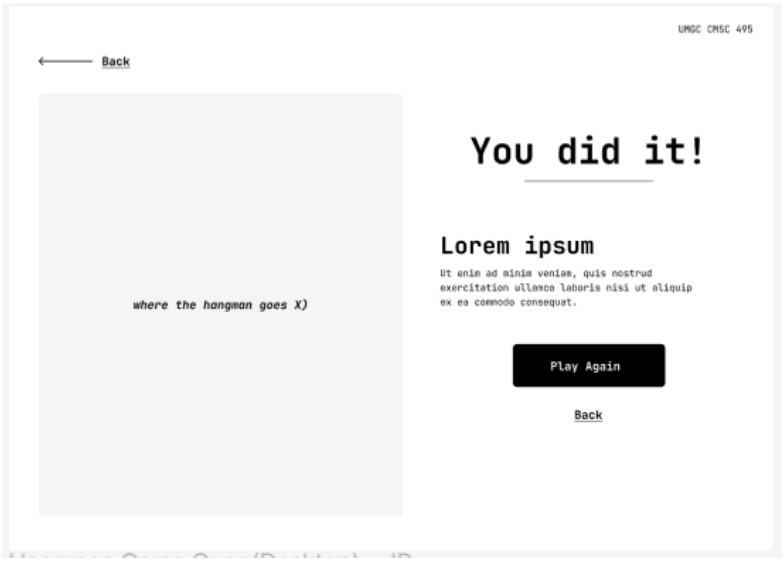
- Hangman Game (upon loading, inactive):



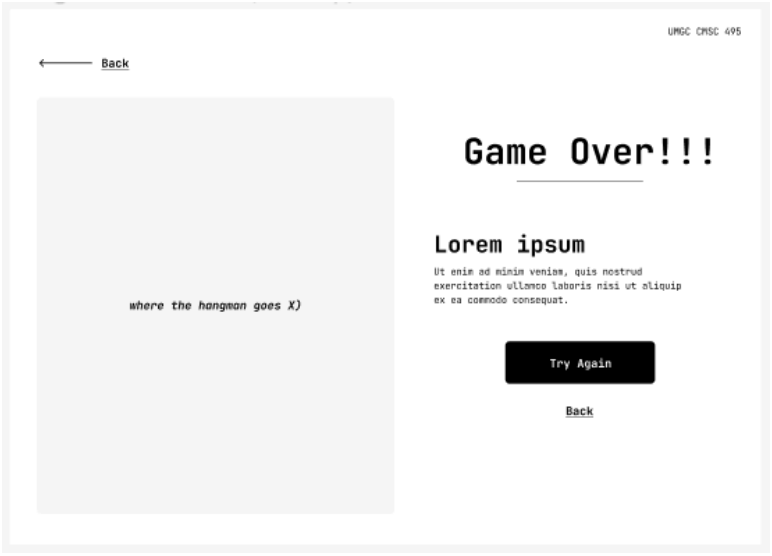
- Hangman Game (mid-game, various button states):



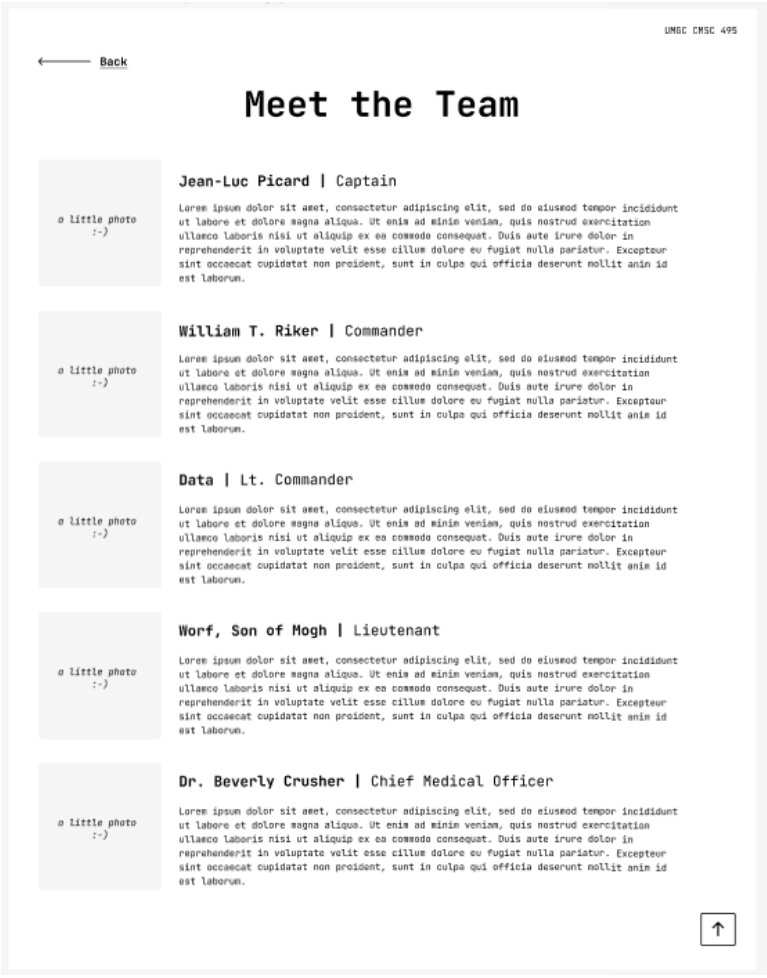
  - Black background letter buttons with a drop shadow indicate an active hover state. Black background letter buttons without a drop shadow indicate an active click state. Gray background letter buttons indicate that the letter has already been chosen.
- Hangman Game (end, user successfully guessed the word/phrase):



