GitHub Username: leoberteck

What The Word?

Description

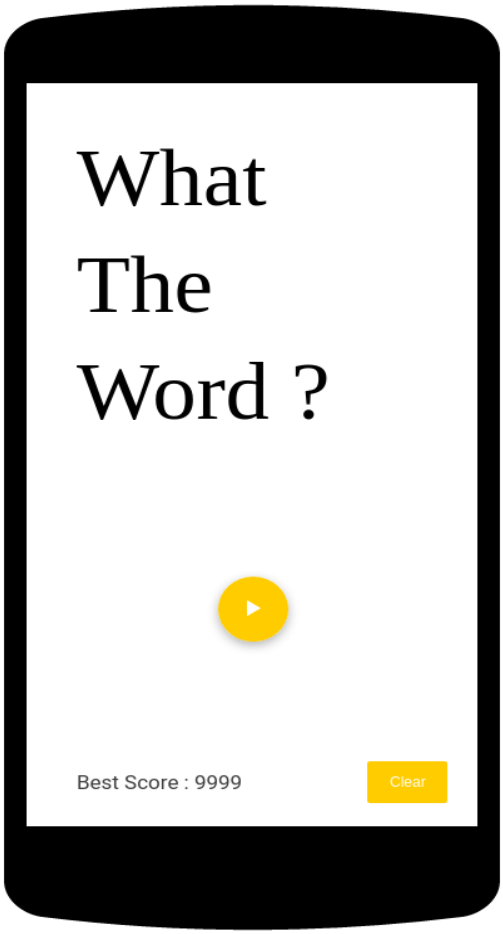
A game that challenges users to find a hidden word based on its definition. The user has 5 attempts to figure out the word, or else, the game is over. The score improves the less attempts the user makes to find the word.

Intended User = Everyone.

Features

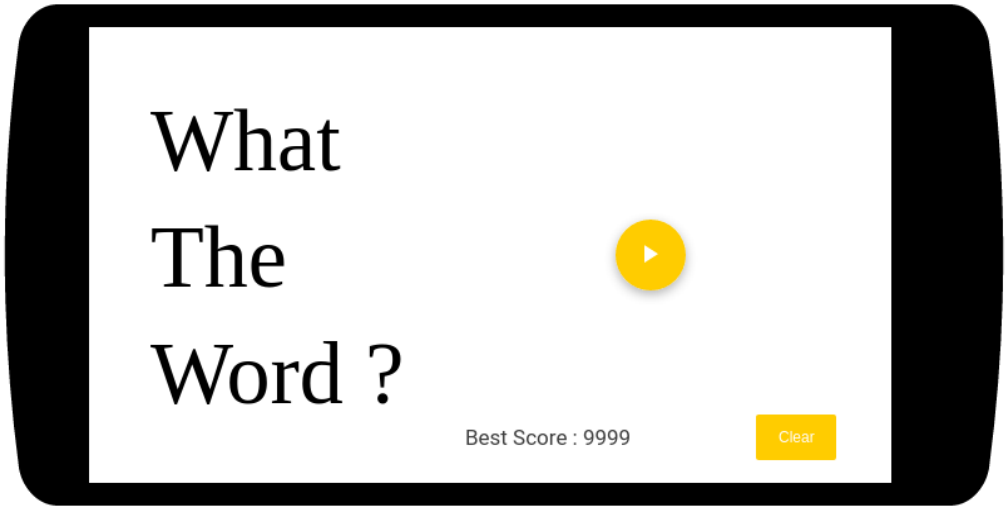
- The app will save the user progress if it leaves the app without finishing a game.
- The game will provide a leaderboard for users that log in with their google play games accounts.

