Final Report

Wordle Solver

December 13, 2022

Version 1.0

Team Chihuahua
Ben Allen
Kristina Williams
Rashid Imtiaz
Annjanie Kearse

Table of Contents

Overview

This final report is dedicated to the last 8 weeks of development over the Wordle Solver. This app was brainstormed in the first week of this project, and the following weeks were dedicated to creating test plans, a user manual, the project design, three separate phases for development, and the official deployment of the app.

During the weeks of development - feedback was provided and applied to the project to ensure the documentation was meeting all requirements and properly included all the details necessary for the app. The app has been under several reviews and edits to meet its full functional use.

The final report will include the purpose of the Wordle Solver, the scope of the project, the project timeline, team members and their roles and contributions, the general description of the application and what it requires, the specifications, a user manual, the test plan and full test report, the design of the application, and a history of the development.

1. Introduction

1.1 Purpose

The purpose of this project is to create an extension website of the popular web-based game app “Wordle.” Wordle is a 5-letter word guessing game that allows users 6 tries to guess the word of the day. Wordle Solver will be a web-based application that serves as an extension of the game to help users solve the word of the day.

1.2 Scope

The project plan outlines the means to coordinate schedules, allocate resources, and monitor development progress. The Wordle Solver application is set on an 8-week timeline. The project should be successfully completed by accomplishing each task listed below.

Task	Category	Description
Project Plan	Analysis	Developing the plan for the overall project. Defining roles, goals, purpose, and requirements
Test Plan	Analysis/Code	Create test data files and begin required inputs and expected outputs
Project Design	Code	Define classes, methods, fields, and interfaces. Begin collecting data files to be implemented into the program.
Phase 1 Source	Code	Begin developing backend functionality.
Phase 2 Source	Code	Continue coding - Begin developing user facing functionality.
Phase 3 Source	Code	Continue coding - Finalize user interface design and functionality
Deploy	Monitor	Deploy the application and monitor

1.3 Project Schedule

Task	Duration	Begin	End	Persons Responsible
Project Plan	7	10/26	11/1	Ben Allen, Kristina Williams, Rashid Imtiaz & Annjanie Kearse
User Guide	7	11/2	11/8	Ben Allen, Kristina Williams, Rashid Imtiaz & Annjanie Kearse
Test Plan	7	11/2	11/8	Ben Allen, Kristina Williams, Rashid Imtiaz & Annjanie Kearse
Project Design	7	11/9	11/15	Ben Allen, Kristina Williams, Rashid Imtiaz & Annjanie Kearse
Phase 1 - Backend functionality/ Java + Spring Boot + Gradle "Hello World" app	7	11/16	11/22	Ben Allen, Kristina Williams, Rashid Imtiaz & Annjanie Kearse
Phase 2 - Backend/frontend interconnectivity	9	11/21	11/29	Ben Allen, Kristina Williams, Rashid Imtiaz & Annjanie Kearse
Phase 3 - User interface design	9	11/28	12/6	Ben Allen, Kristina Williams, Rashid Imtiaz & Annjanie Kearse
Final Deliverable	7	12/7	12/13	Ben Allen, Kristina Williams, Rashid Imtiaz & Annjanie Kearse

1.4 Team Member Roles

Name	Role
Ben Allen	***Design Lead*** Leads GUI design. Assists with overall project development and documentation. Collaborates to establish interconnectivity between all application components.
Kristina Williams	***Documentation Lead*** Leads the gathering of and collaborates on all project documentation for inclusion in the final report. Assists with overall application development and testing.
Rashid Imtiaz	***Development Lead*** Leads and collaborates with the team on the overall application

	development and testing. Collaborates with the design lead to ensure interconnectivity between all application components.
Annjanie Kearse	***Team Lead*** Collaborates with team members to ensure deliverables are completed according to specifications and within the given time frame. Organizes team calls, deliverables, and communications with the professor. Assists with overall application development, testing, and documentation.

1.5 Resources

The table below contains links to resources used during development.

Resource	**Description**
Wordle (nytimes.com)	Wordle is a web-based word game created and developed by Josh Wardle, and owned and published by The New York Times Company since 2022. Wordle Solver is built to solve Wordle games.
cfreshman/wordle-nyt-answers-alphabetical.txt (github.com)	GitHub repository containing Wordle answers from source code in alphabetical order. Current as of 09/02/22.
Diagrams.net	A web application used to create diagrams, process charts, and interface mockups.
Heroku	Heroku is a cloud platform as a service (PaaS) supporting several programming languages.

2. General Description

2.1 Product Perspective

The product is a web based application that provides a simple mechanism for users to solve the Wordle guessing game.

2.2 Product Features

The application features a user interface, accessible via web browser, where the user may input the results of a guess made in the Wordle game by inputting each character into the corresponding position (green, yellow, gray). A solve button will calculate the next best guess based on the user's input. The web application will track each word the user inputs to filter suggested guesses.

2.3 User Characteristics

The extent of the application considers that users have the basic knowledge of accessing and operating a web browser and are considered familiar with the Wordle word guessing game.

2.5 Operating Environment

The website front-end will be coded in Javascript, CSS, and HTML. The back-end will be run as a Java+Spring Boot+Gradle application hosted on Heroku. Heroku is a cloud platform as a service supporting several programming languages.

2.6 Constraints

The application will be developed primarily in JavaScript, requiring users to use a compatible web browser. All modern web browsers should be compatible.

2.7 Assumptions and Dependencies

The algorithm used to suggest word guesses to the user assumes that the word list referenced in Section 1.5 is complete and accurate. It is also assumed that users will correctly input the result of their guesses on the main Wordle app.

