Wordle Report

Summary

[Introduction 2](#_Toc166948639)

[Frontend Implementation 2](#_Toc166948640)

[Game Grid 2](#_Toc166948641)

[GridView.builder Widget: 2](#_Toc166948642)

[Grid Configuration: 2](#_Toc166948643)

[Tile Rendering: 2](#_Toc166948644)

[Tiles Animation 2](#_Toc166948645)

[Dance Animation: 2](#_Toc166948646)

[Bounce Animation: 3](#_Toc166948647)

[Animation Integration: 3](#_Toc166948648)

[Keyboard Interface 3](#_Toc166948649)

[Statistics Display 3](#_Toc166948650)

[Backend Setup 3](#_Toc166948651)

[Database Structure 3](#_Toc166948652)

[API Endpoint 3](#_Toc166948653)

[Core Game Logic 4](#_Toc166948654)

[Guess Processing 4](#_Toc166948655)

[Answer Validation 4](#_Toc166948656)

[Game Completion 4](#_Toc166948657)

[Level Advancement 4](#_Toc166948658)

[Settings 4](#_Toc166948659)

[NLP (Machine Learning) 4](#_Toc166948660)

[Screens 6](#_Toc166948661)

[GitHub Link 11](#_Toc166948662)

[Conclusion 11](#_Toc166948663)

# Introduction

This report explores the educational potential of Wordle, a targeted word game designed to promote vocabulary expansion and intelligible engagement. Implemented in the adaptable Flutter framework, Wordle pull natural language processing (NLP) capabilities to dynamically generate various categories of word sentences and clues.

At its core, Wordle is further than just a game; It is a carefully designed educational tool designed to immerse users in an enveloping journey of language discovery. Wordle emphasizes focused attention and active problem solving, encouraging players to engage with different categories of knowledge, from cars to animals to capitals…

Additionally, we examine the impact of Wordle's educational design on academic institutions and explore its potential to complement traditional language instruction and increase student engagement. Harnessing the addictive appeal and focused teaching power of games, Wordle represents a promising milestone in the field of educational technology and is poised to revolutionize the way we learn and teach vocabulary.

# Frontend Implementation

## Game Grid

The game grid (grid.dart) is responsible for rendering the visual representation of the Wordle game. It utilizes Flutter's GridView.builder widget to create a grid of tiles, each representing a guessable letter.

### GridView.builder Widget:

The GridView.builder widget is used to create a scrollable, 2D array of widgets (tiles) in a grid format.

It efficiently creates grid tiles on-demand as the user scrolls, which is essential for performance, especially in games with large grids.

### Grid Configuration:

The grid is configured to display a fixed number of tiles horizontally and vertically using the SliverGridDelegateWithFixedCrossAxisCount class.

The mainAxisSpacing and crossAxisSpacing properties control the spacing between tiles in the grid.

### Tile Rendering:

Each tile in the grid represents a guessable letter in the Wordle game.

The itemBuilder callback function is invoked for each tile, dynamically generating and returning a widget based on the provided index.

## Tiles Animation

Each tile in the grid has animation effects applied to it. The Dance and Bounce animations provide visual feedback to the player when selecting tiles and upon successful completion of the game.

### Dance Animation:

Implemented using the SlideTransition widget, which animates the position of the tile along a specified axis.

The animation sequence consists of multiple keyframes, smoothly transitioning the tile's position to simulate a dance-like motion.

### Bounce Animation:

Implemented using the ScaleTransition widget, which animates the scale of the tile, making it appear to bounce.

The animation sequence comprises keyframes that scale the tile up and down, creating a bouncing effect.

### Animation Integration:

The Dance and Bounce animations are seamlessly integrated into the tile widget (Tile) using Flutter's StatefulWidget and AnimationController.

Upon triggering events such as tile selection or game completion, the corresponding animation is initiated, providing immediate visual feedback to the player.

## Keyboard Interface

The keyboard interface (keyboard\_row.dart) allows players to input their guesses. It dynamically generates keyboard rows and handles user interactions with individual keys.

## Statistics Display

The statistics box (stats\_box.dart) presents various game statistics to the player, including the number of games played, win percentage, current streak, maximum streak, and user’s level.

# Backend Setup

## Database Structure

The Wordle game relies on a database to store words and their descriptions. The database structure typically includes a table for words, each with fields for the word itself and its description.

**Word Field:** Stores the actual word that players attempt to guess during gameplay.

**Description Field:** Stores a brief description or hint related to the word, providing context or clues to the player.

## API Endpoint

In the Wordle game, the API endpoint is implemented using a PHP script named word\_api.php. This endpoint acts as a bridge between the game application and the database, allowing the game to retrieve word data dynamically during gameplay. The API endpoint follows RESTful principles and accepts parameters to customize the data retrieval process.

The word\_api.php script is responsible for handling incoming HTTP requests from the game application.

It processes these requests to retrieve word data from the database and formats the response in JSON format for consumption by the game.

# Core Game Logic

## Guess Processing

When players input their guesses, the game logic (Controller.dart) processes each guess, evaluates its correctness, and updates the game state accordingly.

Upon receiving a guess, the game logic processes it to determine its correctness and update the game state accordingly.

**Controller.dart:** This file contains the implementation of guess processing logic, orchestrating interactions between the game UI and backend game components.

## Answer Validation

Each guess undergoes validation against the hidden word to assess its correctness.

The validation process categorizes guesses into correct letters, incorrect letters, and letters present in the word but in the wrong position.

The game logic, implemented within Controller.dart, includes algorithms to perform answer validation based on the comparison between the guessed word and the hidden word.

## Game Completion

The game reaches its conclusion under specific conditions, such as when the player correctly guesses the word within the allotted attempts or exhausts all attempts without success.

Upon completion, the game displays the outcome to the player, indicating whether they successfully guessed the word or failed to do so.

The game logic within Controller.dart manages the game completion process, including result determination, display, and replay functionality.