User Interface Mocks



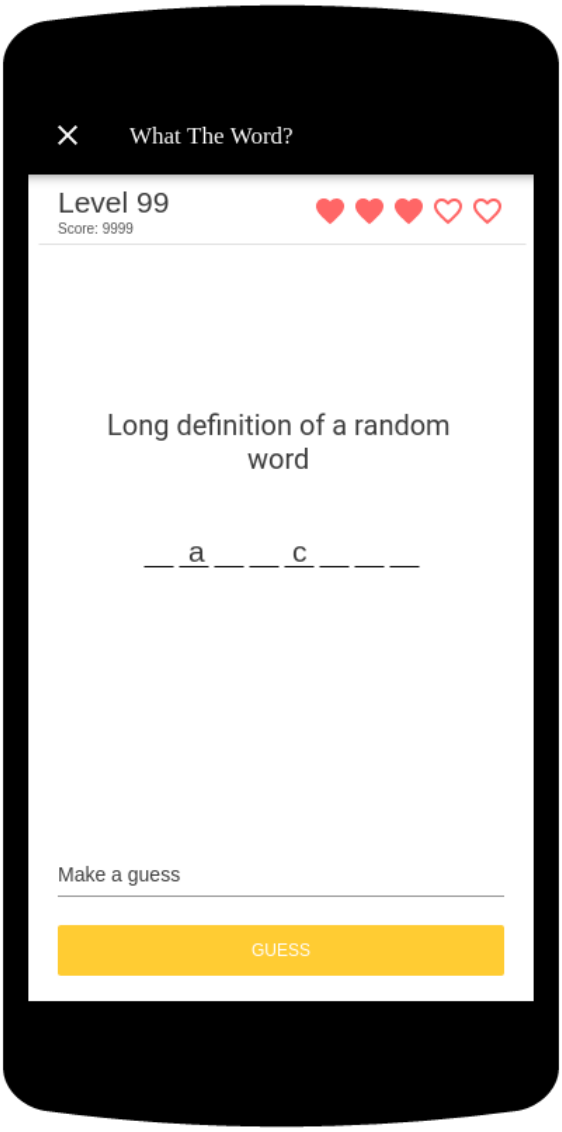
Screen 1

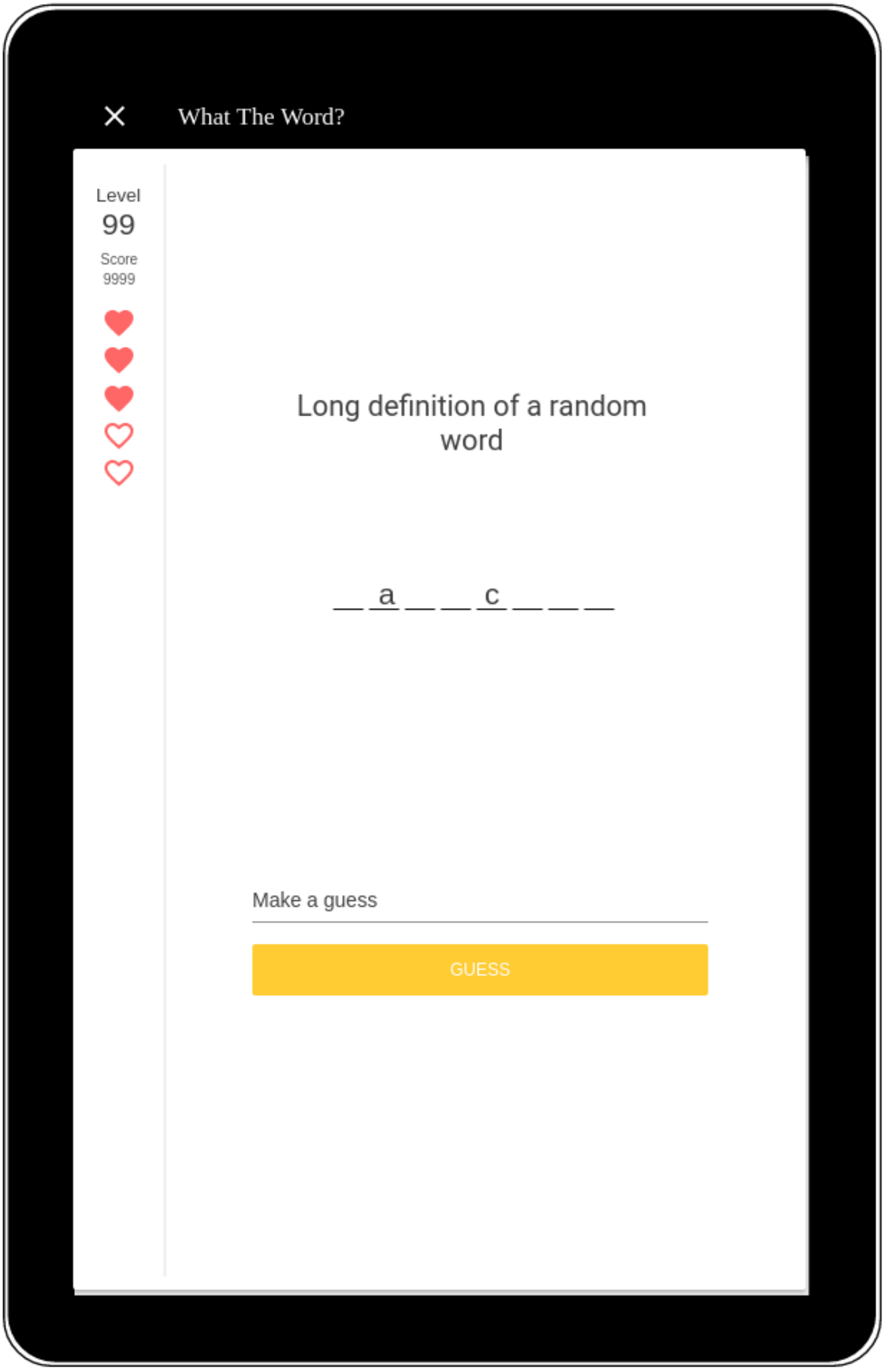
The first screen is the Title Screen, It presents the name of the game and the user's best score.