- Hangman Game (end, user failed to guess the word/phrase):

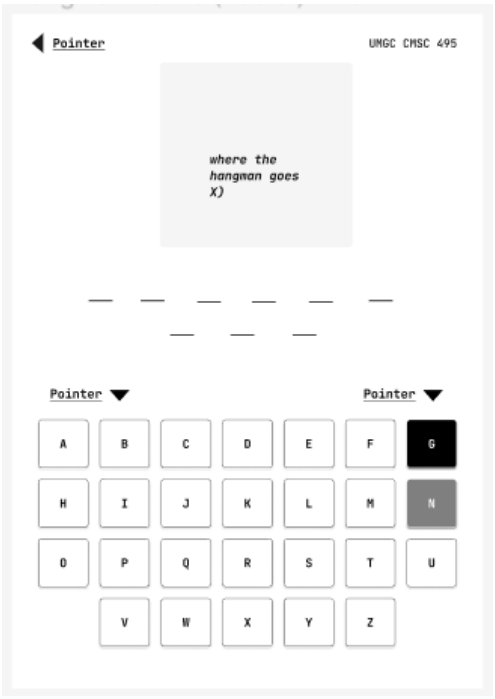


- Meet the Team:

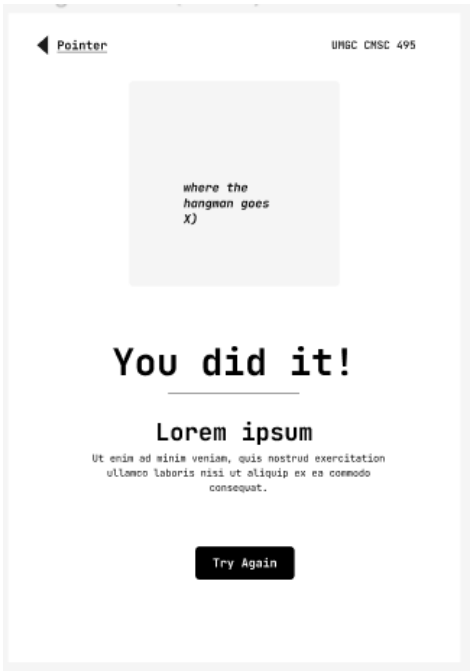


Tablet Frames

- Hangman Game (mid-game, active states):



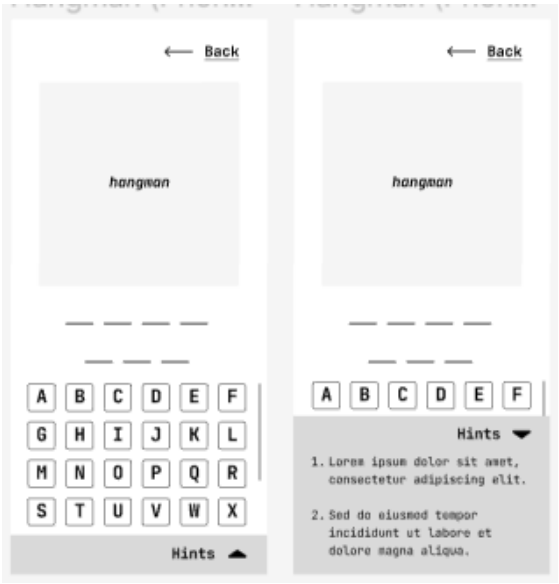
- Hangman Game (end):