## Level Advancement

If you guess the word correctly, a statistics screen will display, showing the number of games you've won, your current streak, other relevant information, and your level. Achieving an 80% win rate will advance you to the next level, where a new category will be added to the previous ones, increasing the difficulty.

## Settings

In the app's settings, you can reset your progress to start from level zero or switch the app's theme between light and dark modes.

# NLP (Machine Learning)

In this project we use fundamental NLP (Natural Language Processing) techniques, which are a subset of machine learning, such as:

* **Data Processing**: Reading and parsing JSON data containing words and descriptions.
* **Random Sampling**: Selecting random words to ensure a varied and unpredictable gameplay experience.

The API.php file acts as a bridge between the frontend and the backend, handling user requests and executing the appropriate Python script based on the level code.

* **Receive Request**: The PHP script receives a POST request containing a code that corresponds to a specific game level.
* **Execute Python Script**: Depending on the code (e.g., "662" for level 1, "663" for level 2), it executes the respective Python script using shell\_exec.
* **Process Output**: The output from the Python script is a JSON string, which is decoded and checked for a successful status.
* **Send Response**: If successful, the PHP script formats the response data into a JSON object and sends it back to the client. If there's an error, it sends an error message.

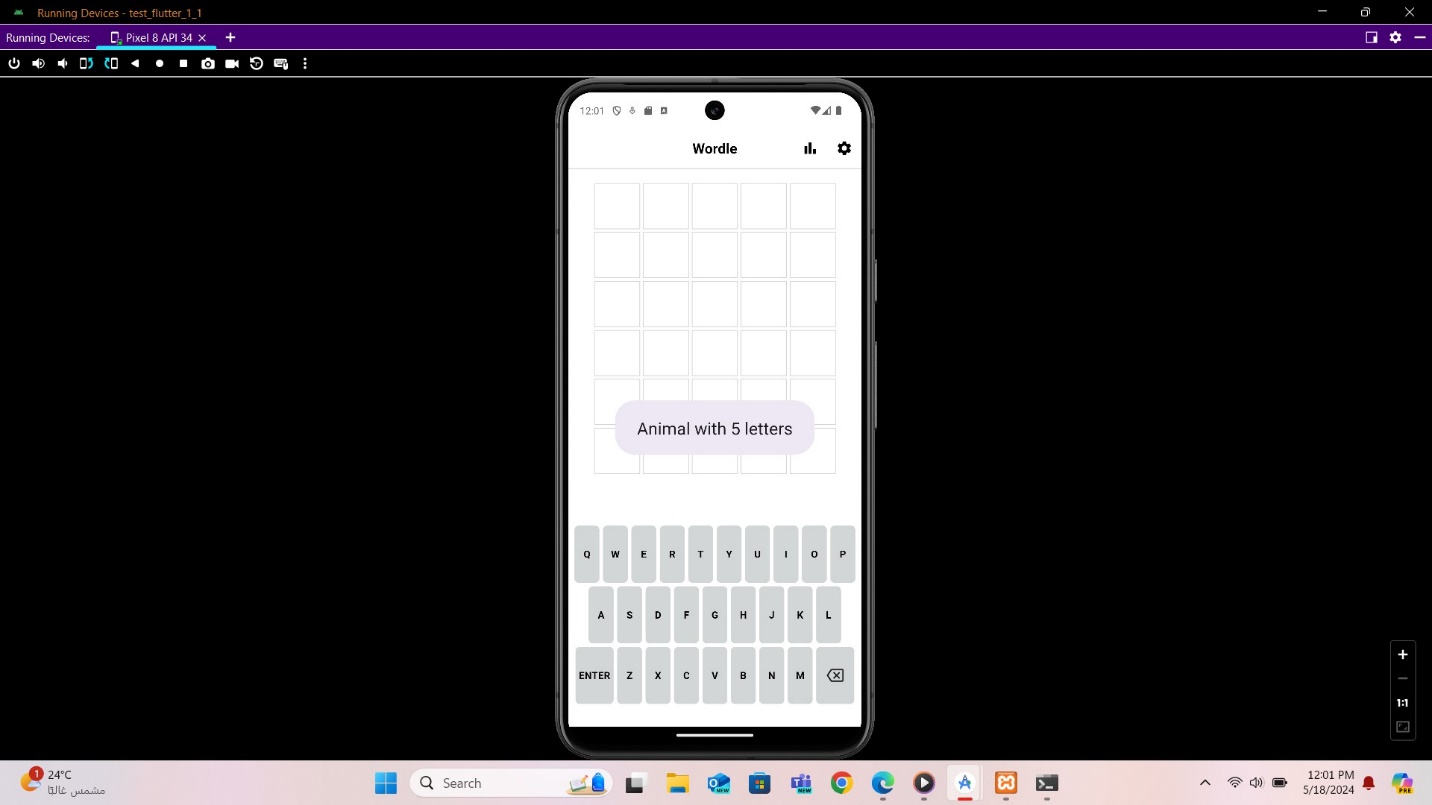
The Python script of the levels exp: (level1.py) handles the word selection process for the level mentioned.

* **Read Data**: The script reads a JSON file containing words and their descriptions.
* **Select Words**: It randomly selects a specified number of words and their corresponding descriptions from the list.
* **Format Response**: The selected words and descriptions are formatted into a JSON response, which includes the status, message, size, words, and descriptions.