3. System Requirements

3.1 Functional Requirements

The main interface will be required to accept user input in a 3-row, 5-column grid-like structure where each cell in the grid represents a letter position of the guessed word. The top row represents correctly guessed letters. The middle row represents valid letter guesses with an invalid position. The bottom row invalid guesses. The application will ingest the input, run it through a word selection algorithm, and display recommendations back to the user.

In addition to the input grid, 2 text boxes flanking the grid will display output. The text box on the left displays suggested guesses. The text box on the right displays words the user guessed previously. The words in each text box will be displayed in a vertical list. The application will be required to update the words displayed in each list after user interaction.

A *clear* button under the word input grid will allow the user to reset the solver. An *update* button under the input grid must be clicked by the user after inputting a new guess. When clicked, the new guess will be parsed with the results displayed on-screen. Cookies will be used to track input data and results. Clicking the clear button reset the cookies managing the user's session. See the image displayed below in Section 4.1 for an initial prototype of the GUI.

There are minimal graphics requirements. Input and output can be managed with text boxes and buttons. Placement of these objects can be managed with CSS. Cosmetic graphics, such as an application logo, will be included in the application file package.

Source code version management will be performed using Heroku Git / CLI.

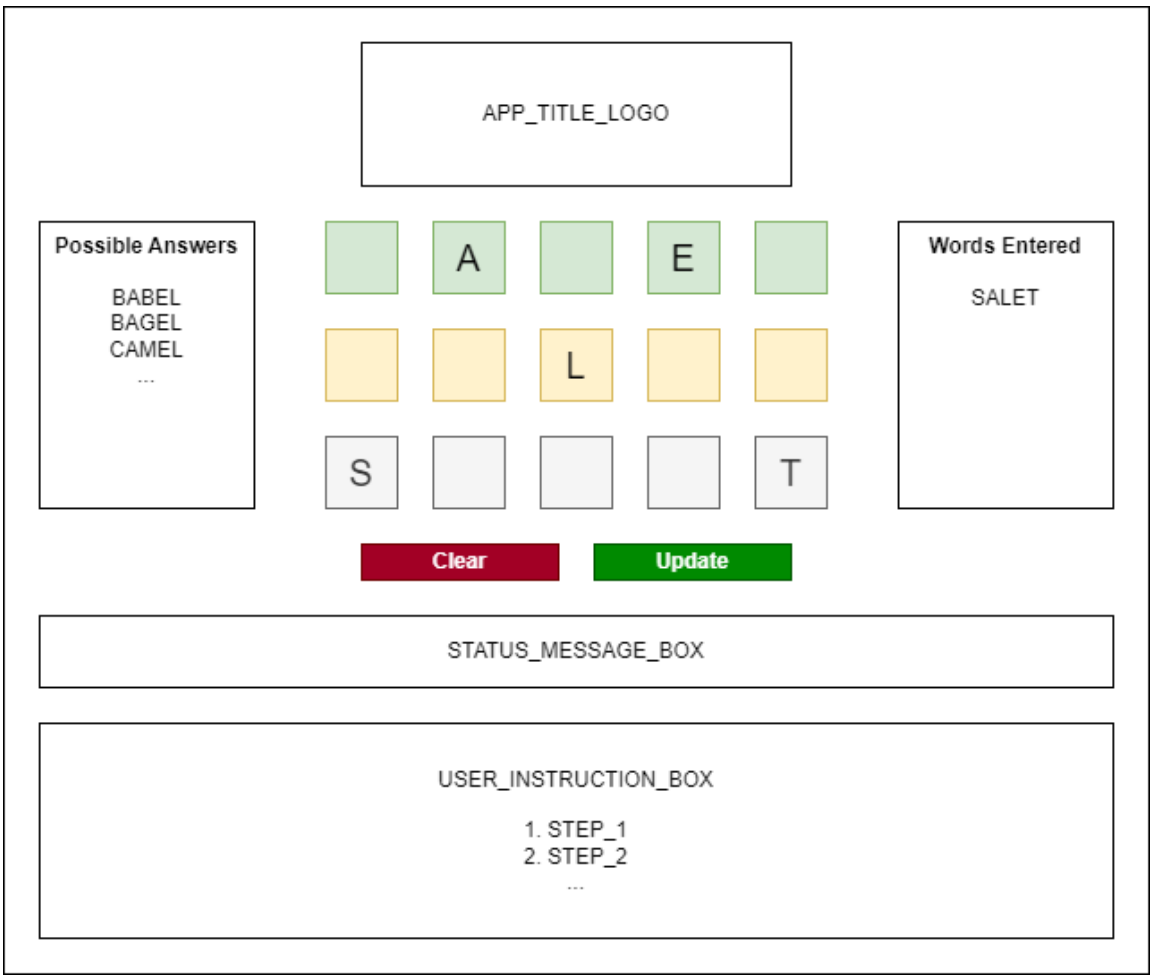
4.External Interface Requirements

4.1 User Interfaces

The image below displays a mockup of the application GUI, which will be built using the following:

- Front-end software: CSS, and HTML
- Back-end software: Java, Spring Boot + Gradle

The original UI prototype is displayed below.



4.2 Hardware Interfaces

No hardware interfaces will be required.

4.3 Communications Interfaces

Wordle Solver will exist as a web application that uses traditional HTTP communications. No specialized communication interfaces will be required.

4.4 Software Interfaces

4.4.1 Front-end

The frontend combines HTML and CSS to develop the user facing interface.

4.4.2 Back-end

The backend of the application will be developed using
Java and Spring Boot + Gradle to develop interconnectivity between the frontend and backend components.

4.4.3 Hosting

The Wordle Solver will be hosted on Heroku, providing users with application access via a web browser. Domain name registration and TLS certificates for HTTPS will be managed by Heroku.

