Federal State Autonomous Educational Institution
Higher education
National Research University Higher School of Economics

Faculty of Computer Science
Educational program “Data Science”

Contexto Game
(Project Name)

Performed by student.

Anwar Ibrahim
(Full name)

_____
(Signed)

Project Manager:

Dmitry I. Ignatov
(position, full name of the project manager)

_____
(assessment)

_____ _____
(Date) *(Signed)*

Moscow 2023

Anwar Ibrahim

My contributions to the Contexto Project:

In this project, I was mainly responsible for creating the Graphical User Interface. To implement this game, I am going to use python programing language “Python 3.10.10”, and to create the graphical user interface “GUI” I will use the open-source python app framework “Streamlit” (streamlit library, n.d.).

To implement this task, I decided to split the application into two classes, a wrapper class “Game” and an “Application” class.

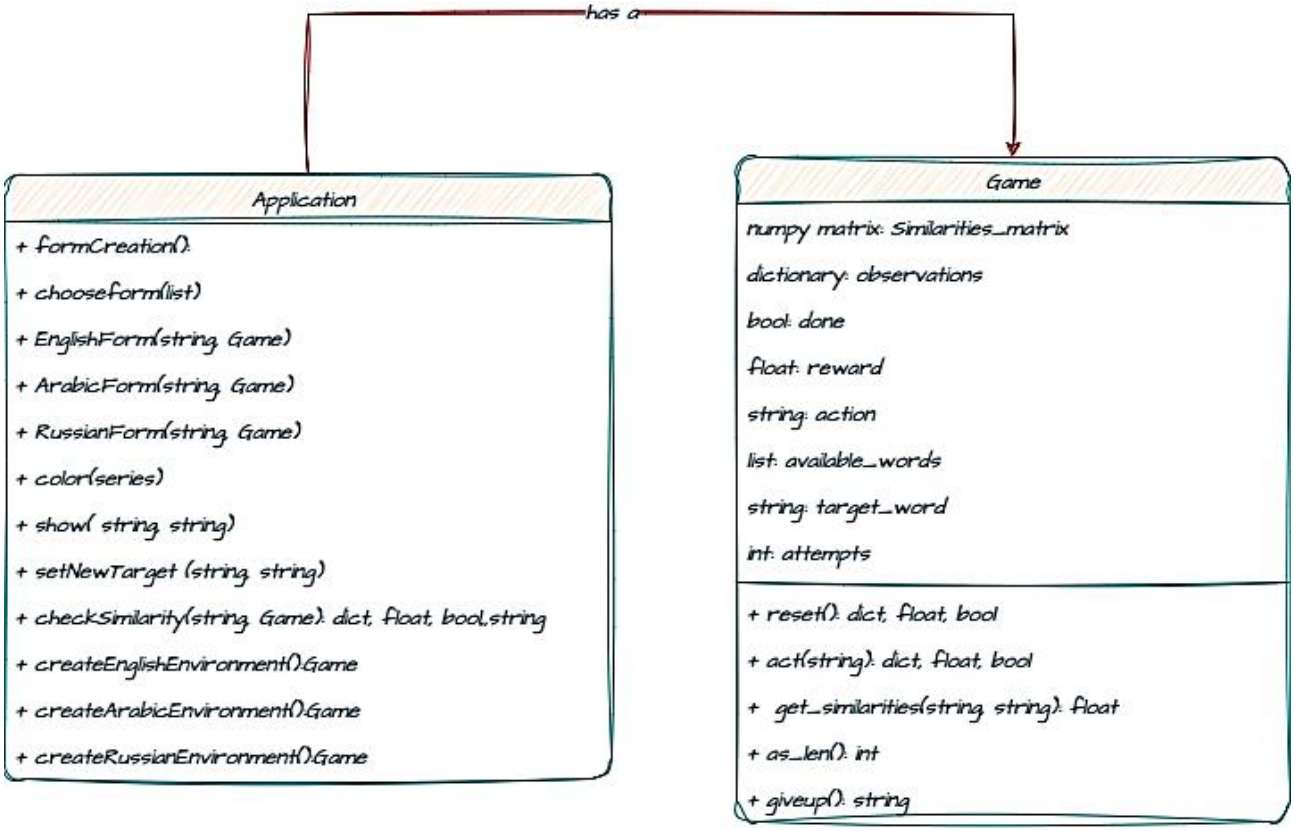

Figure 1: class diagram of the programs structure.

