

Wordly Game - Project Update

Renn Gilbert

CSC1061: Computer Science II

March 28, 2024

Section 1

Requirements

Requirements

- ① The project shall only allow the user to guess 5 letter words.
- ② The project shall allow the user to enter 6 guesses.
- ③ The project shall compare users' guesses to the correct word, giving them feedback using colors.
- ④ The project shall read a list of words from a list of upcoming words.¹
- ⑤ The project shall store user data and statistics in a file.
- ⑥ The project should validate users' guesses by comparing them with a dictionary file.
- ⑦ The project should allow the user to continue playing after the day's word is complete.
- ⑧ The project will use the Model-View-Controller-Interactor architecture.
 - Article on PragmaticCoding
- ⑨ The project will use AtlantaFX for GUI controls.

¹this requirement was changed

Requirement Change

Requirement #4 was changed.

- ④ The project shall read a list of words from **Wordle's** list of upcoming words.



- ④ The project shall read a list of words from **a** list of upcoming words.

Why?

After the New York Times bought Wordle, they switched the word lists many times. Now, they are curating a word each day.

Section 2

Class Diagrams

Class Diagrams

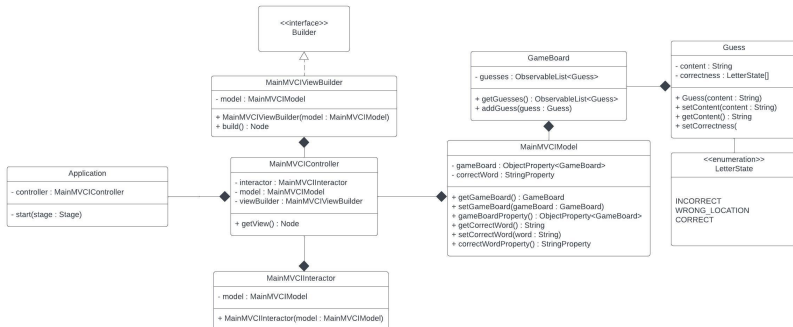


Figure 1: UML Class Diagram

Section 3

Flow Diagrams

GUI Flow Diagram

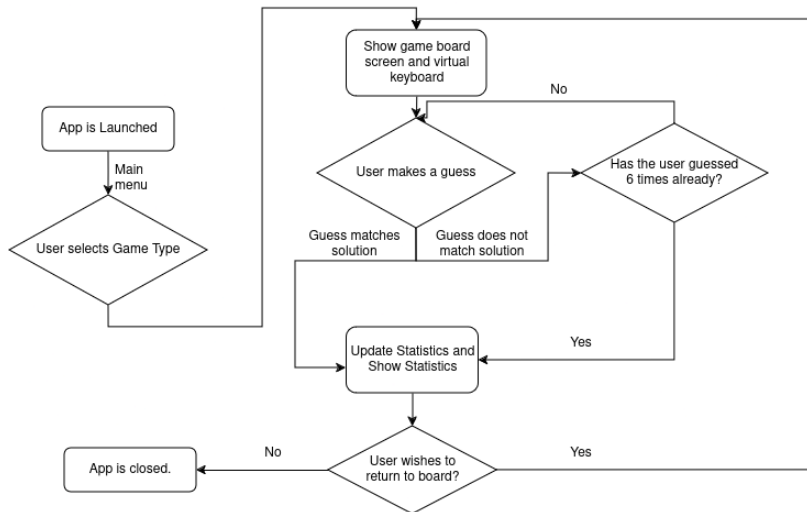
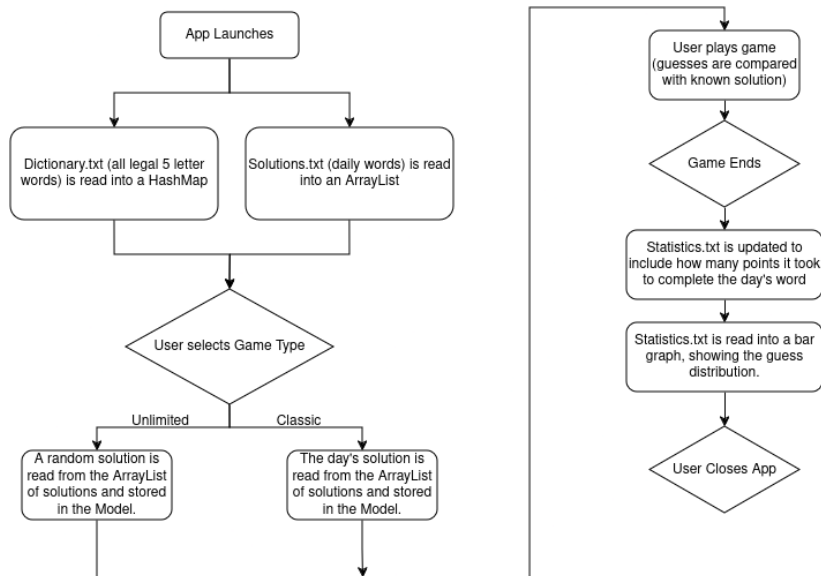