5. Specifications

5.1 Requirements Specifications

5.1.1 Solve Wordle

- The user interface shall display all Wordle character guess boxes that may be configured.
- The user interface shall display all possible Wordle answers according to user input.
- The user interface shall display all words entered by the user.
- The user interface shall enable users to solve the Wordle based on user input.
- The user interface shall display the solved Wordle according to user input.

5.1.2 Clear User Input

- The user interface shall enable the user to clear all user input via a clear button.

5.1.3 Use Case

Use Case ID	1
Use Case Name	Solve Wordle
Actors	User
Description	The user will be able to input characters into the UI to solve the Wordle based on their input.
Normal Flow	1. User navigates to the Worlde Solver website 2. The user will input characters into their corresponding positions. 3. The user will click the ‘update’ button. 4. The UI will display a list of possible Wordle answers based on the user’s input.
Alternate Flow	None

5.2 Security Requirements

Users will not be required to make an account to use Wordle Solver. Session and application data can be tracked using cookies. No user data will be stored server-side.

5.3 Software Quality Attributes

The application should have a GUI that is intuitive and requires minimal user instruction. Once a minimum viable product is developed, future updates will only be required when the word list resource referenced in Section 1.5 is updated.

This section will be updated as needed during later development stages to describe quality attributes related to maintainability, adaptability, flexibility, usability, reliability, and portability.

5.4 Other Requirements

No other requirements are necessary.

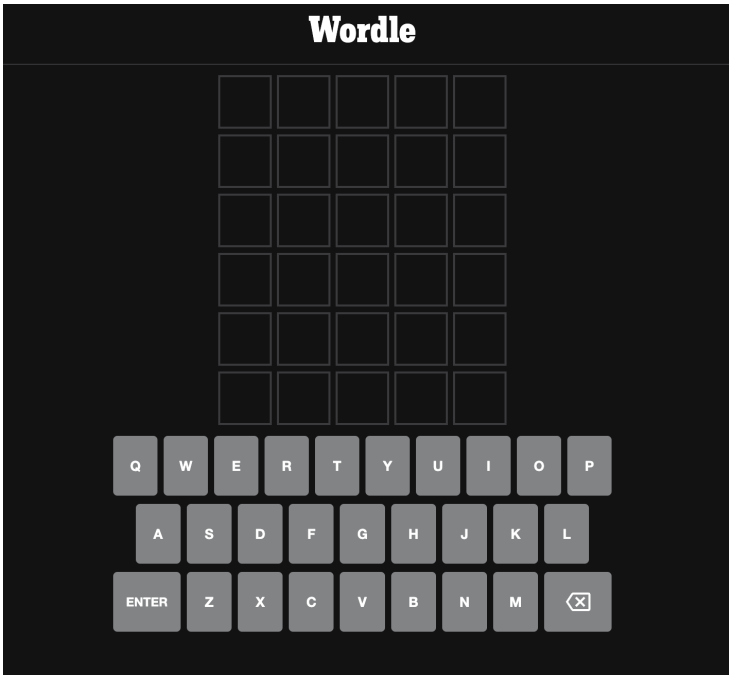
6. User Guide

6.1 App Use Overview

The Wordle Solver is a web-based application that helps users solve the word of the day on another popular web-based application called Wordle by organizing the letters and suggesting possible solutions.

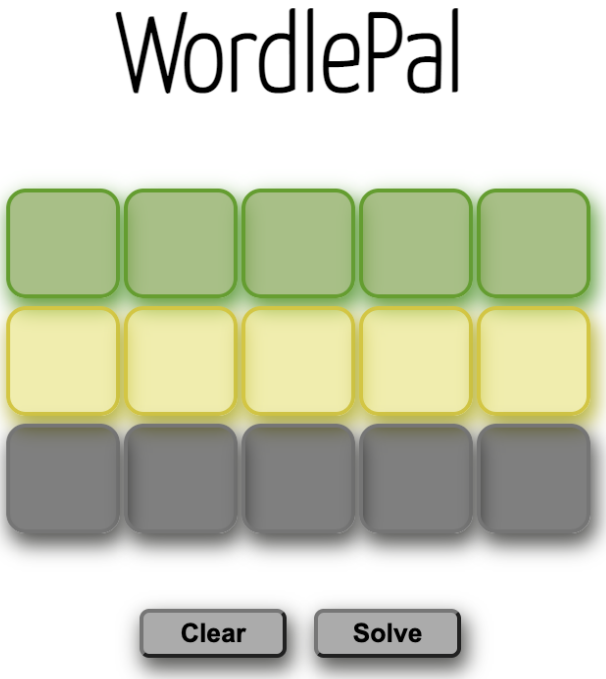
The Wordle Solver will need to be accessed through a web browser. There is no account that needs to be created. Sessions are created each time the browser loads or refreshes. Users should have Wordle opened as well in order to use the application as intended. The authentic Wordle game can be accessed at the link below.

https://www.nytimes.com/games/wordle/index.html



6.2 Getting Started

1. Open up the Wordle Solver by entering this link **https://wordle-pal.herokuapp.com/** while having Wordle open.



2. After entering a word in the Wordle app, input the letters based on the corrective placements (correct letters in correct positions go in green boxes, correct letters in incorrect positions go in yellow boxes, and incorrect letters go in gray boxes) in the Wordle Solver.
3. Hit the solve button to output potentially correct words based on the input.
4. Once finished with the Wordle Solver, exit the browser to end the session.

6.2.1 Setup Considerations

Users will need to access both Wordle and Wordle Solver through a web browser.

6.2.2 User Access Considerations

There is no specified user access to use this web-based application. Users do not create an account or link any accounts. Similar to Wordle - the user simply needs to start a session by entering the website.

6.2.3 Accessing the Application

Users will access the application by entering the link below in a browser.

