Word Search Game

Final Report

James Bell
Daniel Montes

Spring 2021
CSCI 2300: Object Oriented Software Design
Professor Fairouz Medjahed
Saint Louis University

SAINT LOUIS UNIVERSITY™

EST. 1818

Table of Contents

Version 2 Modifications

2.2.1 Non-Critical Functional Requirements and 2.3.1 Non-Critical Non-Functional Requirements removed.

2.5 Use Case Scenarios, 2.7 User Interface, 3.1.2 Class Diagrams updated.

3.3 MVC Design included

1. Introduction

A word search, word find, world seek or mystery word puzzle is a classic word game that consists of scrambled letters placed in a rectangular grid or table with a hidden word among them. The hidden word can be placed vertically, horizontally, or even diagonally. The objective of the came is to find all the hidden words in the table before time runs out.

2. Requirement Analysis

2.1. Overview

Our task was to create a word search game with a timer and a scoreboard. The timer would start as soon as the game does, and if the player finds the word before the timer runs out, they advance to the next level, or else they lose. The score is dependent on how long the player took to find the hidden word. Our implementation will include increasing levels of difficulty, high scores, conditionally playing sounds and different word categories.

2.2. Functional Requirements

- When the player opens the application, a main menu must be shown, allowing the user to see the high score and start/quit the game..

- When the player starts the game, the system chooses a random category from the possible candidates and generates the first level.
- The timer must be started at the beginning of each level.
- A text box is where the user guesses the word.
- If the player enters all words, they advance to the next level
- If the player finds all words of the last level, they win the game.
- If the timer runs out before the player found the words, a message appears letting the user know he lost and the answer is given to him
- The system must always show the score and time remaining.
- The player can start a new game or quit
- A list of candidate words is available in a file.
- More sounds should be added into the game (main menu theme, timer is running out...)

2.3. Non-Functional Requirements

- Words' length should be greater or equal than 5.
- If the player enters the correct word, his score is calculated taking into account the time it takes him to find the word.
- Highscore must be kept.
- The word must be placed vertically, horizontally or diagonally. It also can be read up or down.
- Timer starts when the player enters the level.
- Game is over when the timer runs out or word is found.

- At least 3 levels of increasing difficulty.5
- The window layout and size should adapt to the table size.
- Timer must last 60 seconds.

2.4. Constraints

- Words' length should be greater or equal than 4.
- Timer determines length of game

2.5. Use-Case Scenarios

Title	Player Finds Word
Description	Scenario for when player loads and plays game
Actors	Player
Initial Status	Player has started a new game
Flow	- Game with timer appear on screen - Timer counts down - User finds word - User enters word in text field and submits - System verifies answer - If unverified, delete text and keep timer running - Player's score is updated - System plays a sound - If all the words have been found, the player advances to the next level or

	wins the game if they are at the last level. Otherwise, go back to step 2
Post Condition	Player advances to next level, player wins game or they keep trying to find words in the same level

Title	Player Advances Level
Description	Scenario for when player finds all the words in a level and advances to the next
Actors	Player
Initial Status	Player has found all the words in a game
Flow	- Player's score is increased - System plays next level sound - System generates new level
Post Condition	New level is loaded into the screen

Title	Player Loses
Description	Scenario for when player loses the game
Actors	Player
Initial Status	Game started
Flow	- Game with timer appeared on screen - Timer counts down - Timer runs out
Post Condition	Player given endgame options

Title	Player Wins the Game
Description	Scenario for when player wins the game
Actors	Player
Initial Status	Player is in the last level
Flow	- Game with timer appear on screen - Timer counts down - Player finds all words - System plays sound
Post Condition	Player given endgame options

Title	System Generates New Level
Description	Scenario for when player advances to next level and system generates new one
Actors	Game system
Initial Status	Player must have found all words on previous level
Flow	- Once player has found all words, a sound is played - Difficulty is increased (adding more rows) and each table is generated with a random category and a number of words corresponding to the difficulty level. - Timer resets - New category is displayed on the window, along with the new table.
Post Condition	Game is ready to play