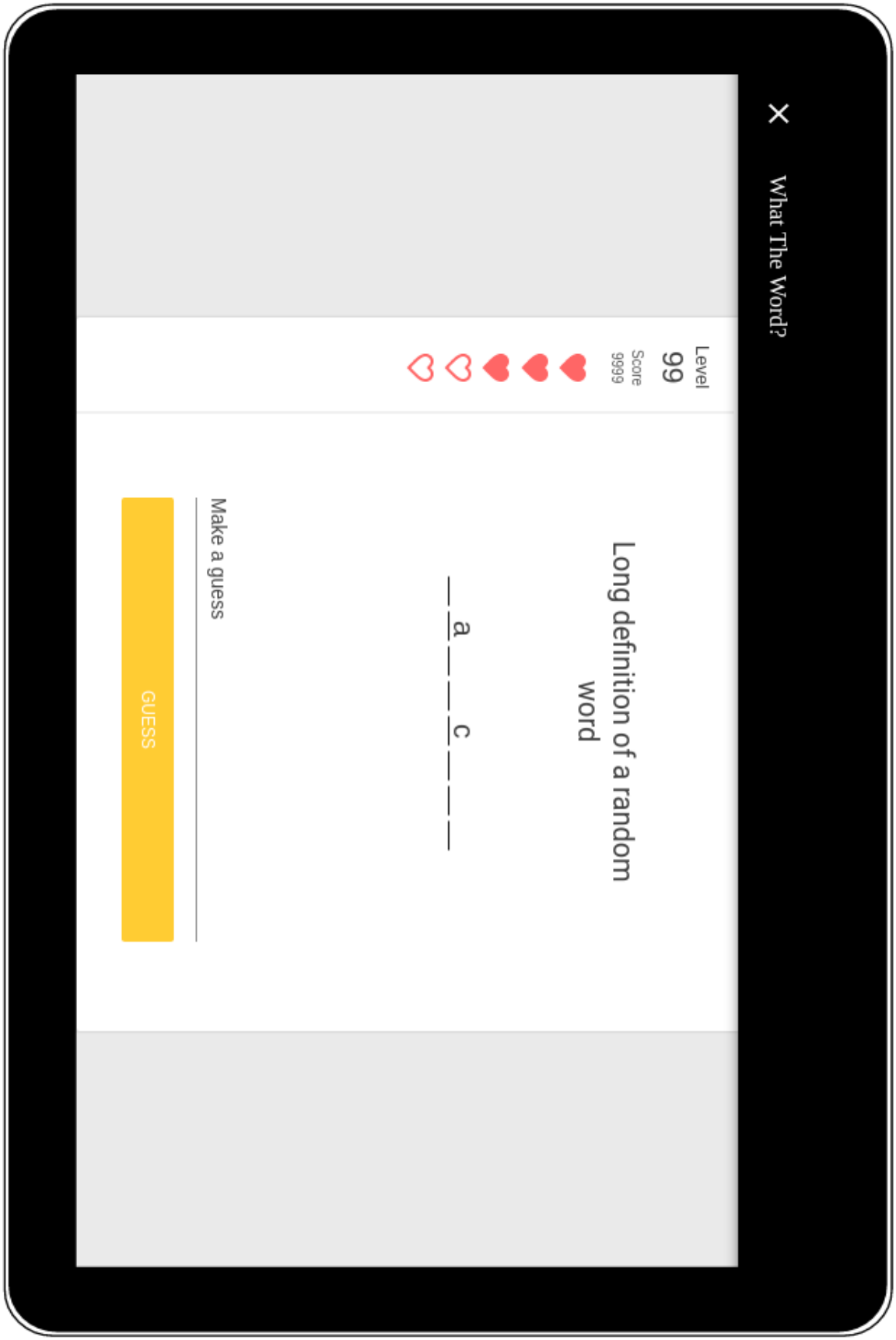


Screen 2

The game screen, for smartphones, this screen will only support portrait orientation, and for tablets, it will support both portrait and landscape orientations.




What The Word?
Level
99
Score
9999
Long definition of a random word
__a___c____
Make a guess
GUESS


What The Word?
Level
99
Score
9999
Long definition of a random word
__a____c_____
Make a guess
GUESS

Key Considerations

How will your app handle data persistence?

I intend to have a limited set of words, and provide them through a Google Cloud Endpoints API. The Application will save the user best score using the SharedPreferences provided by Android.

If the users leaves the app during a game, the game state will be saved in a SQLite database. Only one game state can be saved at a time. The game state must be retrieved through a content provider.

Describe any edge or corner cases in the UX.

There is basically two screens: the main screen with the game title and the game screen. The user can leave the game at any time and go back to the title screen through the confirmation in a dialog.

Describe any libraries you'll be using and share your reasoning for including them.

Databinding

- The Android databinding library will make it easier to update the UI.

ViewModel

- I will use the ViewModel library to manage the presenter instances that will preserve the view states.

JavaRx

- The Reactive Java will replace the async tasks and provide a easier way to query the api in the background thread.

Describe how you will implement Google Play Services or other external services.

The google play services will be used to log the user into the google play games account and record the user best score. With this information the application will be able to show the leaderboard activity.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

1. Ghater a list of at least 50 words and their definitions.
2. Create a android project with a Android module. This project will contains all views.
3. Add a java module. This module will contain all game logic and will query the google endpoints api.
4. Add a google endpoints module. This module will provide words and their definitions through a single endpoint that will return a random word from the list of words.

Task 2: Implement Google Cloud Endpoints module

1. Implement an endpoint that receives a list of word ids that must not be returned and returns a random word from the list with an id that is not contained in the received list.