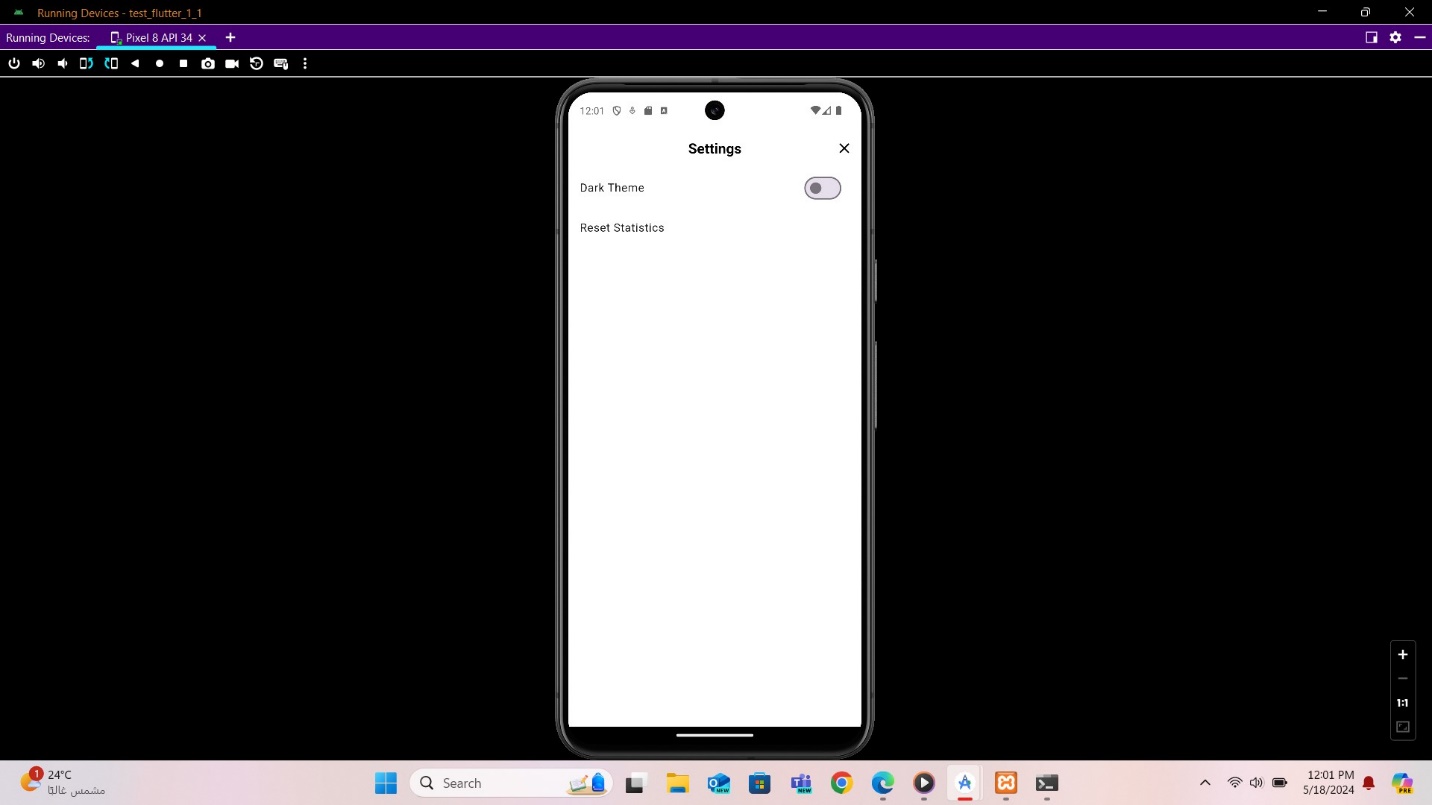
For each level we have a different python script to make it easier to change within each level.

Each new level's Python script (e.g., level2.py, level3.py) follows a similar structure but includes additional categories. For instance, level2.py might handle both animals and fruits, while level3.py could add a third category like countries. The selection process becomes more challenging as the pool of words increases with each new level.