- https://wordle-pal.herokuapp.com/

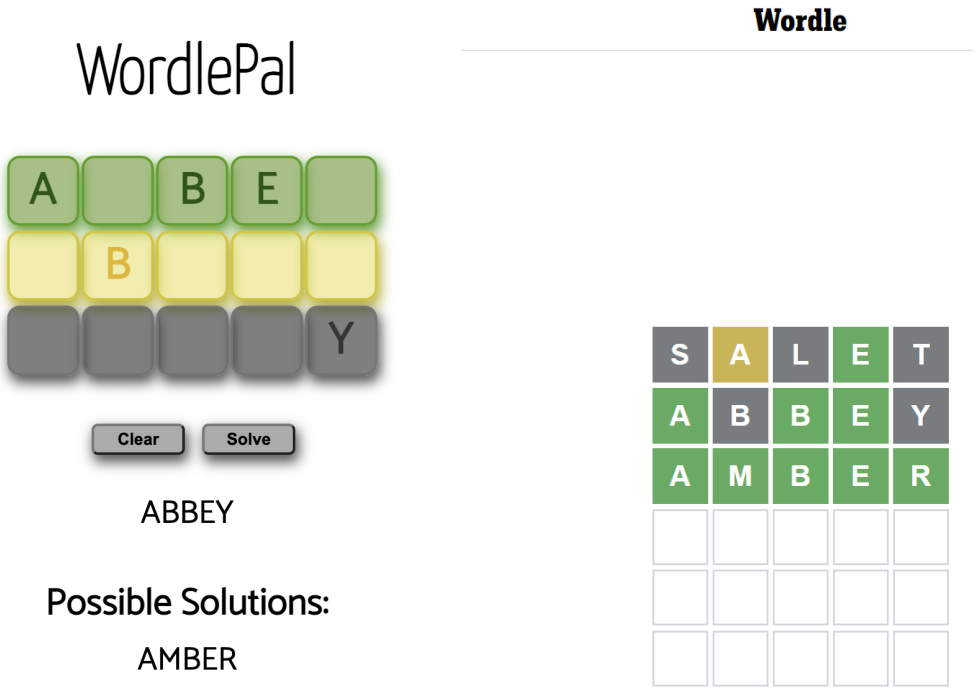
6.2.4 Exiting the Application

Users will simply exit the system by closing out of the browser. A new session will be created any time the website is loaded or refreshed.

6.3 Using the Application

Once users have started a session, the application will display a 3 row, 5 column set of blocks. The user should also open the authentic Wordle game in another browser window or tab. After entering a word for their first guess in Wordle, the user will enter the results of each letter on Wordle Solver in the corrective color block.

For example, if the user enters “ABBEY” as their first guess and the A, B, and E are the correct letters in the correct position, and the word does not contain the letters Y then the input in the Wordle Solve will look similar to the image below.



The user should hit the *Update* button in order for a list of possible answers to be displayed. The user should continue updating their inputs until they have correctly solved the word. If the user wants to clear the inputs they will hit the Clear button. The session will end whenever the user exits out of the browser.

6.3.1 User Interface Elements

Clear

The clear button will be used whenever users want to clear out all inputs entered. Users should only use this button if they want to start the process all over again.

Update

The update button will be used every time the user enters a set of letters in the correct box in the input grid. Once users have entered the letters in the Wordle Solver they need to press the update button in order to refresh the Possible Answers table.

Possible Answers

The possible answers table on the left side of the application will be a list of possible answers that the Wordle puzzle can be based on the input. As long as the letters are entered in the correct positions on the graph - then the answer will be included somewhere on the table.

Words Entered

The words entered in the central input grid will be displayed on the right side of the application. Only enter the words entered on Wordle to help accurately solve the puzzle.

Message Box

The message box area below the Clear and Updates buttons is intended to display status messages, such as an error when invalid input is entered or game reset message.

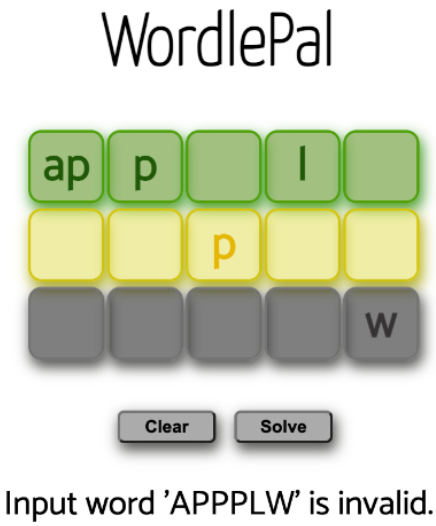
Instruction/Error Box

The instruction box at the bottom of the user interface will display a list of static instructions for using Wordle Solver.

6.4 Troubleshooting & Support

6.4.1 Status Messages

Users will receive an error message whenever they try entering a special character or number stating that they must enter a letter. Users will also receive an error if they do not enter an actual word. Messages will be displayed in the Message Box area below the input grid and buttons.



6.4.2 Support

Wordle Solver is a free product and no official support is provided.

7. Test Plan

7.1 Scope

The application features a user interface, accessible via web browser.

As development progresses, the team will revise the test plan as needed.

7.2 Assumptions

This section lists assumptions that are made specific to this project.

1. The algorithm used to suggest word guesses to the user assumes that the word list included in the Wordle word list repository is complete and accurate.

7.3 Test Approach

Testing will occur at each phase of development to ensure that bugs are fixed before the next phase. The general workflow for testing is displayed below.

1. Test cases will be added, as needed, to verify functionality of features to be developed.
2. Run all identified tests. Newly added test cases must fail, as the functionality has yet to be developed.
3. Write code to implement functionality.
4. Run all tests. New test case should pass, as functionality has been implemented.