The previous figure, shows the initial design of the program, and this design was inspired by the Reinforcement learning game libraries, so I tried to implement all the methods that we could need from the game environment such as:

- ***act(string):*** this function is responsible for interacting with the environment, and it outputs the similarity value between the target and the inputted guess from the player, and a Boolean to check if the player had guessed the secret game, and finally a new target chosen randomly from the list of available words that we provided.

This wrapper was then given to my team mates to wrap their codes in it, and this class got changed to suit their needs.

The second class is the core of my work, which is creating the GUI, I was inspired by user interface of the main game of Contexto (Group, n.d.)

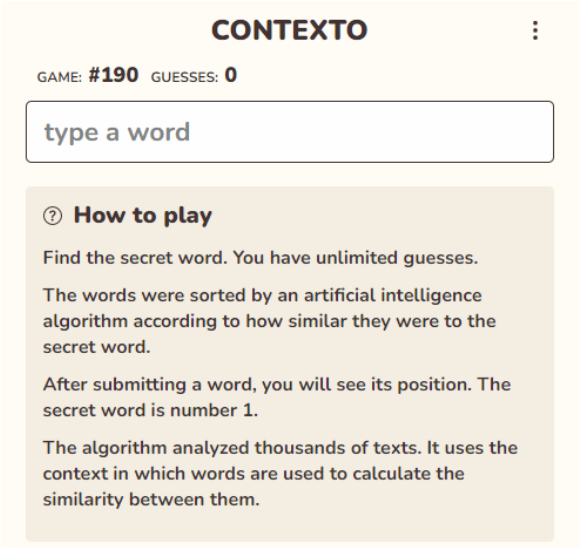

Figure 2: the user interface of the Contexto game. (Group, n.d.)

I tried to mimic their design, because I believe it servers the purpose of the game well, and I tried to make my code as generic as possible, we only need to change a few lines to make the code work for any other language.

Sequence diagram:

The following figure shows the sequence diagram of the study case of a player interacting with our application:

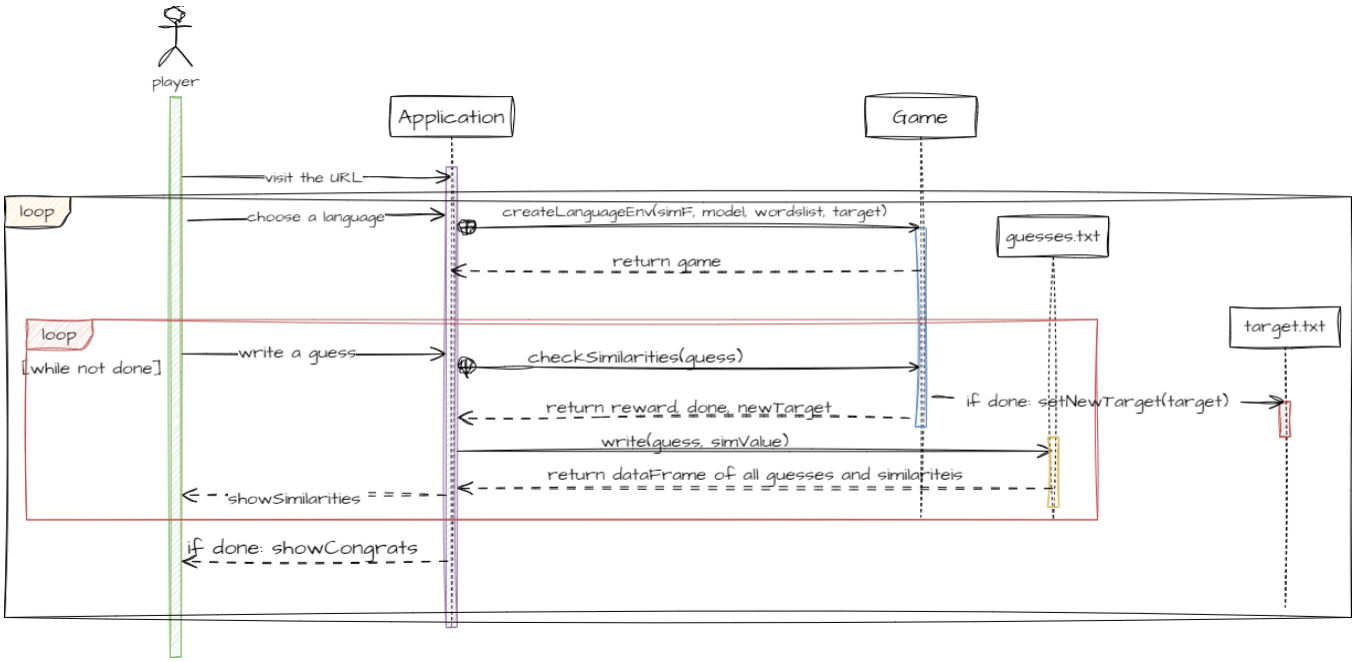

Figure 3: sequence diagram of a player interacting with the game.

Deploying the game:

There are two ways to deploy our application for players to play with it:

- github
- Google Colab

I am going to use the Google Colab method, and this code is explained thoroughly in the file "DeploymentColab.ipynb" running this file alone, will give you the opportunity to play our game, using this method allows different type of players to interact with our game, normal players that are only interested in the game "they can just visit our link and start playing" and experts who are more interested in checking the code "they can just rerun the given colab code and enjoy checking the code and playing at the same time".

The difficulties:

I encountered several difficulties, since this is my first time working with the streamlit environment, so it was challenging and interesting to work with this library, some of my difficulties:

- DuplicateWidgetID, one of the most common problems that one can encounter when working with streamlit library, and this error means that there is an internal Key that identifies each widget one adds to the final form "all the values that comes from this widgets will be stored at the same address in the RAM", and since we are using only one text box to get the input form the user, I had difficulties storing the guessed words because when we get a new input the previous one will be replaced, and I solved this problem by writing each guess on the txt file "guesses.txt", then reading this file using the function show(), after each guess.
- Setting the target when we initialize the Game environment, for some reason Streamlit library reidentifies the game environment whenever the player interacts with the GUI, this means the target will be changed every time the player interact with the environment, and to solve this problem, I decided to store the target for each language in a txt file, and when the player successfully guess the secret word, the environment will randomly choose a new word from the given "available words list" and the new value replace the current target as the new secret word.
- Insufficient Resources: at the beginning, I was implementing the English Language model using the Gensim "GoogleNews-vectors-negative300" model, but this model was taking a lot of resources almost 5 Gigabyte RAM, and 3 to 4 minutes to run on my laptop, on the other hand it was taking

almost 2 minutes to run on the Google Colab platform, and waiting this long will cause any player to get bored and lose interest, so as suggested by one of my teammates I created a similarity matrix using the Google model, which stored the similarities between each two available words from the available words list, and creating this matrix alone was taking almost an hour on the other hand it was speeding up the execution from 2 minutes to one maximum, but this method was not the best solution, because after viewing my teammates updates, and the way they implemented their wrappers, I decided to choose a bert based model "*bert-base-multilingual-cased*" like they did, the execution was noticeably faster, and it needed far less resources to run, it also meant that we can use the same wrapper for the three languages with only changing the name of the model.

References

[1] Group, S. N. (б.д.). *Contexto*. Получено из https://contexto.me

[2] *streamlit library*. (б.д.). Получено из https://streamlit.io