2.6. Use-Case Model

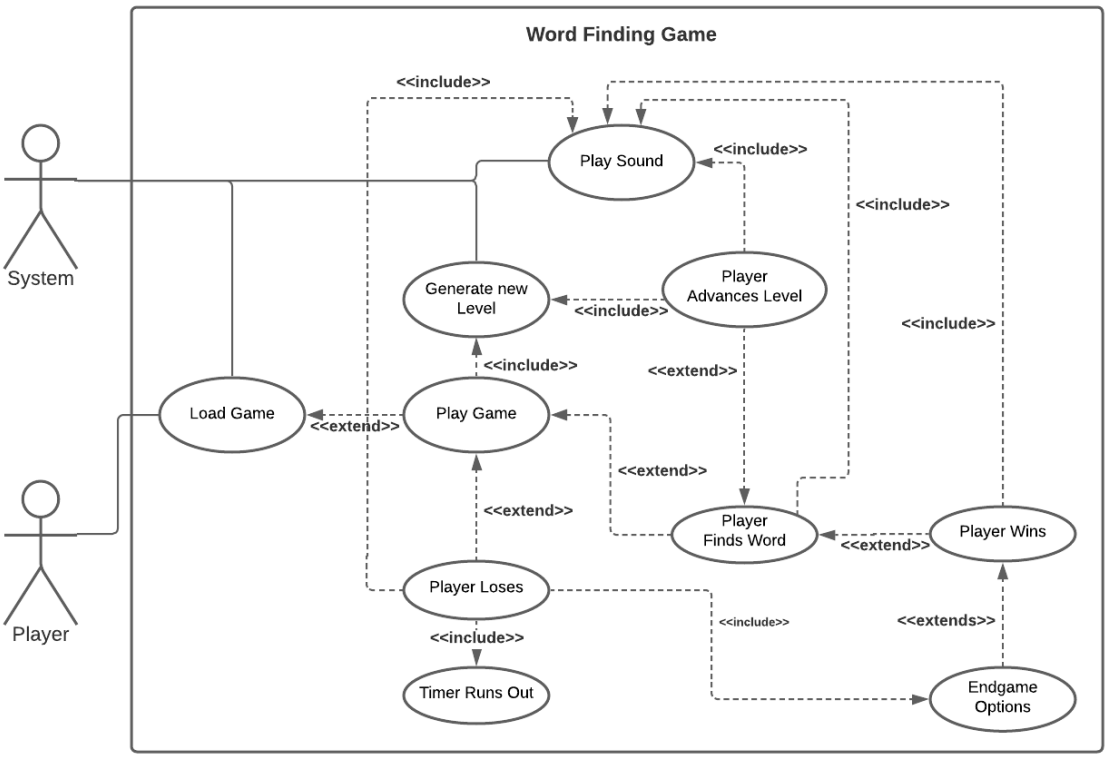


Figure 2.6.1 - Use Case Diagram

2.7. User Interface - Testing Models

The user interface is functional and minimalistic simultaneously providing for a simple yet enjoyable experience.

- Main menu:

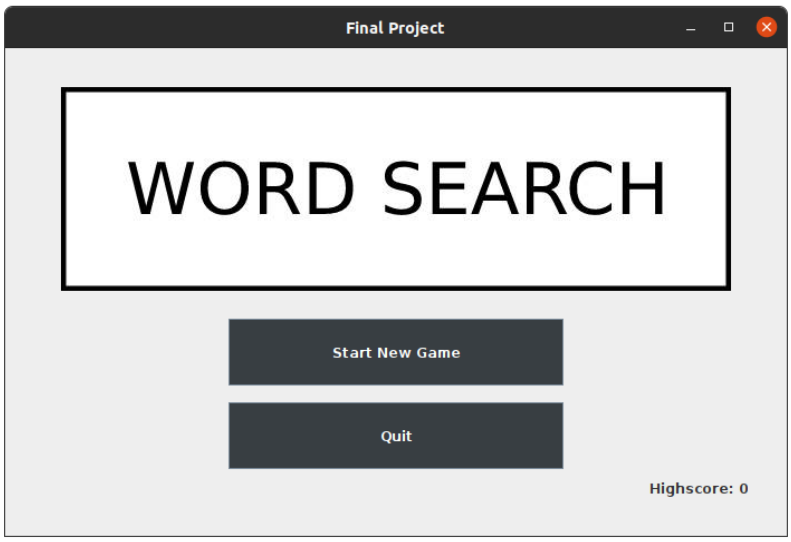


Figure 2.7.1 - Main Menu UI

  - New Game
  - Quit

- In game:

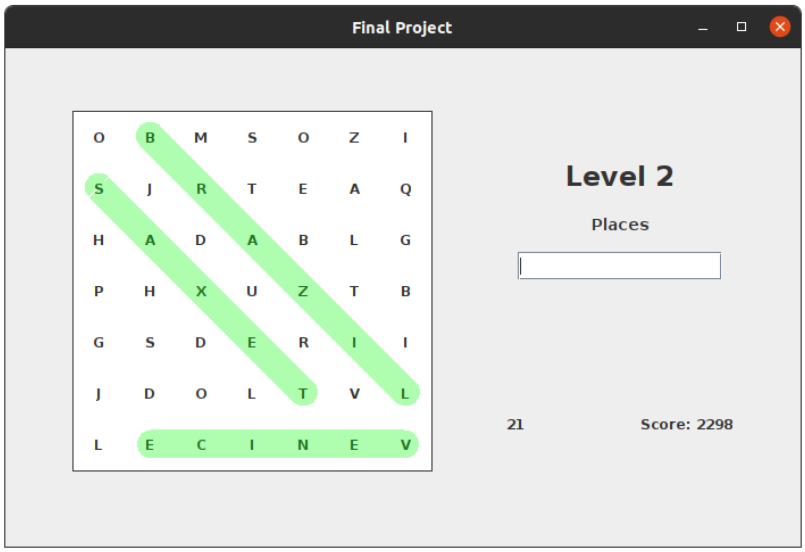


Figure 2.7.2 - In-game UI

  - Score
  - Timer
  - Table
  - Text box
- Endgame options:

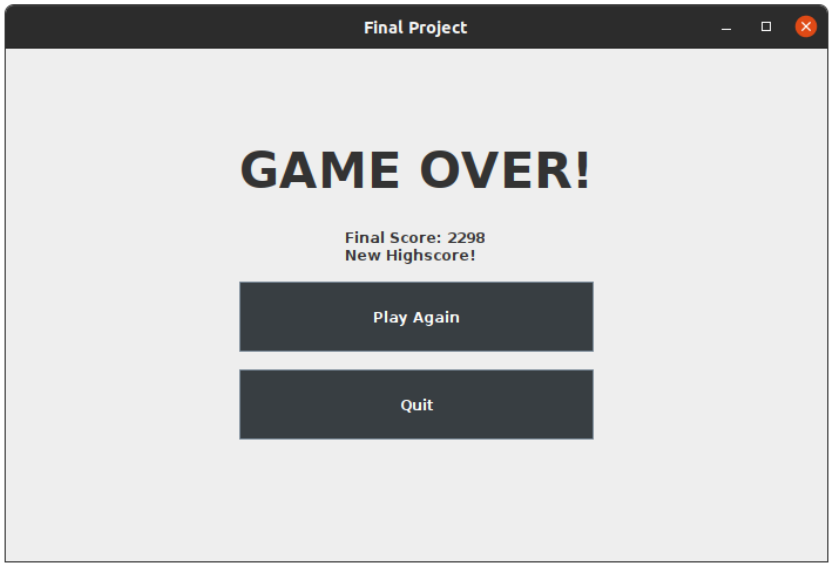


Figure 2.7.2 - End game UI

  - Final Score (Is it a high score?)
  - Start new game
  - Quit

3. System Models

3.1. Object Model

3.1.1. Domain Lexicon

Player: User who plays the game

Table: Square-shaped layout organized in rows and columns in which letters are randomly placed and words are hidden.

Timer: Countdown which determines how much time the user has left to find all the words in seconds.

Category: List of words with relation to each other.