7.4 Test Automation

Automated unit tests are part of the development process. There are no automated functional tests planned at this time.

7.5 Test Environment

Tests will be performed using a web browser to simulate the user experience. A development instance of the Wordle Solver may be used for testing. When the product ships, the production instance should be tested.

7.6 Test Cases

The table below displays planned test cases for the Wordle Solver program. Test cases may change during development. Results in the *Test Result* column will be filled in at a later date.

Test	Scenario	Expected Result	Actual Result	Pass/Fail
1	Default app/page load	Browser loads the Wordle Solver web app	Browser loads the Wordle Solver web app	Pass
2	No input, clear button clicked	No change to input or output areas; message about requiring input displayed	No change to input or output areas; message about requiring input displayed	Pass
3	No input, update button clicked	No change to input or output areas; message about requiring input displayed	No change to input or output areas; message about requiring input displayed	Pass
4	Any input, clear button clicked	Input an output areas cleared; message about being reset displayed	Input an output areas cleared; message about being reset displayed	Pass
5	Invalid input (non-alpha), no previous guesses, update button clicked	No change to input or output areas; message about invalid input displayed	No change to input or output areas; message about invalid input displayed	Pass
6	Invalid input (non-alpha), previous valid guesses, update button clicked	No change to input or output areas; state of existing guesses kept; message about invalid input displayed	No change to input or output areas; state of existing guesses kept; message about invalid input displayed	Pass
7	Valid input (alpha), no previous guesses, update button clicked	Input area cleared; valid word displayed in Words Entered area; Possible Answers area updated; message box cleared	Input area cleared; valid word displayed in Words Entered area; Possible Answers area updated; message box cleared	Pass
8	Valid input (alpha), previous valid guesses, update button clicked	Input area cleared; valid word displayed in Words Entered area; Possible Answers area updated; message box cleared	Input area cleared; valid word displayed in Words Entered area; Possible Answers area updated; message box cleared	Pass
9	Valid input (alpha), previous valid guesses, no remaining answers, update button clicked	Input area cleared; no change to output areas; message about no remaining solutions displayed	Input area cleared; no change to output areas; message about no remaining solutions displayed	Pass
10	App/page reloaded by browser with previous user input	Output areas restore previous data; input area cleared; message box cleared	Output areas restore previous data; input area cleared; message box cleared	Pass

7.7 Acceptance Criteria

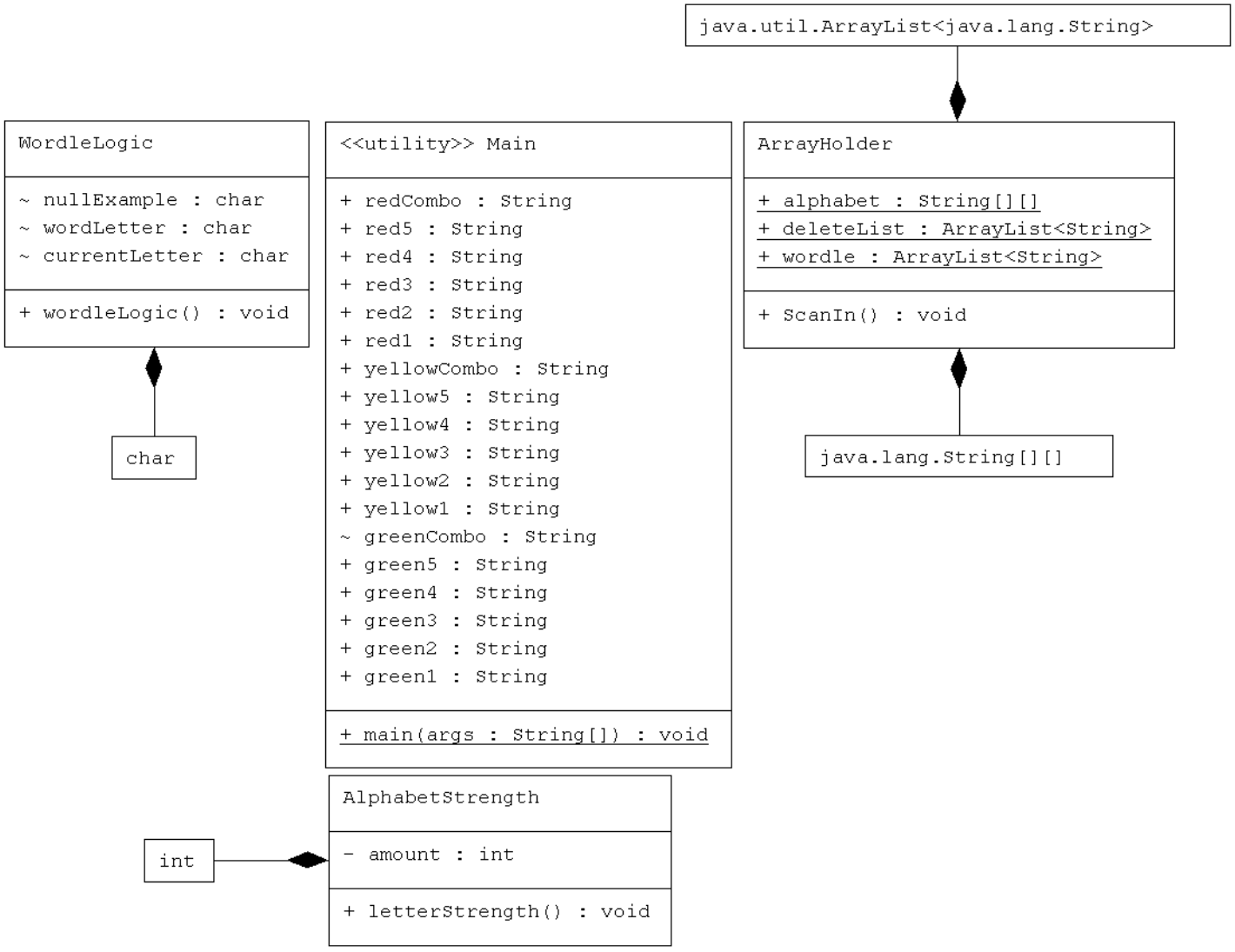
Functional requirements tested and required for acceptance are included in the checklist below.

