# CMSC 447

### Software Design and Development

# Fall 2022

# **Wordle Coach App Design Document**

Wordle Coach

App Design Document

#### **Table of Contents**

Page

## 1. Introduction 3

1.1 Purpose of This Document 3

1.2 References 3

1. System Architecture 4

2.1 Architectural Design 4

2.2 Decomposition Description 5

3. Persistent Data Design 7

3.1 Database Descriptions 7

3.2 File Descriptions 7

1. Requirements Matrix 8
2. Open issues 8

Appendix A – Agreement Between Customer and Contractor 9

Appendix B – Peer Review Sign-off 10

Appendix C – Document Contributions 11

1. **Introduction**

1.1 Purpose of This Document

The purpose of this document is to describe the logical and architectural components of the Wordle Coach for system developers and administrators. It will outline all the system components that meet the requirements from the System Requirements Specification document. Several diagrams will be included to illustrate how the system components interact with each other.

1.2 References

Team Odin. *Wordle Coach System Requirements Specification*. UMBC CMSC 447, 2022. Accessed 11 Nov. 2022.

Visual Paradigm. "What is Class Diagram?" *Ideal Modeling & Diagramming Tool for* *Agile Team Collaboration*, Visual Paradigm, 27 Nov. 2022, www.visual paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/. Accessed 27 Nov. 2022.

Code Repository:

<https://github.com/wwills2/wordle_coach>

2. **System Architecture**

2.1 Architectural Design

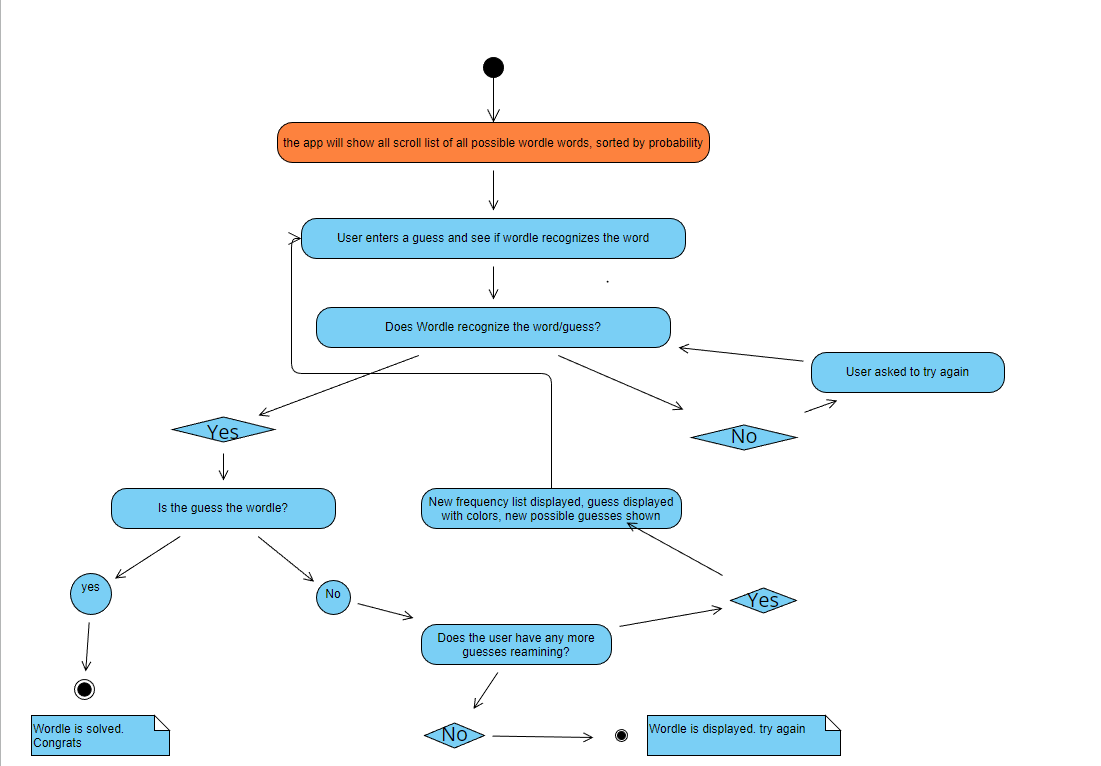


Figure 1

The first thing that the user will encounter when loading and running in the app is the app’s main screen. The user will be able to enter a word and long press on each letter to cycle through the colors gray, yellow, and green. When user clicks “Enter guess” the app will display the top 25 words that have the highest probability of being the Wordle Word of the day based on the frequency percentage of every single letter. If the guess entered by the user isn’t in the list of five letter words that wordle recognizes, or if the word isn’t a word which possible given previous guesses, it will prompt the user to enter a different guess. If the guess is in the list, then it will check to see what letters are in the wordle based on the position of the word and assign a color. Once the guess and colors are displayed, then the algorithm will eliminate words based on the letters that aren’t a part of the wordle and letters based on positions of the letters that are in the wordle.