Figure 2: GUI Flow Diagram

File Flow Diagram



Data Flow Diagram

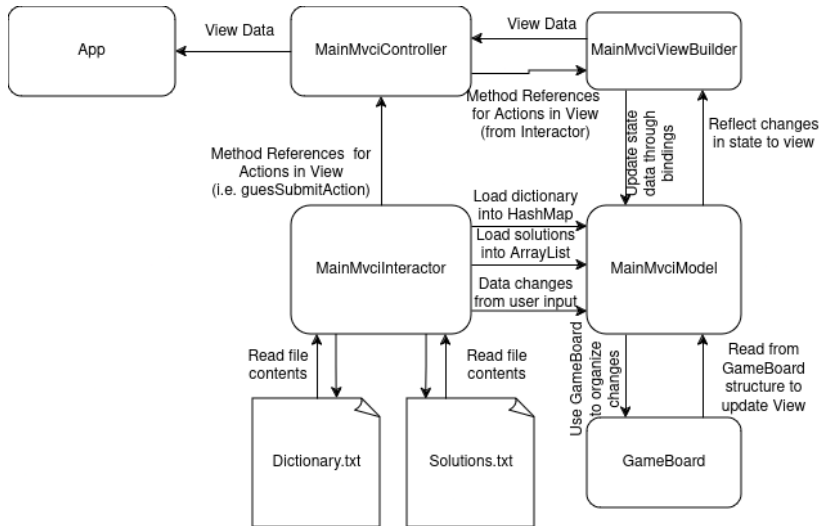


Figure 4: Data Flow Diagram

Section 4

Project Setup

GitHub Repository

The screenshot shows the GitHub interface for a repository named 'WordlyGame' by user 'Kylo33'. The repository is private and has 0 forks and 0 stars. The main branch is 'main' with 1 branch and 0 tags. The repository contains several files and folders, including 'LICENSE', 'README.md', 'slides.md', and 'slides.pdf'. The 'README.md' file is selected, showing the project requirements. The requirements list six tasks for the Wordle game, including allowing users to guess 5-letter words, enter 6 guesses, compare guesses to the correct word, read a list of upcoming words, store user data, and validate guesses against a dictionary.

WordlyGame Private

Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file Code

Files

File	Commit	Time
vscode	Initial project files	yesterday
bin	Bare-bones MVC set up	now
resources	Reorganize	9 minutes ago
src	Bare-bones MVC set up	now
LICENSE	Initial commit	yesterday
README.md	Reorganize	9 minutes ago
slides.md	Reorganize	9 minutes ago
slides.pdf	Move all project files here.	15 minutes ago

README MIT license

Requirements

1. The project shall only allow the user to guess 5 letter words.
2. The project shall allow the user to enter 6 guesses.
3. The project shall compare users' guesses to the correct word, giving them feedback using colors.
4. The project shall read a list of words from a list of upcoming words.
5. The project shall store user data and statistics in a file.
6. The project should validate users' guesses by comparing them with a dictionary file.