Task 3: Implement game logic.

1. A single game starts when the user hits the “play button” on the title screen and ends when the user waste 5 guesses on a single word and still cannot get it.
2. Everytime the user guesses a word wright, his wasted guesses are replenished, i.e., every time the user navigate to the Game Activity, it has 5 chances to try to guess the hidden word.
3. The score is based on the number of wrong guesses the user had and it is as above:
    a. No wrong guesses : 10000 points
    b. 1 wrong guess : 5000 points
    c. 2 wrong guesses : 2500 points
    d. 3 wrong guesses: 1250 points
    e. 4 wrong guesses: 625 points
    f. 5 wrong guesses: 0 points - game over.
4. Every time the user guess a word wrong, a single random letter of the hidden word must be shown.
5. Write unit tests to verify the game logic implementation.

Task 4: Implement UI for Each Activity and Fragment

1. Build the ui for the title activity. This will be the main activity and must contain the app name in Serif font, the best score of the user, if it exists, and a “play button” that will start a new game.

2. Build the ui for the game activity. The game activity receives a word and its definition, it has three alternative layouts:
    a. The phone portrait layout
    b. The tablet portrait layout
    c. The tablet landscape layout
3. There must be an animation transition between the title activity and the game activity.
4. There must be an animation transition everytime the users advance in the game.
5. When the user guess the word wrong, a single filled heart must be replaced by an empty heart with an animation.
6. If the user tries to leave the game, a confirmation dialog must show.
7. When the user guess a word right, a congratulation dialog must show.
8. When the user beat's it's personal best score a congratulation dialog must show.
9. If all hearts get empty, the game is over and a game over dialog must show.
10. Implement Espresso test to verify that the title page shows the user best score if it exists
11. Implement Espresso test to verify that the game screen does not crash if a word is not informed.
12. Implement an Espresso tests to verify that all dialogs are in the proper ocasion.

Task 5: Implement game state save and retrieval

1. Implement a ContentProvider that can save and retrieve a single instance of the game state.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"