- ☐ The user interface shall display all Wordle character guess boxes that may be configured.
- ☐ The user interface shall display all possible Wordle answers according to user input.
- ☐ The user interface shall display all words entered by the user.
- ☐ The user interface shall enable users to solve the Wordle based on user input.
- ☐ The user interface shall display the solved Wordle according to user input.
- ☐ The user interface shall enable the user to clear all user input via a clear button.

8. Classes

8.1 Class Design Overview

This section describes various software classes and related logic for the application. The UML diagram below depicts the associations between the application's classes.



8.2 Message Class

The message class is responsible for setting and retrieving messages to be displayed to the user.

8.3 Wordle Pal Application Class

The WordlePalApplicaton class contains the main method and drives Spring application execution. It is responsible for calling other classes in the application, and parsing the Wordle answer list (packaged .txt file) so it's available as an internal resource. This class is responsible for driving the resources that create the user interface, developed with HTML and CSS.

8.4 Main Page Class

This class serves as the web controller for the web application. This class handles web requests and mapping.

8.5 Word Class

This class contains the algorithms and logic for identifying possible solution words. It will interact with all the other classes to successfully run its designated code.

If no user input has been supplied (new game), the class will recommend a list of strategically powerful first-guess words. If user input has been supplied, application logic will run and attempt to generate a list of possible answers. Additional details regarding algorithm logic are discussed in the Logic & Algorithms section below.

9. Data Structures

9.1 Data Structures Overview

The Wordle Solver will implement Arraylists as the primary data structure to store, process, and retrieve user input and possible answers. A high-level outline of use and implementation is provided below.

- Dictionary Arraylist
    - Will store the words found in the Wordle dictionary.

- Letter strength Arraylist
    - Contains each letter alongside the strength of each letter. This will be used to determine the most likely answer.

- Possible Answers Arraylist
    - Contains the list of possible answers generated according to the user's input and application logic.

- Words Entered Arraylist
    - Contains a list of the words entered by the user.

9.2 Base Wordlist

The base wordle list is predetermined for the next 12 years. For this application, it will be stored in a .txt file that the application will import and store in the dictionary arraylist. This method will allow us to easily update the Wordle dictionary if needed.

9.3 Word Suggestions / Possible Answers

The possible answers provided by the application use a series of steps to determine the best possible answer. The list of possible answers will be stored in an arraylist. Most of the steps are just a process of elimination, but the last step is to order the possible answers list according to how common the letters in that word are.

9.4 Words Entered / User Input

The UI created to allow the user enter their word is very straight forward. There will be a 3x5 grid of text boxes that will allow the user to enter the letters of a word. Using color coded text boxes will also allow the user to know where to enter correct, valid or invalid words.

10. Logic & Algorithms

10.1 Logic & Algorithms Overview

The logic used to determine the strength of each letter will be found in the strength class. Below is a quick summary of how that logic would go.

A *for loop* will be used to iterate through the Wordle text file as the words are being read into the dictionary arraylist. It will use a case statement to check what letter is present, then increase that letter's strength by one. This will be done until the Wordle text file is fully read.

The second logic found in this project is the core of the application. Comparing user input and available words in the dictionary arraylist, it will help generate a list of possible words. The logic below will be used.

The application will grab the letters provided by the user. First it will use the invalid letters to cross reference and remove any words that contain those letters, reducing the possible answers. Secondly, it will grab the green letters and remove any words that don’t match the letter positions, further reducing the list of possible answers. Third, it will use the yellow words to check the possible words and sort them out, even further reducing the possible answers by a large margin. Lastly, it will use the strength list to organize the possible words in order by the highest strength to least and list the top 3. Once this is done, it will list the possible words, add the used word to the used list and color code the alphabet indicator at the bottom to inform the user what words are correct, valid or invalid.

10.2 Word Suggestions / Possible Answers

The Possible Answers will be pulled from the Wordle text file. As users correctly input their letters throughout each word they enter on the Wordle app - the Wordle Solver will be able to match the letters in their respective arrays to eliminate and cross reference the words in the text file.

The Wordle Solver should only be able to output possible words that match the user input. This is fully dependent on users to correctly input their data, or else the Possible Answers section will provide an incorrect list of possible answers.

The Possible Answers section will display only the top five possible answers as the users enter their data. The outputted data will possibly change as more data is inputted by the user. This gives users some autonomy to pick and choose from several words to be inputted into the Wordle app.

11. Development History

11.1 Part One: Project Plan

Designing the Project Plan was tasked for week two. The project plan was designed to include the scope of the project, designate roles, creating a timeline, and including a general description with the app's specifications.

