WordVoyage

Parts

Login/Signup

Dashboard/Profile

Learn/Quiz

Translate/Progress

Database

Auxilliary

LOGIN/SIGNUP

1. Splash screen

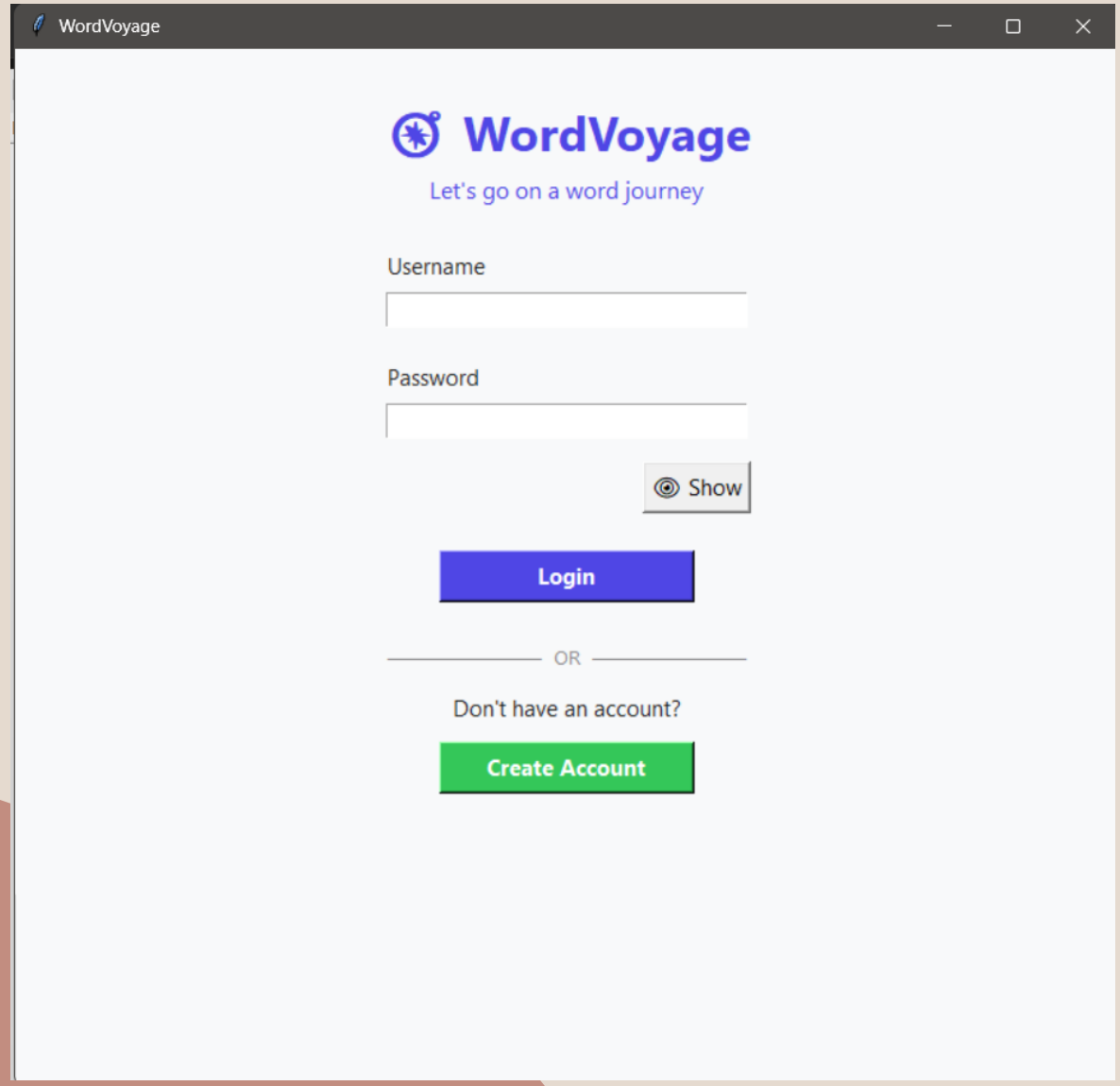


1. Done after whole project was done.

2. Uses pillow instead of tkinter -> less complicated and easier to implement

LOGIN/SIGNUP

2. Login screen:



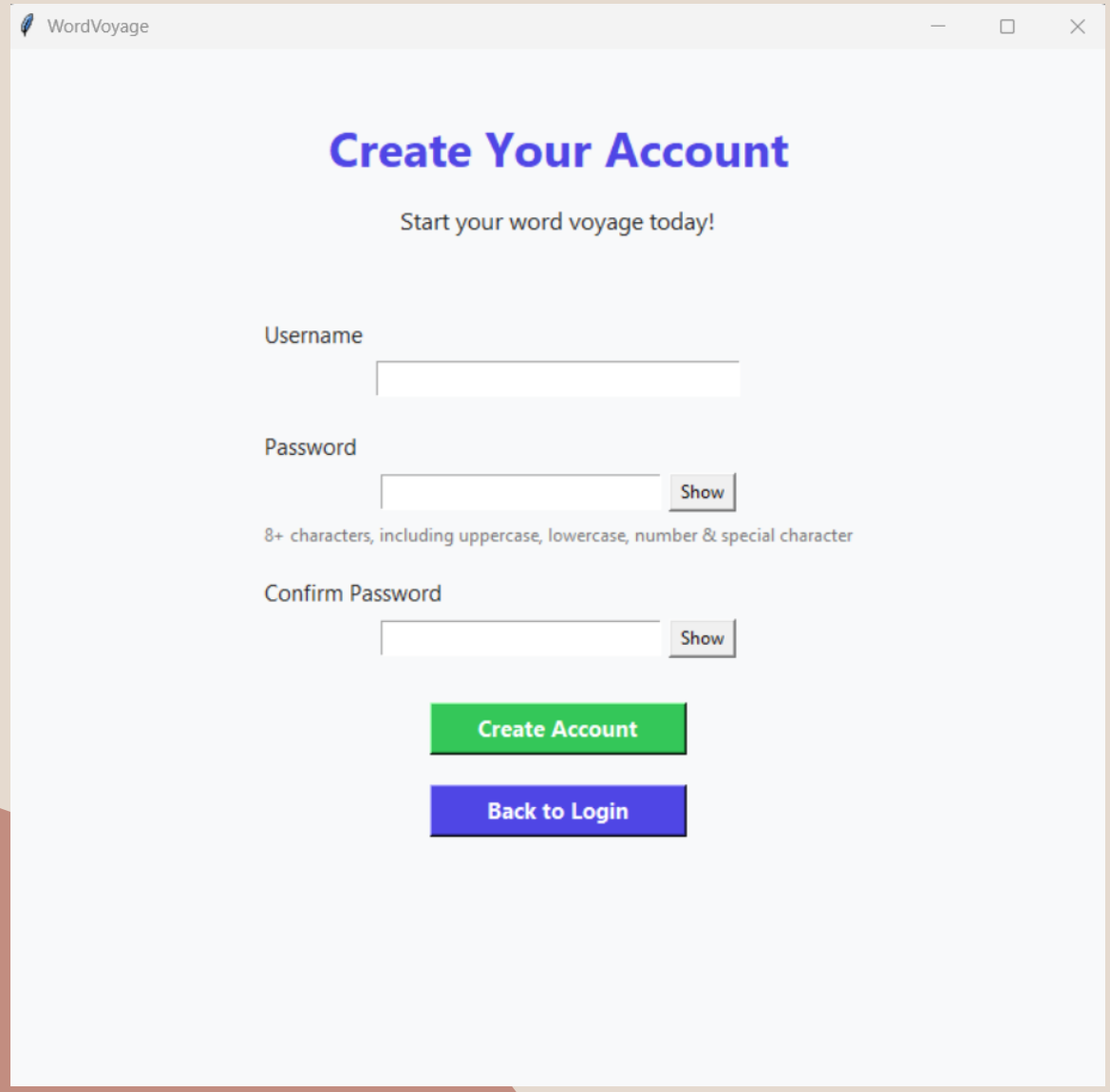
1. title, subtitle, alignment of elements done using tkinter coding

2. *Username and Password*: compares user given info on the text boxes with the table "Users" over the Admin database, if input's are correct, dashboard appears, else a pop message alert appears stating the username/password is wrong.
   If data is not entered in either or both boxes, shows a pop message stating that username/password is not entered.

3. *Toggle button:* toggles from Show/Hide *(done by using def function and if and else within it)*

LOGIN/SIGNUP

2. Signup screen:



1. title, subtitle, and all elements placed and aligned using tkinter coding.