3.1.2. Class Diagrams

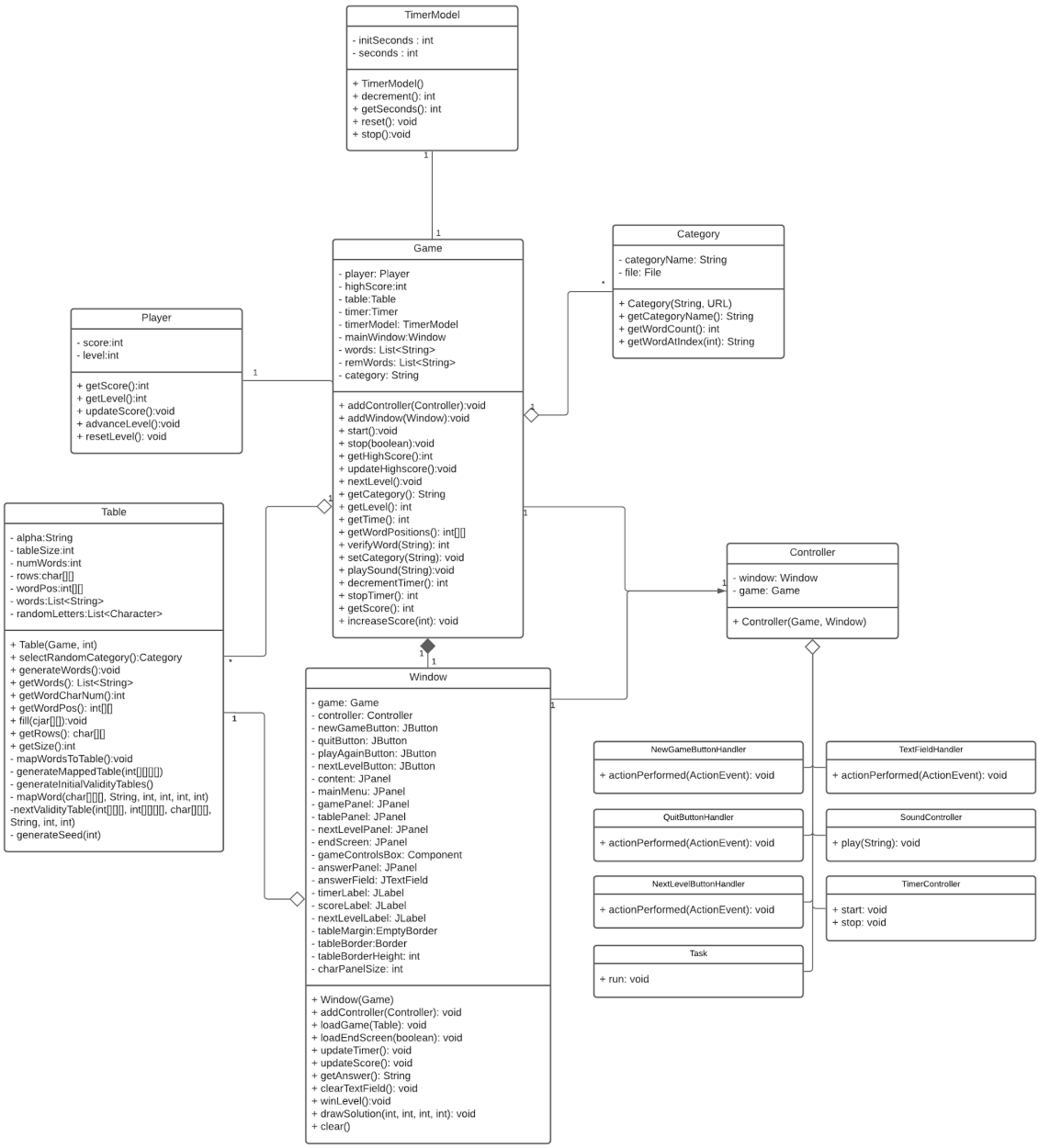


Figure 3.1.2.1 Class Diagrams

3.2. Dynamic Models