11.2 Part Two: User guide and Test Plan

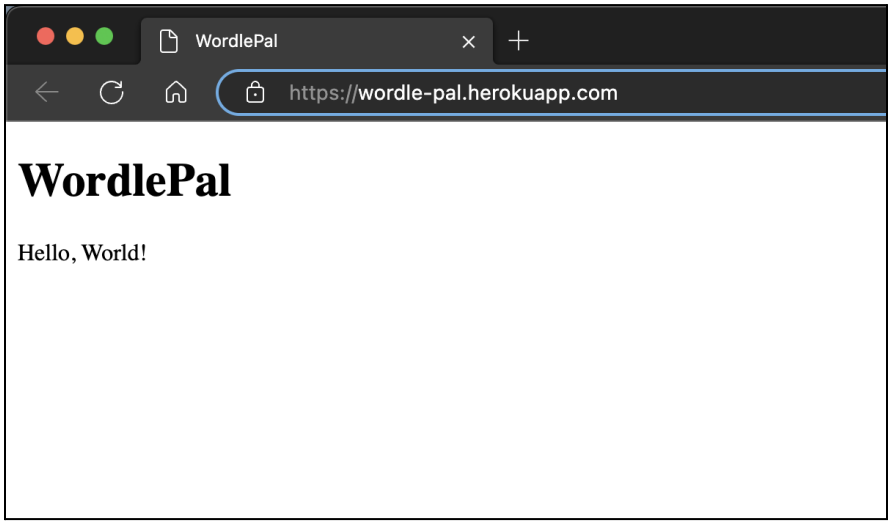
The user manual and an advanced test report was created for week three. The user manual is designed for user's to have a full guide on how to use the Wordle Solver. The user manual includes starting up the application, how to use the application, what to expect when an unexpected input is entered, and screenshots for reference.
The advanced test report was created to show future test cases to be run as the app further developed. The test report used in week two was added into the final report with the actual outputs and pass/fail results.

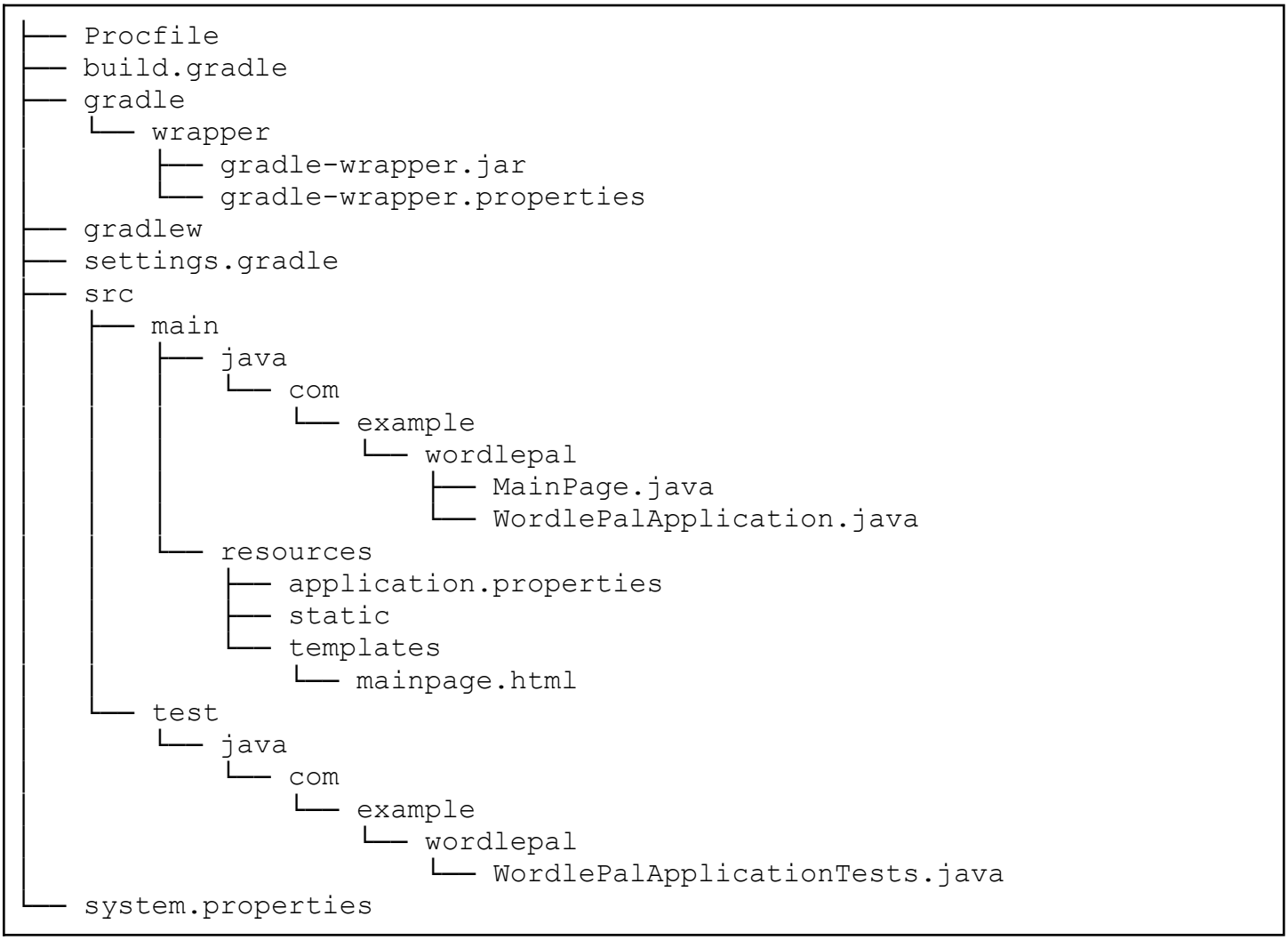
11.3 Part Three: Project Design

The project design was created in week four. The project design report includes the classes, data structures, and the algorithm overview of the entire application. The project design was followed for the entirety of the application.

