Alphabet Soup Game Report

Project for the course of Web and Real Time Communication Systems

Professor: Simon Pietro Romano

Written By: Marta Custódio Rosa

Matricola N000124825

Held in: 2024/2025

Abstract

This project consists of the development of a simple, retro-styled word search puzzle game called *Alphabet Soup*. It has been written with a JavaScript back-end for logic robustness and HTML/CSS and also JavaScript for a dynamic front-end interface. The goal was to provide a fun and nostalgic user experience with its retro styling and basic user interaction. This report outlines the development process, tools used, implementation details, and challenges encountered.

1 Introduction

The idea behind the development of an alphabet soup game was to be able to implement dynamic and interesting features in my project while still keeping it at lower difficulty level, as I am not a computer engineering student. This way, I was able to develop a fully interactive game, like it is intended in this course, and learn a whole lot of new skills without getting too overwhelmed. The main objectives of my project include:

- Providing a simple, nostalgic front-end design.
- Implementing game logic to verify word selection, track the player's progress and show solution if intended.
- Allowing player to customize some game settings such as grid size and word input.

2 Technologies and Tools Used

- **Languages**: JavaScript (back-end and part of front-end), HTML (front-end).
- **Tools**: VS Code (development), GitHub (version control).
- **Audio and Styling**: Retro music and custom CSS for styling.

3 Implementation

The implementation of the "Alphabet Soup Game" involved building a web-based application that allows players to interact with a word search puzzle game. The project contains the following components:

3.1 Codebase Overview

The project is structured as follows:

- **HTML Pages:** There are four HTML files: welcome.html, setup.html, game.html, and victory.html. Each page serves a simple but specific purpose in the game flow, that purpose is noticeable in the file name.
- **Front-end JavaScript:** The two JavaScript files, gameLogic.js and setupLogic.js, can be somewhat complex because they manage the front-end interaction and functionality for the game and game setup process respectively.

- **Back-end JavaScript Server:** A server.js handles routing and API calls, serving HTML pages and generating the game grid.
- **CSS File:** A single style.css file is used for all the retro styling with a black and neon green being the main colors.
- **Images and Audio:**
  - Images folder that includes both GIFs used.
  - Audio folder that contains the background audio loops and coin sound effect to further provide a retro arcade feel.

3.2 Detailed Explanation of the Codebase

HTML Pages: The HTML files define the structure of the game:

- **welcome.html:** Serves as the home page with a welcome message, alphabet GIF and a button to start the game setup.
- **setup.html:** Allows the user to input game configurations, choosing the grid size from 3 button options (7×7, 10×10 or 12×12), choosing word number (between 1 and 5) and finally inputing the actual words that the user wants to play with.
- **game.html:** Displays the game grid and provides two interactive buttons, one for verifying if the selected (highlighted) word is one of the words inputed and the other for peeking at the solution in the grid.
- **victory.html:** Congratulates the user upon completing the game.

Front-end JavaScript:

- **setupLogic.js:** Handles user inputs for grid size and words. Ensures all data is validated before passing it to the game page. The number of words to play with is conditioned to be between 1 and 5. The inputed words are conditioned to only contain letters and need to be smaller than the grid size.
- **gameLogic.js:** Manages game state, including highlighting selected letters in the grid, handling the 'verify word' button logic (checking if the position selected and the text that those cells contain match exactly the position randomly generated and inputed text) and solution button logic (highlighting cells that contain the solution).

Back-end JavaScript Server: A server.js manages server-side functions like starting the server on port 8181, routing HTML pages and sending data to the player, as well as handling all puzzle grid generation (placing input words in random coordinates and directions and filling empty grid spaces with random letters) and its conditions (ensuring the words fit within the grid and don't overlap unless they have a common letter). Server uses local storage to preserve player's chosen settings across game states.

4 Challenges and Solutions

- **Challenge 1: Ensuring verifying the selected words and solution button worked properly .**
  Solution: The initial game didn't contain a position mapping of the words placed in the grid, so later was developed a simple return on the grid generating function that included the word and the exact coordinates of each of its letters. After this implementation, the verify word and solution button functions were easily corrected.
- **Challenge 2: Maintaining grid aesthetics across dynamic buttons.**
  Solution: Sometimes the use o a custom CSS file made the code harder to implement, even though it looked "cleaner". So I decided to keep some styling in the front-end JavaScript code to make the dynamic buttons work easily, namely the solution button and verify word button, while maintaining grid aesthetics.

5 Conclusion

In this project I was able to successfully achieve its main goals which are providing a functional and engaging word search puzzle game with some customizable settings and featuring a nostalgic retro style. Some future improvements could include:

- Adding mobile responsiveness.
- Introducing multiplayer modes, players compete for best time.
- Enhancing word list by generating random words from a database.

Appendix

GitHub Repository: `https://github.com/MartaCRosa/webrtc-project`

`https://github.com/WebRTC-Projects-Unina/AlphabetSoupGame`

Credits

- **Audio:**
  - Background music 1: "Retro Game Music" by moodmode, retrieved from `https://pixabay.com/music/`.
  - Background music 2: "That Game Arcade Medium" by moodmode, retrieved from `https://pixabay.com/music/`.
  - Button sound effect: "Retro Coin 1" by Driken5482 retrieved from `https://pixabay.com/sound-effects/`.
- **Images:**
  - Welcome GIF: `https://www.pinterest.com`.
  - Victory GIF: `https://tenor.com/search/victory-gifs`.