3.2.1. State Chart

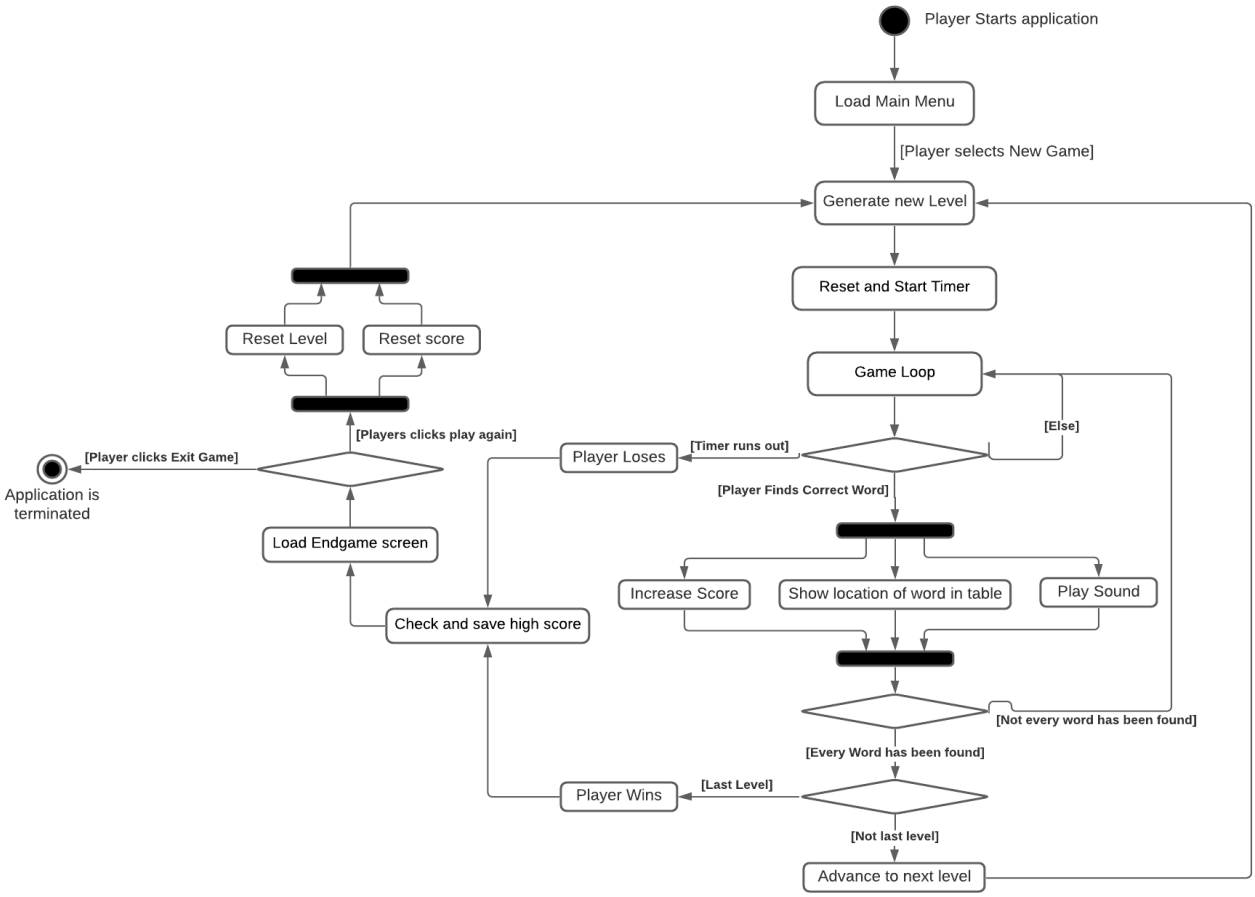


Figure 3.2.1 State Chart Diagram