About

A well-designed and maintainable version of the Wordle game created with JavaFX.

Readme MIT license Activity 1 watching 0 forks

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

Languages

Java 100.0%

Suggested workflows

Based on your tech stack

Scale Configure

Figure 5: GitHub Initial Setup

Bare-bones setup

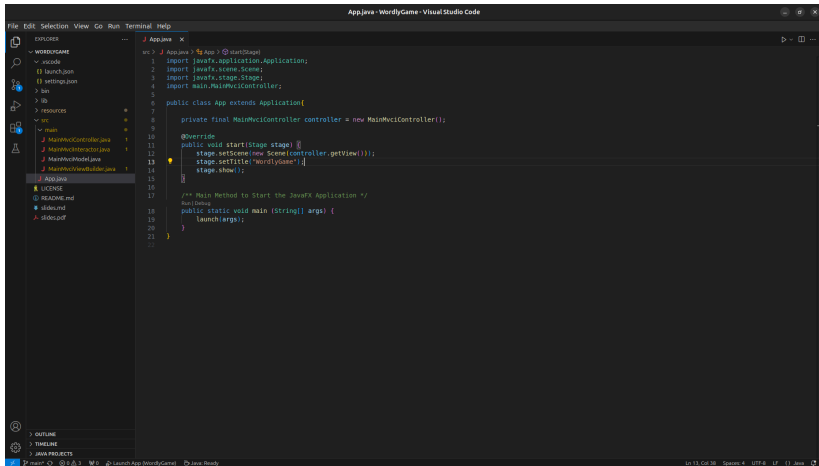


Figure 6: Bare-bones project setup with VSCode

Section 5

Research

Compiling Word Lists

For my research, I decided to compile my list of words and solutions.

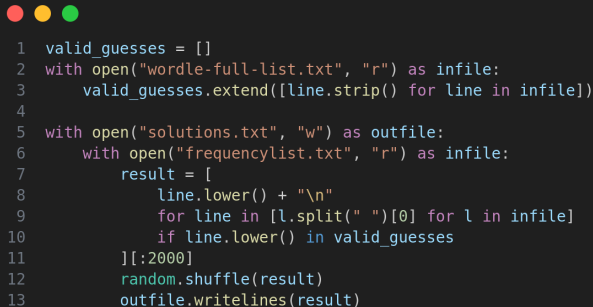
dictionary.txt

To compile `dictionary.txt`, a list of valid words, I manually scraped the data from the official Wordle valid word list. [Link](#). The resulting list is over 14,000 words long.

solutions.txt

To compile `solutions.txt`, a sequential list of upcoming solutions, I used `FrequencyWords`, a GitHub repository with words sorted by their frequency in a corpus (in various languages). I filtered it for words that are in `dictionary.txt`, limited it to the 2,000 most common, and shuffled it.

Script



```
1 valid_guesses = []
2 with open("wordle-full-list.txt", "r") as infile:
3     valid_guesses.extend([line.strip() for line in infile])
4
5 with open("solutions.txt", "w") as outfile:
6     with open("frequencylist.txt", "r") as infile:
7         result = [
8             line.lower() + "\n"
9             for line in [l.split(" ")[0] for l in infile]
10            if line.lower() in valid_guesses
11        ][:2000]
12     random.shuffle(result)
13     outfile.writelines(result)
```

Figure 7: Python script to filter solutions

Sample Data

dictionary.txt

aahed
aalii
aapas
aargh
aarti
abaca
abaci
abacs
abaft
abaht

solutions.txt

whine
spell
saved
parts
drank
sites
ducky
fatty
steak
stock

Section 6

The End

The End



Figure 8: NYT Wordle Image