11.4 Part Four: Phase 1

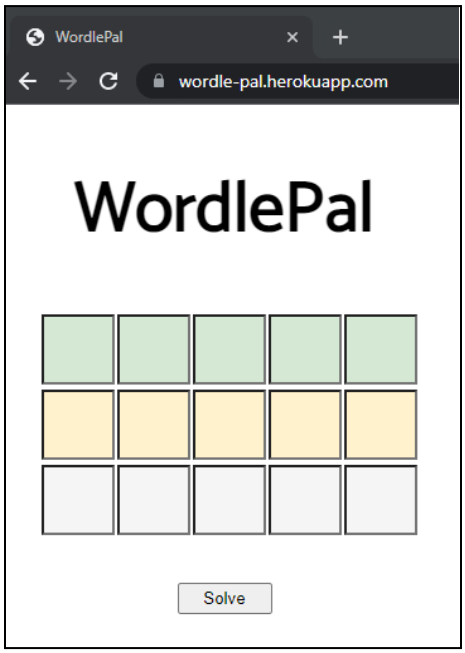
The first phase of actually developing the application started in week five. This is when the initial app was created and successfully displayed through Heroku. The source structure for the back-end of the application was designed for week four as well.




├── Procfile
├── build.gradle
├── gradle
│   └── wrapper
│   ├── gradle-wrapper.jar
│   └── gradle-wrapper.properties
├── gradlew
├── settings.gradle
├── src
│   ├── main
│   │   ├── java
│   │   │   └── com
│   │   │   └── example
│   │   │   └── wordlepal
│   │   │   ├── MainPage.java
│   │   │   └── WordlePalApplication.java
│   │   └── resources
│   │   ├── application.properties
│   │   ├── static
│   │   └── templates
│   │   └── mainpage.html
│   └── test
│   └── java
│   └── com
│   └── example
│   └── wordlepal
│   └── WordlePalApplicationTests.java
└── system.properties

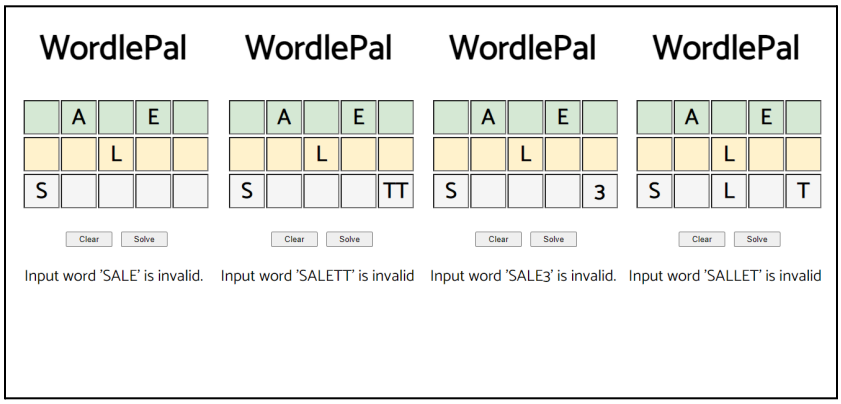
11.5 Part Five: Phase 2

The second phase of developing the app took place in week six. The general design of the app is beginning to finalize. The logic classes are beginning to be developed as well as the rest of the classes through Replit. The app successfully displays a 3x5 grid that includes a green row, yellow row, and gray row.



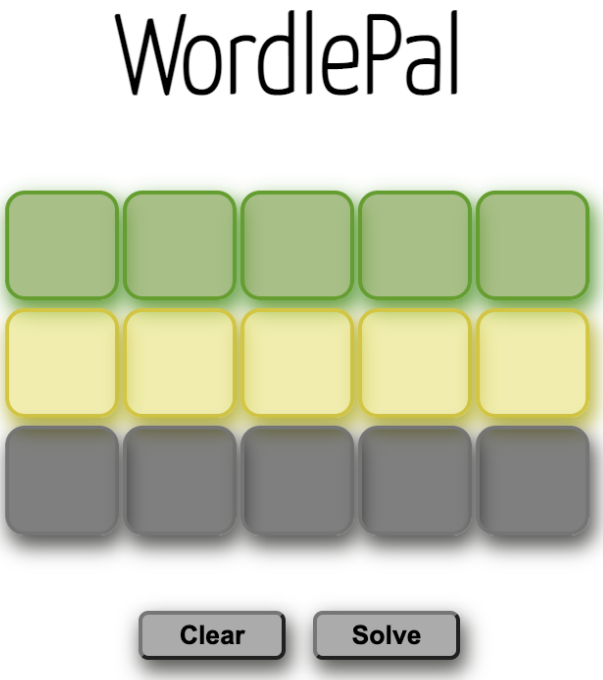
11.6 Part Six: Phase 3

The third phase of developing the app took place in week seven. The app can now recognize if inputs are entered incorrectly and produces an error message to the user. This includes entering multiple letters into one box, if less than five letters are entered into the grid, and if users are misentering letters into the grid. The app will also output the possible answers to the users.
All necessary classes were also developed for the back-end of development.



11.7 Part Seven: Final Deployment

Finalizing the application and full deployment of the app takes place in week eight. The classes have now fully integrated with the GUI, and users are able to successfully use the app. All minor editing and formatting was applied during this week to ensure the app was ready for deploying.



12. Conclusion

12.1 Lessons Learned

Throughout the course of this project - real world processes were implemented in order to successfully develop the application. Outside of the mentioned platforms used to create this application, useful apps such as Discord and Google docs were used to communicate and document effectively as all team members worked remotely on this project.

12.2 Design Strengths

- The Wordle Solver accounts for any unexpected input and alerts users that their input is invalid.
- The list of possible words are all stored in a txt file, but only display (at most) the top 5 words to avoid an overflow of output.
- Users can refresh, clear, and enter invalid inputs without it affecting the stored data from the session.
- The app does not involve any type of login and each session ends as users close out the application.

12.3 Limitations/Suggestions

- The Wordle Solver allows users to execute invalid input
- The Wordle Solver could potentially implement a way for users to only be able to enter letters and only 1 letter per box, and it avoids users from even having an error output.