3.2.2. Sequence Diagrams

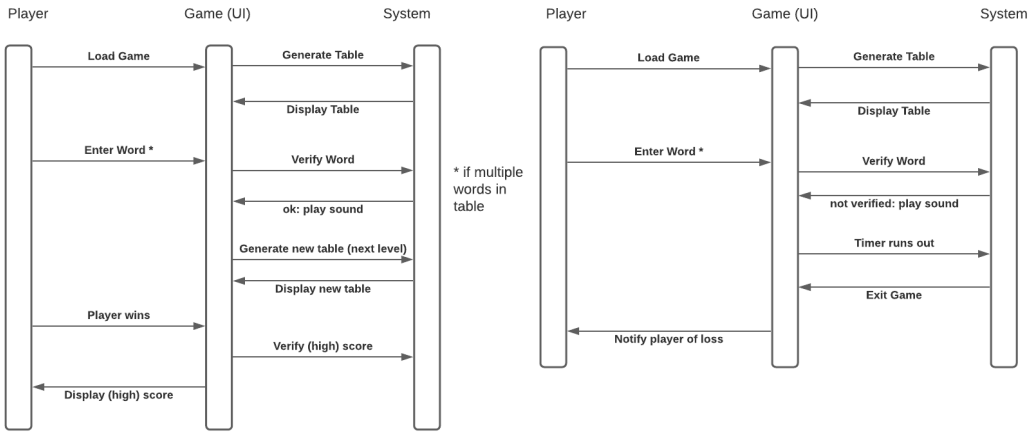
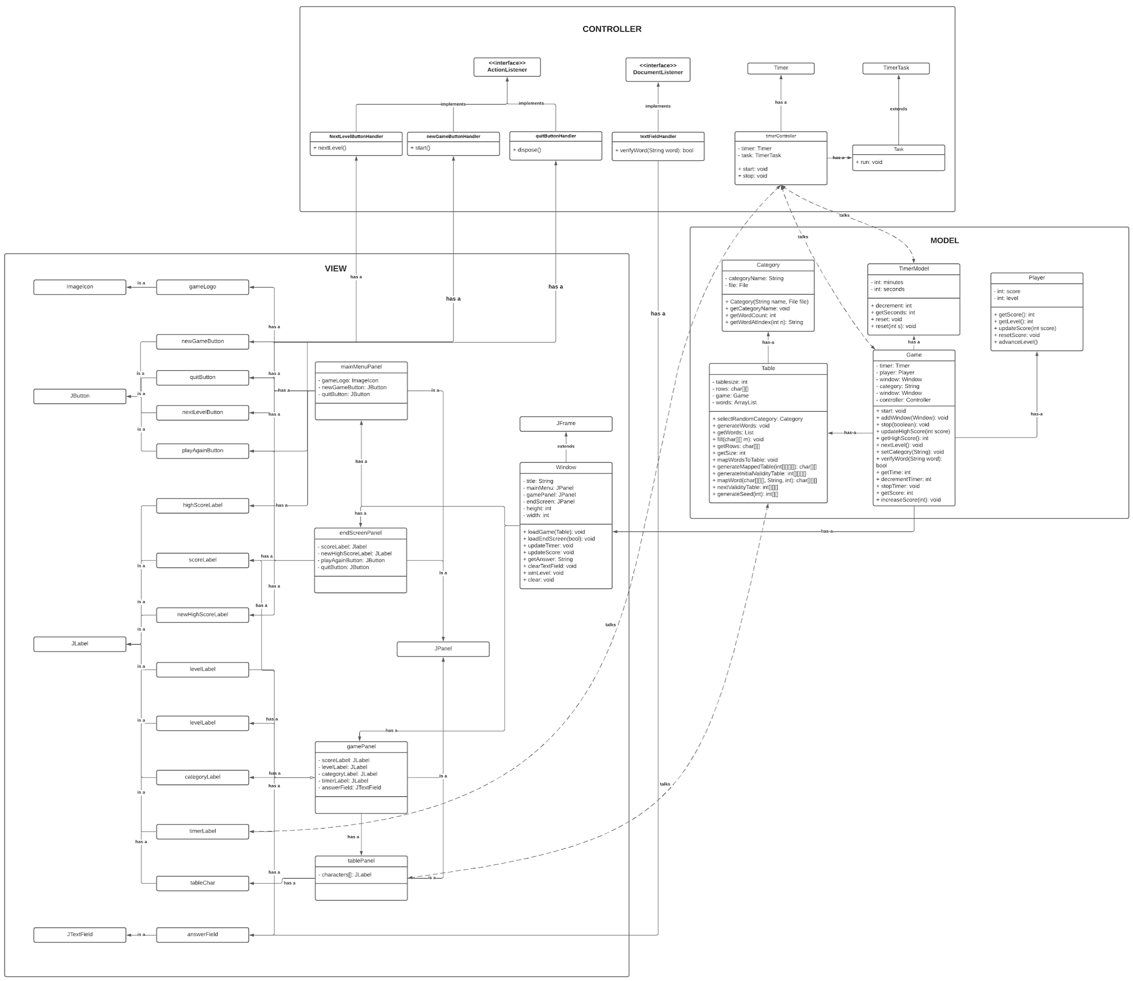