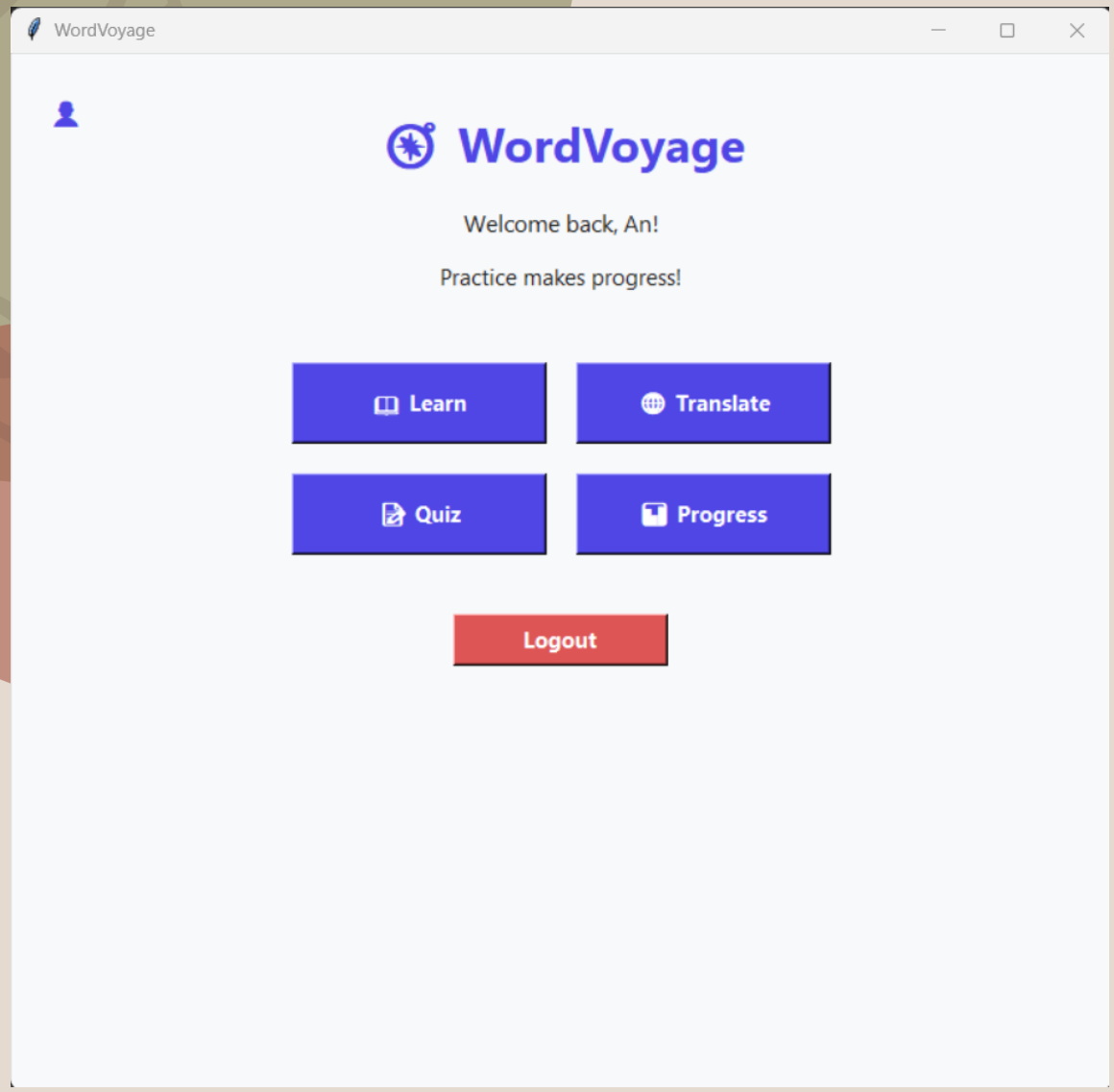
2. Username: checks whether user input already present in “Users” table, allows only if it isn’t.

3. Password: should be a min. of 8 characters, should have atleast one number, one uppercase, one lowercase, and one special character. If anyone of the criteria not fulfilled, it will give a pop-up alert for the same.

4. Create account: Once clicked, data given by user gets saved over the “Users” table and the user can now go back to login and login in through these credentials.