After that is done, the algorithm will create a new list of words based on the frequency of letters after it has eliminated words based on the guess. Every time that the user enters a guess that isn’t in the list of five-letter words that wordle recognizes, it will ask the user enter a different guess until the user enters a word that wordle recognizes or is possible given previous guesses. Every time that the user makes a guess that isn’t the wordle, it will keep eliminating words that aren’t possible. Each attempt that the user makes will shrink the best possible guess list until either the user uses all 6 of their possible guesses and didn’t solve the wordle, or if the algorithm displays only one possible word based on the user’s guesses and guarantees them the wordle. If the user figures out the wordle using anywhere from 1 guess to all their guesses, then they solved the wordle otherwise the wordle will be displayed after the user’s last attempt.

2.2 Decomposition Description

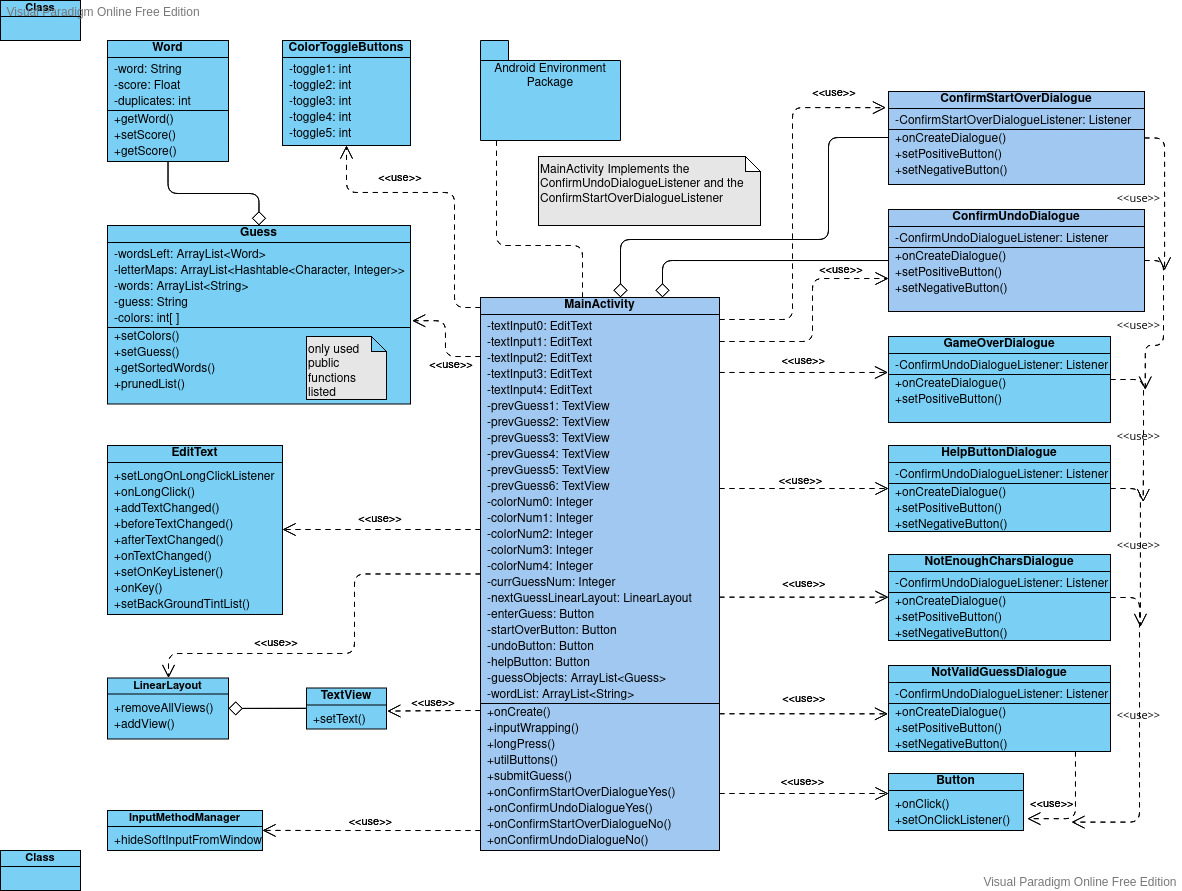


Figure 2

Figure 2 is the Wordle Coach Class diagram, which shows a decomposition of the wordle coach app into the component classes which implement core functionality of the app. Please note that the app consists of many classes which exist as part of the android SDK. Though these classes support and enable our app’s code, they are not included in this diagram because they do not contain any of team Odin’s intellectual property, are not referenced in a meaningful capacity within the developed code, and are common to every android project. The SDK is represented by the Android Environment Package in the diagram, which is connected to the main activity. It should be noted that the SDK supports the interactions and relations between most of the classes in this diagram, which are outside of the scope of this diagram.