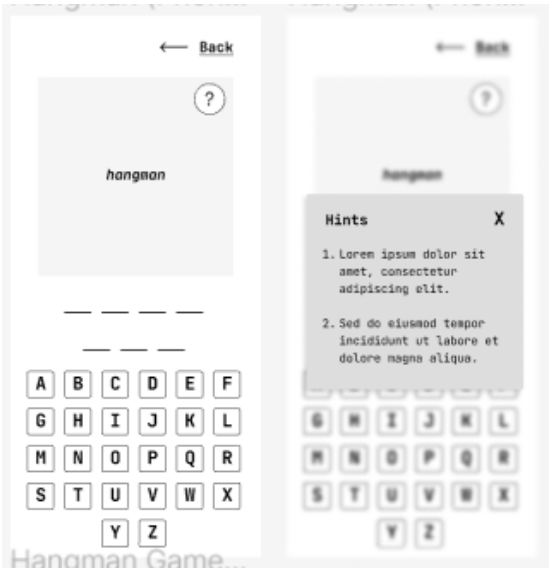
- Meet the Team:

Mobile Frames

- Hangman Game (option 1: hint expands and collapses):

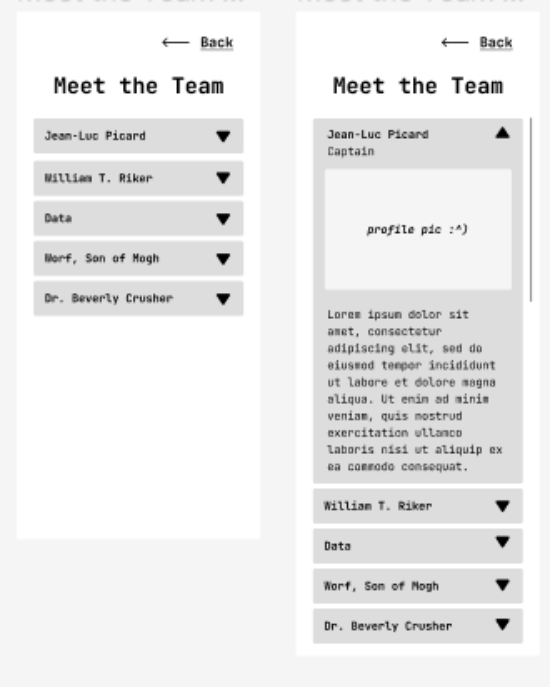


- Hangman Game (option 2: hint opens and closes):



- Hangman Game (end):

- Meet the Team (all profiles collapsed, one profile opened):



Proposed Color Scheme

The proposed colorway will align with UMGC school colors (black, yellow, and red). Most elements will be in grayscale, with either white text on a black/gray background or black text on a white background. Given potential issues with contrast (per the WCAG 2 Contrast Ratio requirements), yellow and red will be limited to decorative accents, and will not hold any semantic meaning.

Development and Deployment:

We plan to utilize GitHub for collaborative development and employ CI/CD practices through GitHub Actions for efficient deployment.

Conclusion

The Web Game Devs project represents a holistic approach to interactive learning, combining game development's engaging nature with a dedicated website's broad educational potential. By detailing both components separately, this document highlights the project's dual focus: to entertain and educate users in programming. Through continuous development and iterative improvements, the project aims to be a beacon of innovative educational solutions in the digital age.

References

GitHub. (n.d.). *GitHub Learning Lab*. Retrieved from https://github.com/apps/github-learning-lab

GitHub. (n.d.). *GitHub Pages*. Retrieved from https://pages.github.com/

GitHub. (n.d.). *GitHub Pages Documentation*. Retrieved from https://docs.github.com/en/pages

GitHub. (n.d.). *Introduction to GitHub*. Retrieved from https://skills.github.com/

Godot Engine. (n.d.). *Documentation*. Retrieved from https://docs.godotengine.org/en/stable/index.html

Institute of Electrical and Electronics Engineers. (n.d.). *278253*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/document/278253

Game Development Document link (GDD): https://github.com/AwaywithCharles/CMSC495-Capstone/blob/main/documentation/GDD.txt

Minimum Viable Product (MVP): https://github.com/AwaywithCharles/CMSC495-Capstone/blob/main/documentation/MVP.txt

Software Requirements Specifications (SRS): https://github.com/AwaywithCharles/CMSC495-Capstone/blob/main/documentation/SRS.txt

Project GitHub Repository: https://github.com/AwaywithCharles/CMSC495-Capstone

WebAIM. (n.d.). *Contrast and color accessibility*. https://webaim.org/articles/contrast/