DASHBOARD/PROFILE

1.DASHBOARD:



1. Connecting login window with different segments of the application.

2. contains profile and logout button (logout button leads back to login window

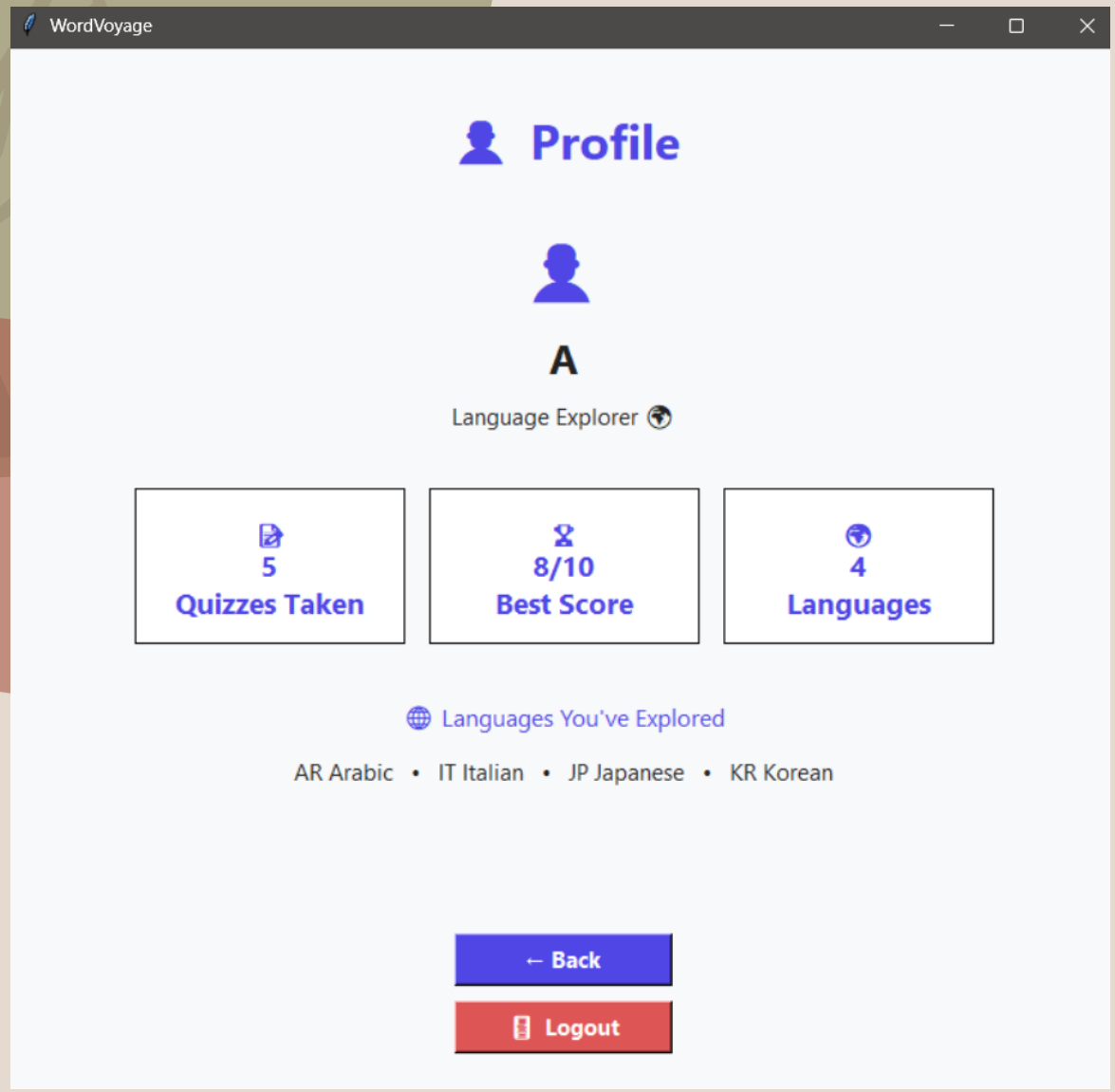
3. “Welcome back <username>” uses parametrized query.