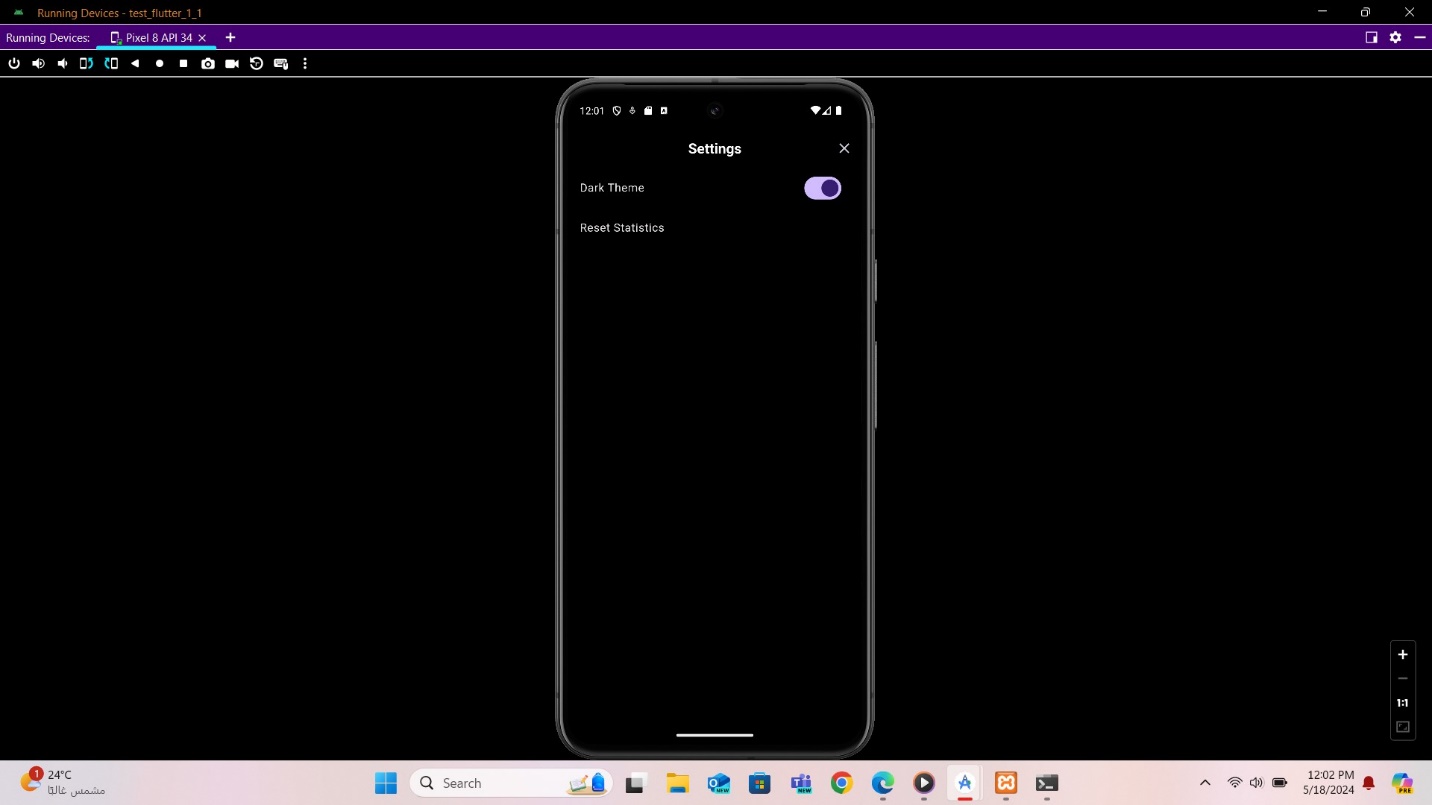
# Screens

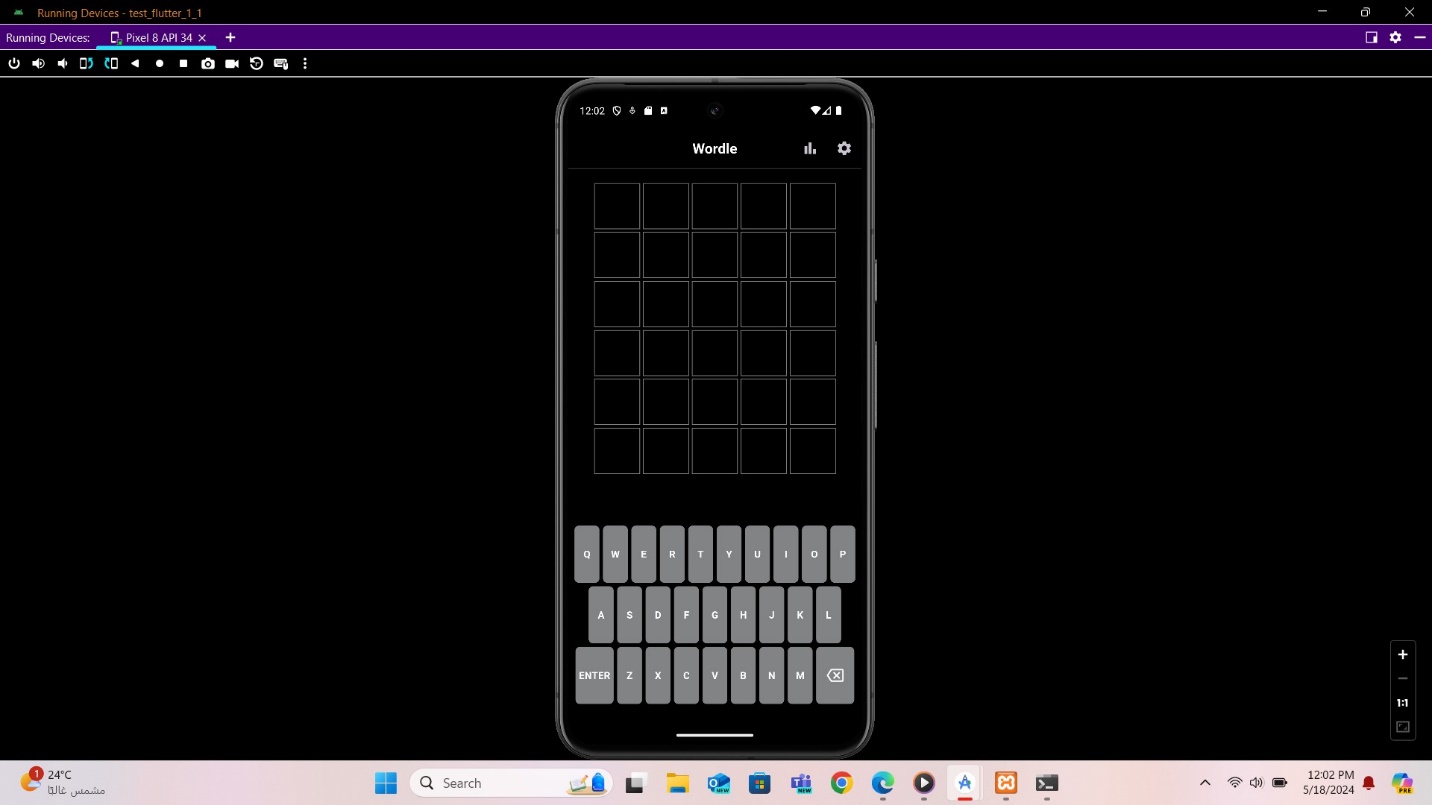


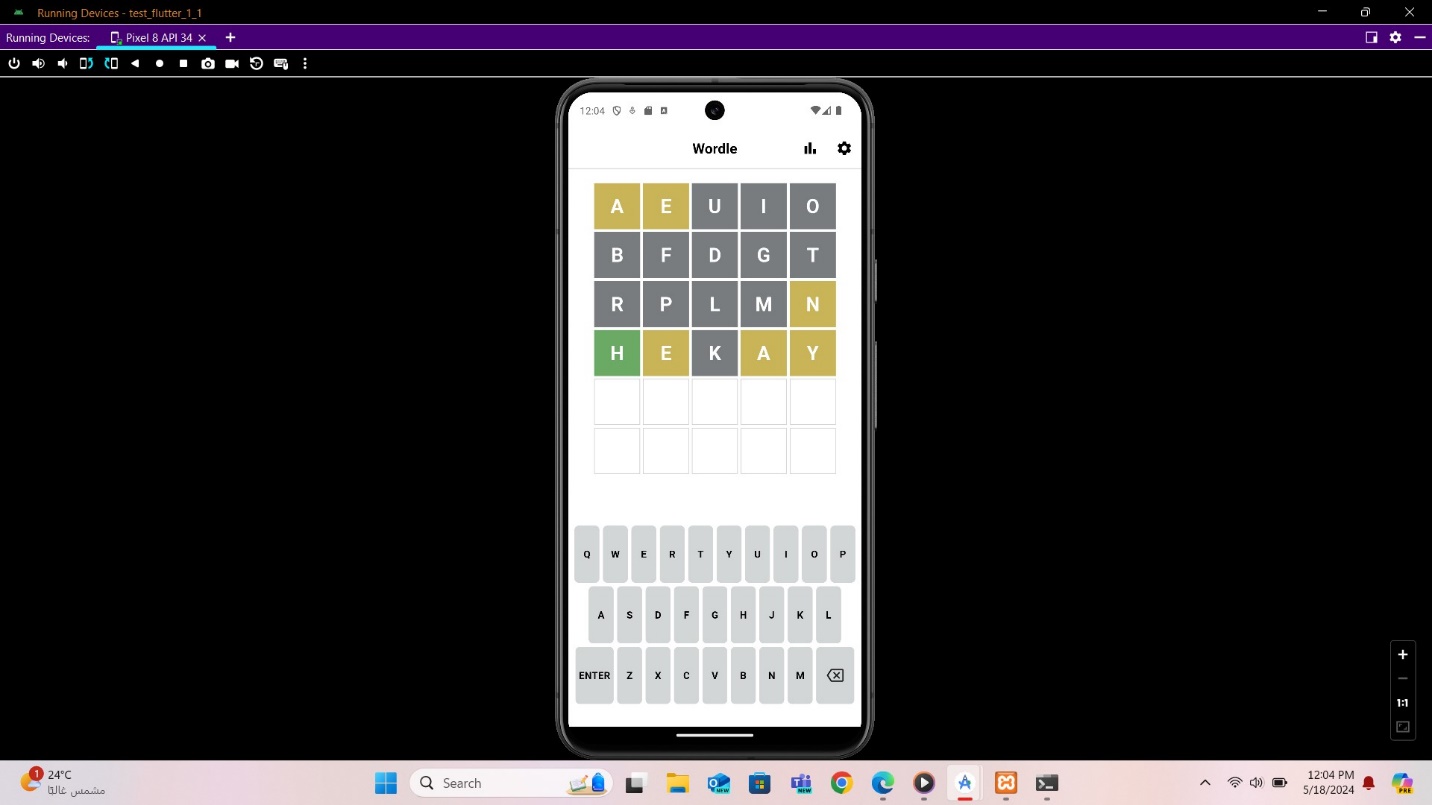
When you open the app, you'll receive a hint about the category and the word you need to guess.