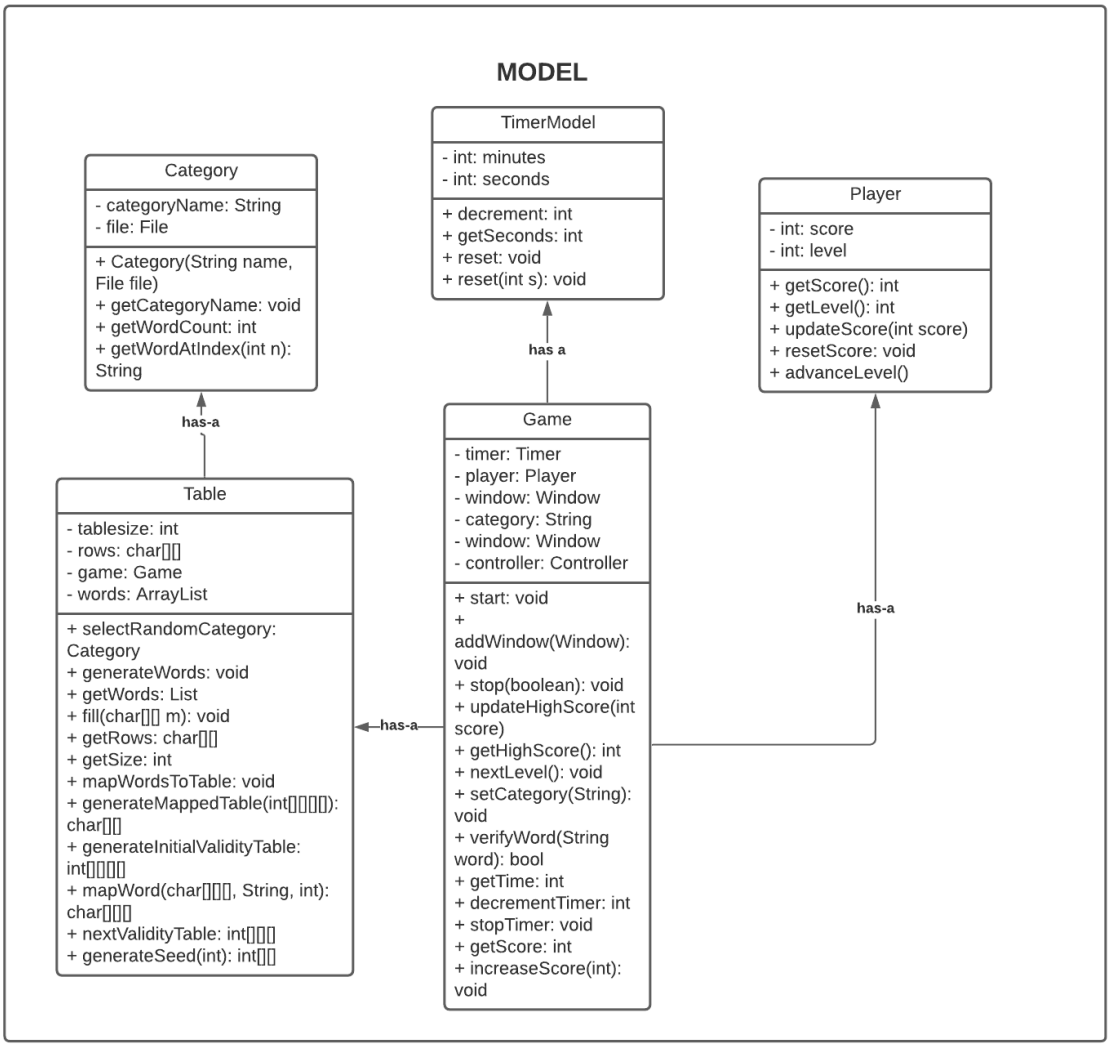
Figure 3.2.2 Sequence Diagram

3.3 MVC Design

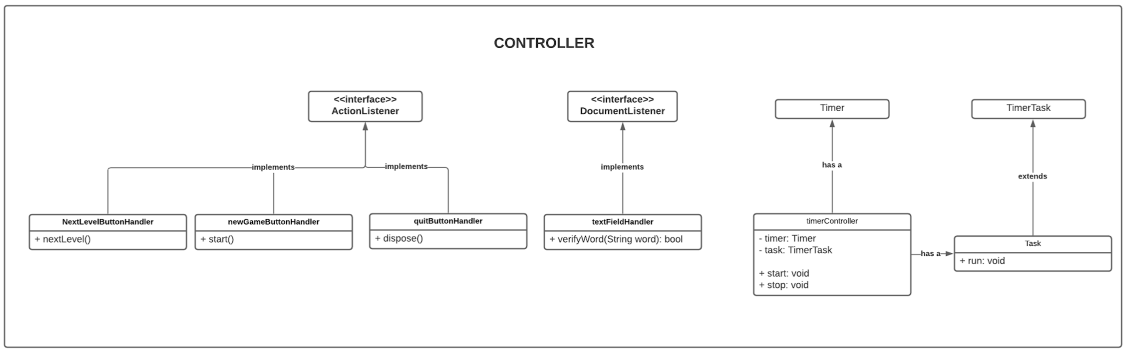
3.3.1 Top View



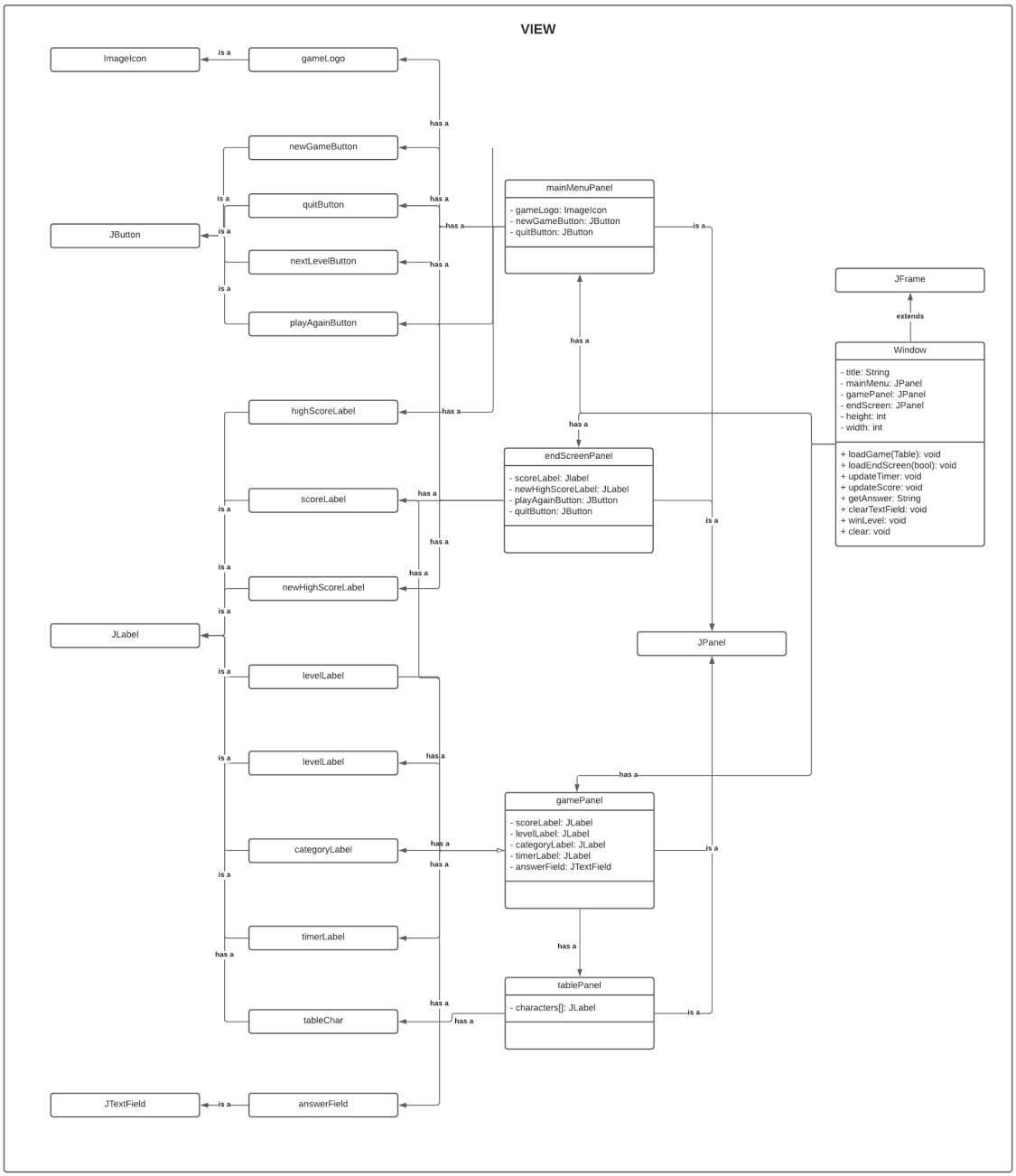
3.3.2 Model Diagram



3.3.3 Controller Diagram



3.3.4 View Diagram



4. Conclusion

4.1 Future Development

Although we accomplished and went beyond the original requirements for this project, we would have liked to implement more features to enhance the user experience and further our learning. For example, we would have designed a game logo to attach in the main menu, and in general a more appealing UI with fancy animations as the timer is running out or highlighting the word once it is found and being able to click and drag to select the word. A simple feature would have been to give the player the option to exit out at any time, or giving a penalty in the form of a reduction of time if they entered an incorrect word. A more advanced feature we thought of was having the user enter their name in the main menu, and keeping a database of the ranking of the best players and their high scores using JDBC API.

4.2 Team Contributions

Daniel was responsible for the frontend view aspect, due to his extensive familiarity with Java's Swing API. James was in charge of the model, or the backend, and both James and Daniel held responsibility for the controller. Although we had various responsibilities, we stayed in constant contact with each other and helped one another on their software design if they were having trouble. Most, if not all, of the Window class was designed by

Daniel, as was the algorithm for placing the words on the table and many of the controller classes. James implemented most of the Model class and its inner classes except for the word placement algorithm as well as introduced the timer class to which it was expanded upon by Daniel. We met roughly two to three times a week to keep each other up to date on our progress and for testing.

4.3 Final Remarks

To summarize, our word search game consists of a dynamic gameplay where the player will advance to the next level if they manage to guess all the words correctly. The game's user interface will include the letter pool, timer, and a text field to enter the words. A standard runthrough would start when the player loads the game, and the word pool of the first level is shown to him, which then starts the timer. If the player manages to guess all the words correctly before the timer runs out, a sound will play and the player will advance to the next level. The next level consists of a more difficult pool with more rows and columns. The same process is repeated as before. If the player reaches the last level and also guesses the words correctly, the game will display the player's score and notify whether they have the highest score. If during any of the levels, the timer runs out before the player enters the words, the game will play a losing sounds and notify the player that they have lost.