4. The line below the greeting line changes with each opening.
   BTS, python randomly chooses a greeting from a given list in the code.

```
# Random greetings
self.greetings = [
    "What would you like to do today?",
    "Ready to learn something new?",
    "Let's continue your language journey!",
    "Practice makes progress!",
    "Every word you learn opens a new door."
]
```

DASHBOARD/PROFILE

2. PROFILE:



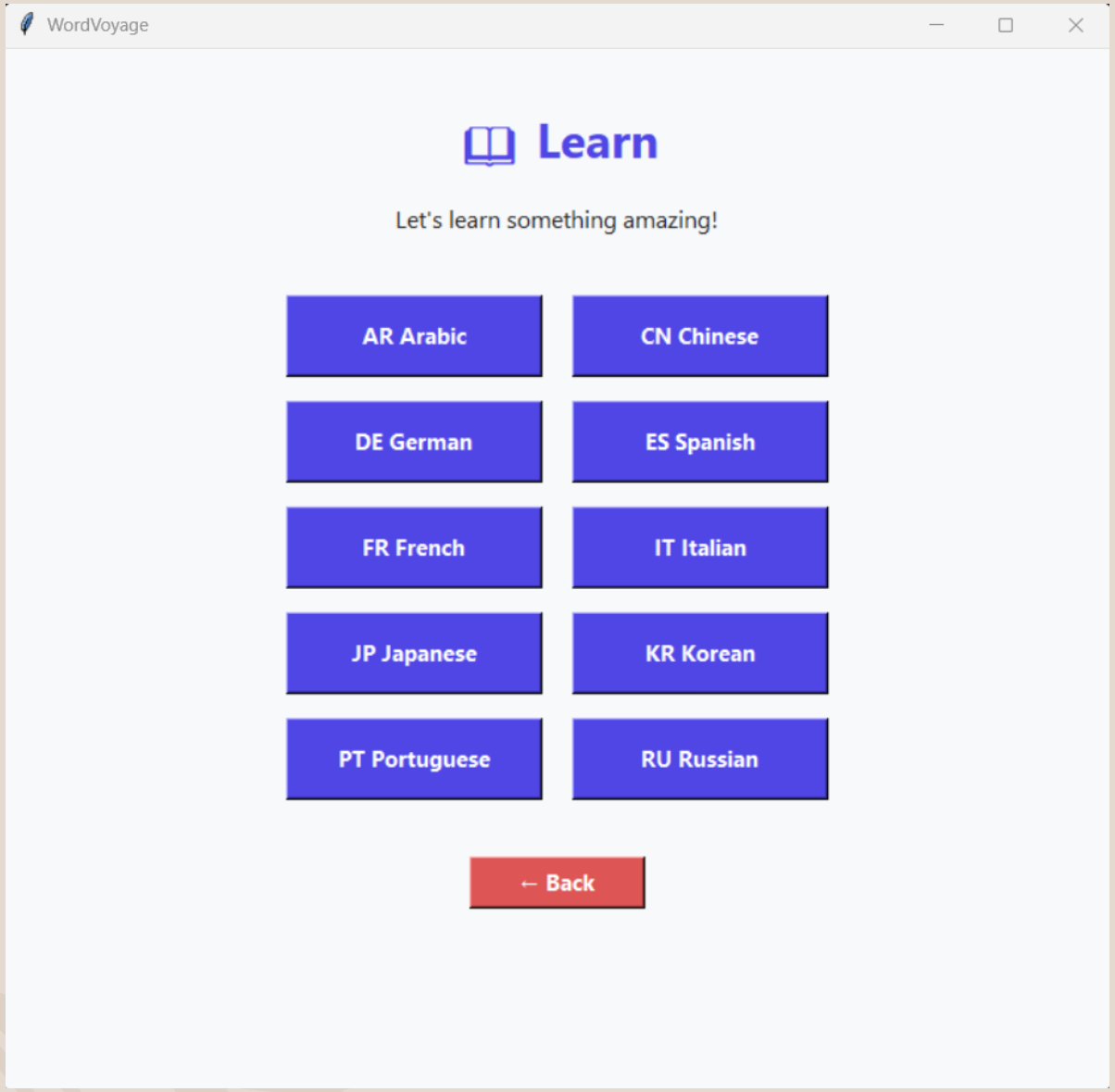
1. Shows statistical data gathered from the table “quiz_results”

2. Languages You’ve Explored section: shows the languages you have practiced through quizzes and not counting the flashcards you have gone through.