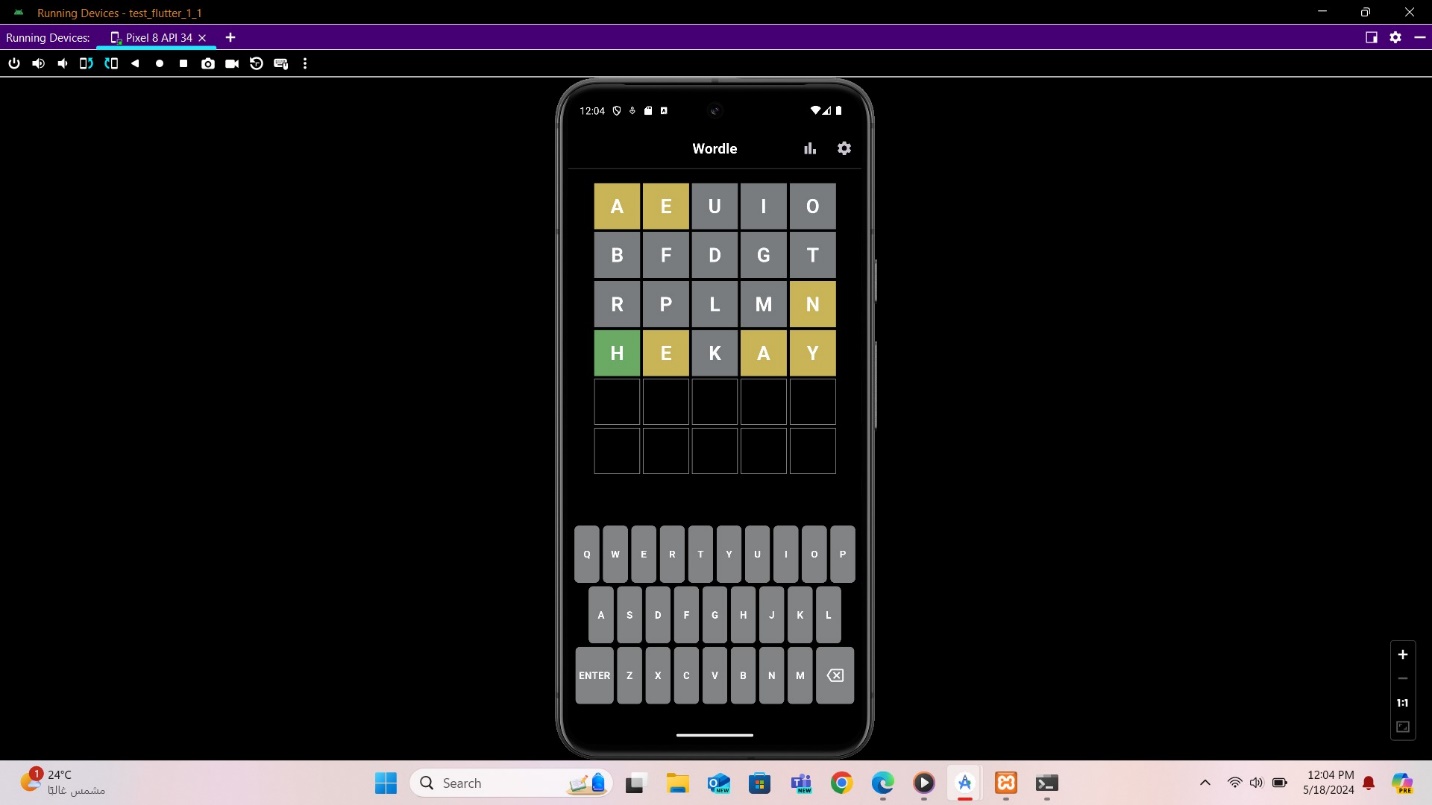
In the app's settings, you can reset your progress to start from level zero or switch the app's theme between light and dark modes.