The MainActivity class defines and implements the functionality of the main (and only) screen for the Wordle Coach app. While not a main function in the traditional sense, it fills the role of one. It implements much of the UI and algorithm logic, takes in user input, sanitizes user input, tracks the state of the current Wordle game, and makes calls to other classes for functionality as needed, which is illustrated by the <<use>> relationships between it and the other classes. Note that all classes have a <<use>> relationship with MainActivity, except for the ConfrimStartOverDialogue and ConfirmUndoDialogue, which also have an aggregation relationship and are slightly different shade of blue to match the blue used in MainActivity. This relationship exists because MainActivity implements functions of these classes. ConfrimStartOverDialogue and ConfirmUndoDialogue both need to execute actions which can only be taken if it exists within the scope of MainActivitiy.

It is worth noting that the TextView class is both used by the MainActvity class and aggregate to the ScrollView. MainActivity uses the TextView class to display static information and the previous guesses, and the ScrollView class uses TextView classes to display the list of probable words.

The Guess class contains the logic for generating the list of words and their probabilities based on the current guess and the colors of each letter in the guess. Each time the user enters a guess, a new guess object is created and saved. The guess object does not remember any previous guesses, and instead takes in a list of words when the object is instantiated. When the filtered list of probable words is passed onto the next Guess object, the next object can only choose from words which are plausible given the previous guesses. The Word class is aggregated to the Guess class because each word object contains the frequency and color data of each letter in the word, which the guess class relies on to generate and sort the list of probable words.

3. **Persistent Data Design**

3.1 Database Descriptions

We do not have any databases in our application.

3.2 File Descriptions

We have a text file that contains all the five letter words that wordle approves and considers as a five-letter word. The text file is all five-letter word strings. Next there is the algorithm file that opens the five-letter word text file and generates a list based on those words. The algorithm then finds the frequency of each letter in the dataset and returns the percentage of frequency based on each letter which is a float. Next up is getting the words with the highest frequency based on the frequency of the letters and will return a dictionary in which the key is the word and the value if the percent of frequency. Afterwards, there is return a list of the output based on the colors since wordle uses grey, yellow, and green to give the user an idea of how effective their guess was in determining the wordle. Finally, there is eliminating the words that aren’t possible based on the user’s guess. It will eliminate the words based on the letters that aren’t in the word, letters that are green in the same position, and letters that are yellow in the same position since yellow letters are in different positions but are part of the word. Then it will generate a new list of words that have the highest frequency based on the user’s guess and it will keep doing that for every guess that the user makes until either they get the wordle within their 6 attempts or if they use all 6 attempts and fail to get the wordle.

Note: This is an initial file mockup. The actual design varies slightly, and this section is superseded by the class decomposition description and the class diagram.

4. **Requirements Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement Number** | **Use Case Name** | **Components Which Satisfy the Use Case** | **Priority** | **Test Case Number** |
| **1** | **Undo guess** | **Button to undo a guess, as well as a box to list all previous guesses** | **4** | **4** |
| **2** | **Insert word** | **5 boxes that hold each letter of the guess, which include the ability to tap on them to turn them yellow, green, or gray. Also a button to confirm the word guess** | **5** | **1** |
| **3** | **Scroll through words** | **Box that contains the recommended word list. The user swipes down to see more guesses** | **4** | **2** |
| **4** | **Reset wordle board** | **Button to reset the wordle board** | **4** | **3** |
| **5** | **Open help menu** | **Button to open the help menu** | **3** | **5** |

**5. Open Issues**

There are no open issues at this time.

**Appendix A – Agreement Between Customer and Contractor**

By signing this document, the customer agrees that all components of the system are accounted for and that all functional requirements are covered by the components. Further, they agree to incur any additional costs associated with changing the SDD after it has been completed.

By signing this document, the team agrees to design the components of the system so that all the defined requirements can be implemented appropriately. The dev team also agrees to not charge the customer additional fees for the remediation of bugs and errors.

**Appendix B – Team Review Sign-off**

This sign-off agreement confirms that all members of the Wordle Coach development team have both reviewed and agree with the provided content. Each member who has signed this has validated the system design specifications and agrees that this work will be submitted as a team effort.

Signature: Date:

Zan Wills 12/11/2022

Jamie Kirk 12/10/2022

Dennis Mayher 12/9/2022

Parth Patel 12/11/2022

Nathan Hoernlein 12/10/2022

**Appendix C – Document Contributions**

|  |  |
| --- | --- |
| Name | Role |
| Zan Wills | Class Diagram and Decomposition Description |
| Dennis Mayher | File Description |
| Nathan Hoernlein | Requirements Matrix |
| Parth Patel | Architectural Design |
| Jamie Kirk | File Description |