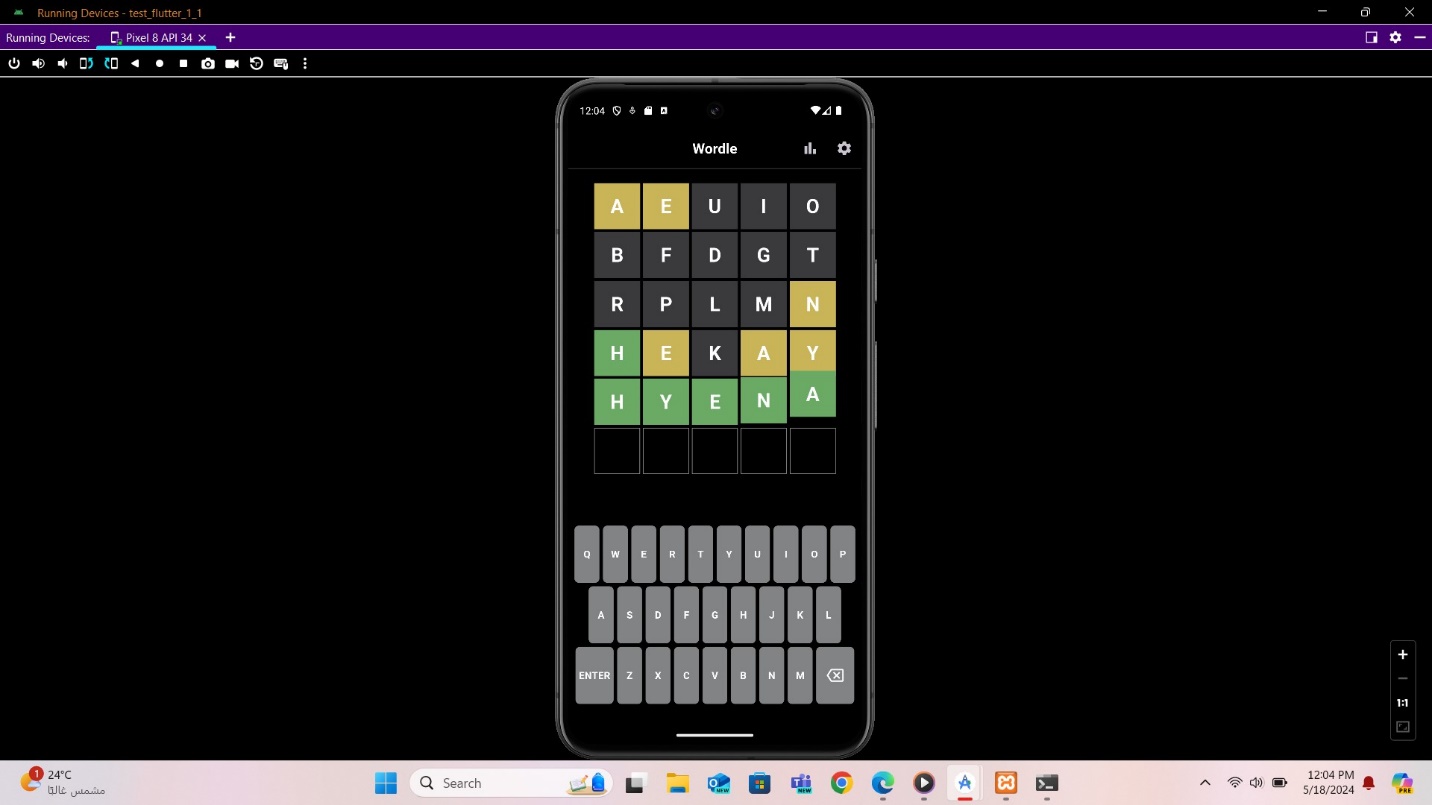


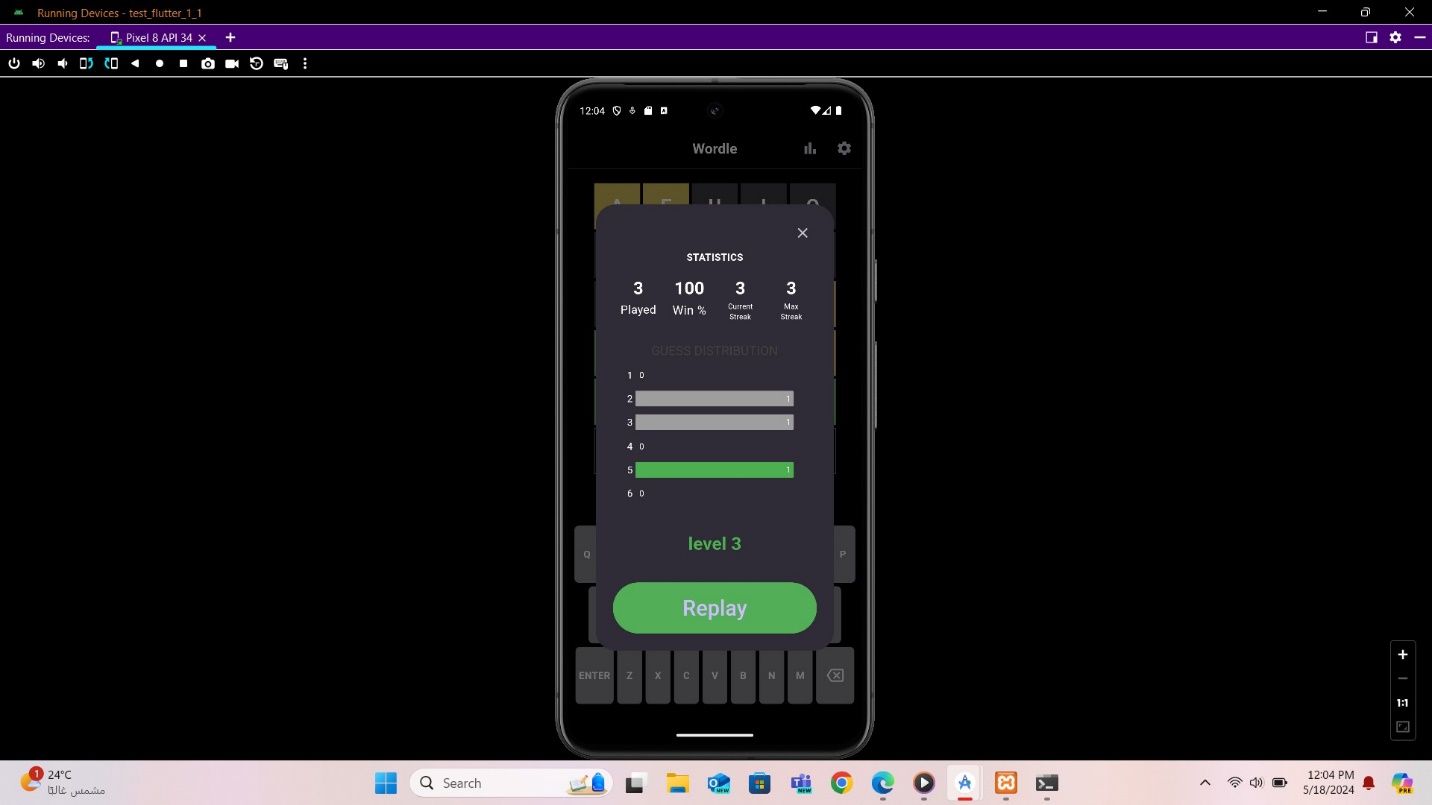




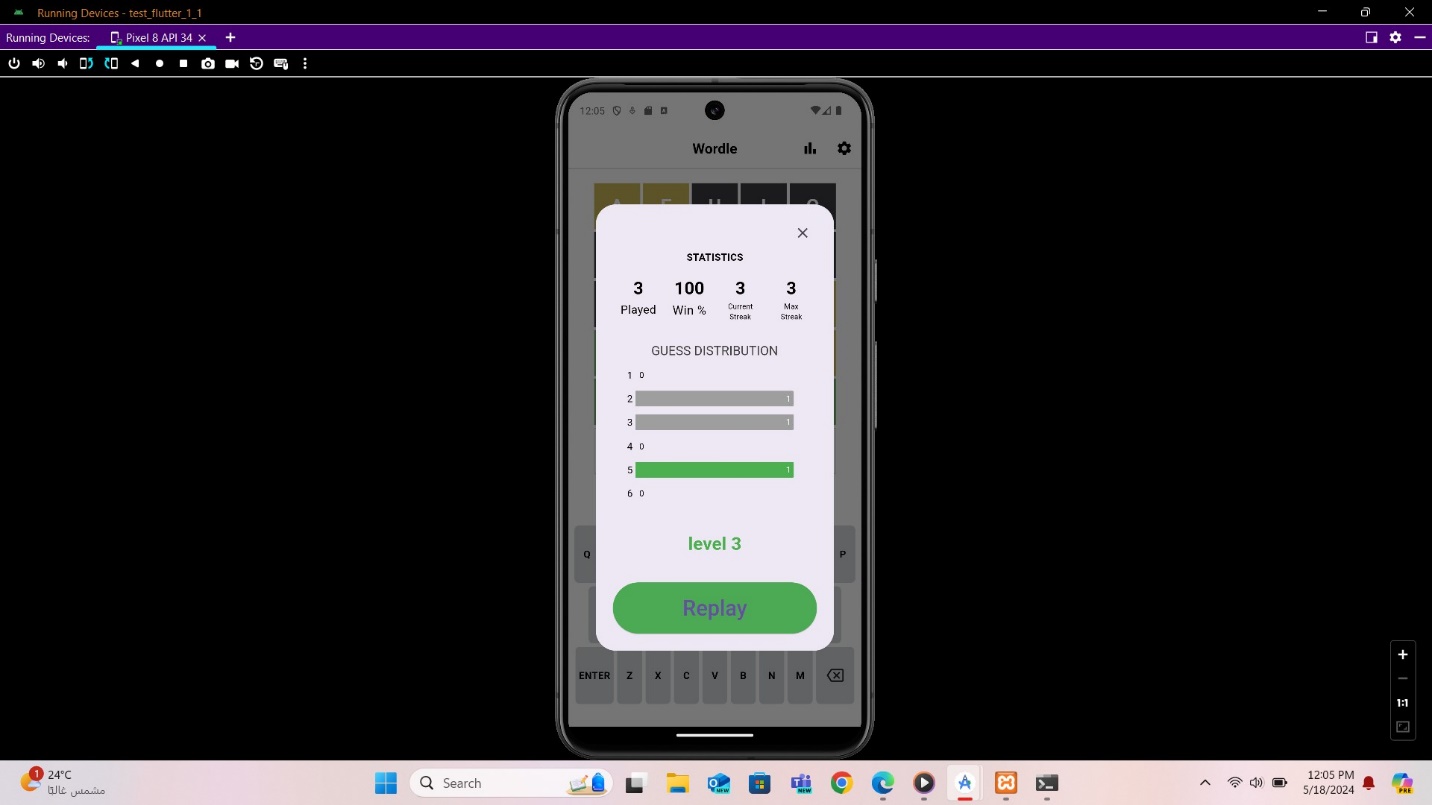
Letters in yellow indicate that the letter is in the word but not in the correct position, while green letters are both in the word and in the correct position. Grey letters mean that the letter is not in the word at all. You have six attempts to guess the word correctly.

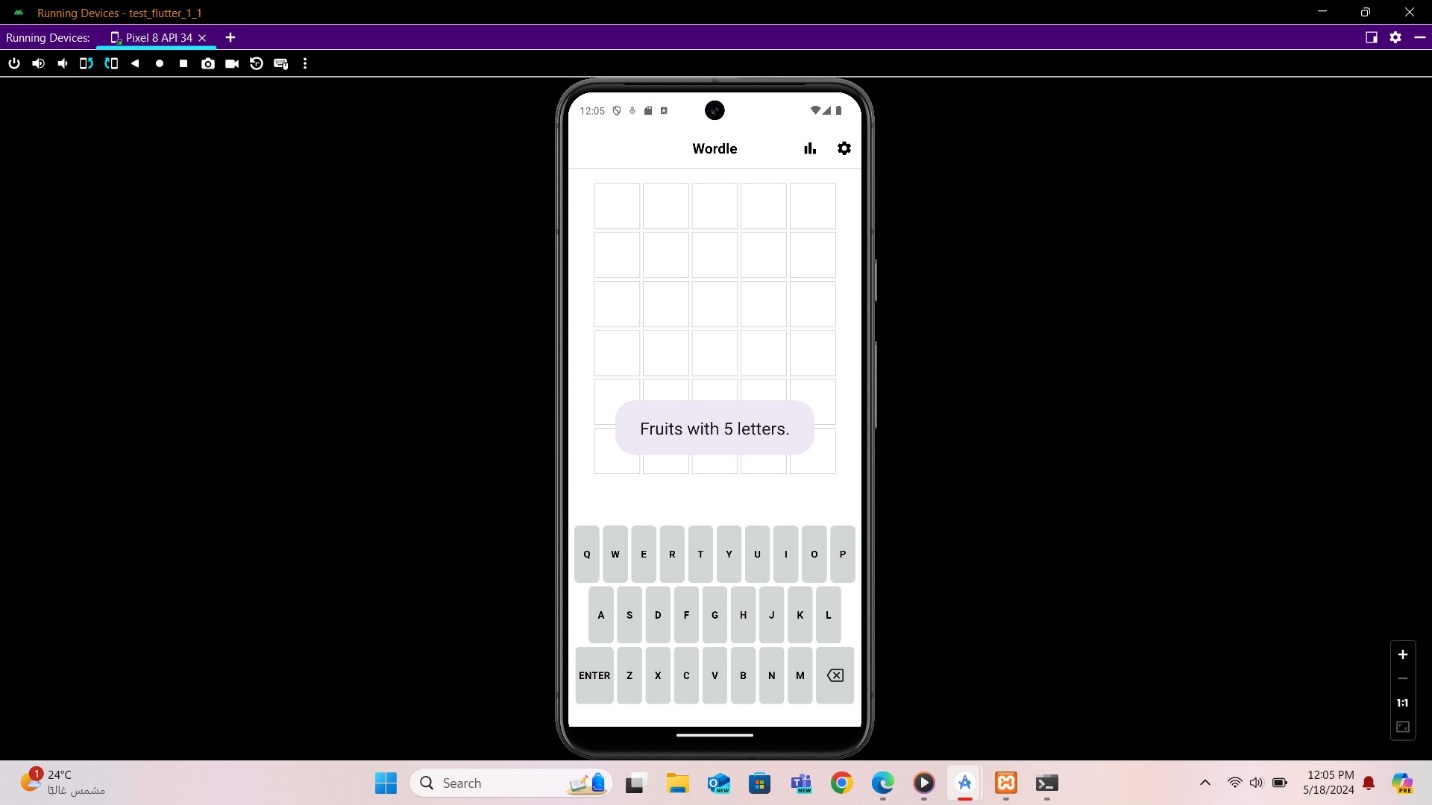






If you guess the word correctly, a statistics screen will display, showing the number of games you've won, your current streak, other relevant information, and your level.





Achieving an 80% win rate will advance you to the next level, where a new category will be added to the previous ones.

# GitHub Link

For full access to the code, please check the GitHub repository at the following link: <https://github.com/Mohammad-Fleity2002/Wordle-Game.git>

# Conclusion

Wordle is further than just a game; It is a carefully designed teaching tool that immerses users in a rich journey of language discovery. Wordle leverages natural language processing (NLP) capabilities and a customizable Flutter framework to provide dynamic word generation across different categories, promoting vocabulary expansion and understandable engagement.

The front-end implementation of Wordle is very powerful and visually appealing. The game grid is rendered using Flutter’s GridView.builder widget, effectively displaying a grid of tiles representing guessable letters. Each tile features animations such as "dance" and "bounce" that provide visual feedback and enhance the gaming experience.

On the backend, Wordle relies on a well-structured database to store words and their descriptions. The API endpoint, implemented with a PHP script, dynamically retrieves word data during gameplay, enriching the player experience with diverse word categories.