3. Logout takes back to login window

Learn/Quiz

1. Learn window (choose a language)



(ref. learn.py)

1. Shows the 10 languages currently registered in the "flashcards" table in the database.

2. Choosing any one language leads to the categories window (it's has the same layout and categories for all languages)

3. Back button leads back to dashboard

4. Languages are in ascending order.

Learn/Quiz

2. Learn window (choose a category)



(ref categories.py)

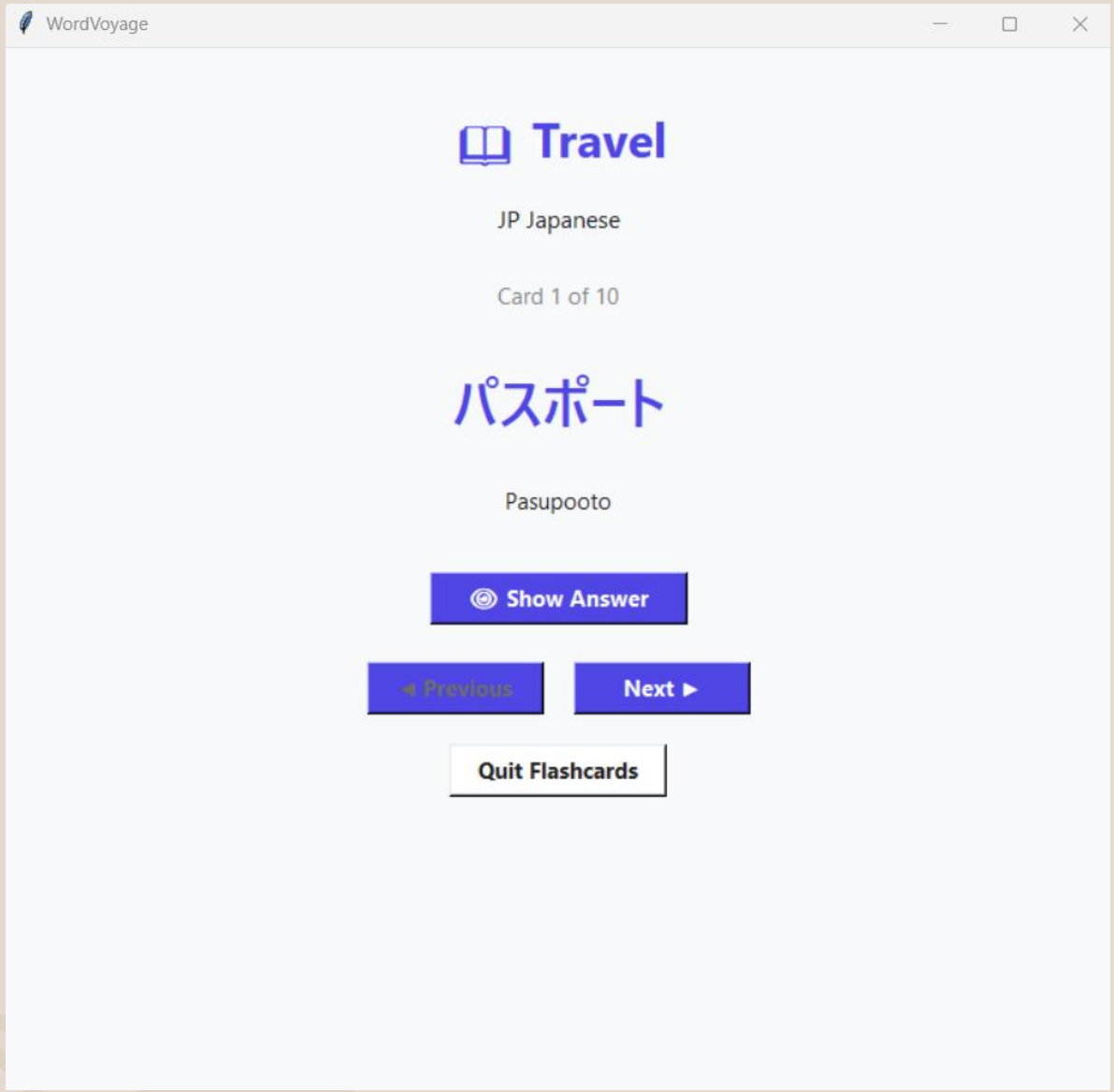
1. The same 6 categories to choose from from all the languages alike.

2. Clicking anyone of the categories loads up a lineup of 10 flashcards for that language-category combo, the selection takes place through a query written within the python code to “flashcards” table.

3. The back button takes back to the learn window.

Learn/Quiz

3. Learn Window (Flashcards interface)



(ref flashcards.py)

1. First line is the category, second line is the language.

2. Card Counter: "Card x of 10", within the python code, this counter starts from 1/10, if next button is clicked, increases by one, if previous button clicked, decreases by one.

3. The text is written in the respective language script, while adding the words to the "flashcards" table, MySQL Workbench was used.

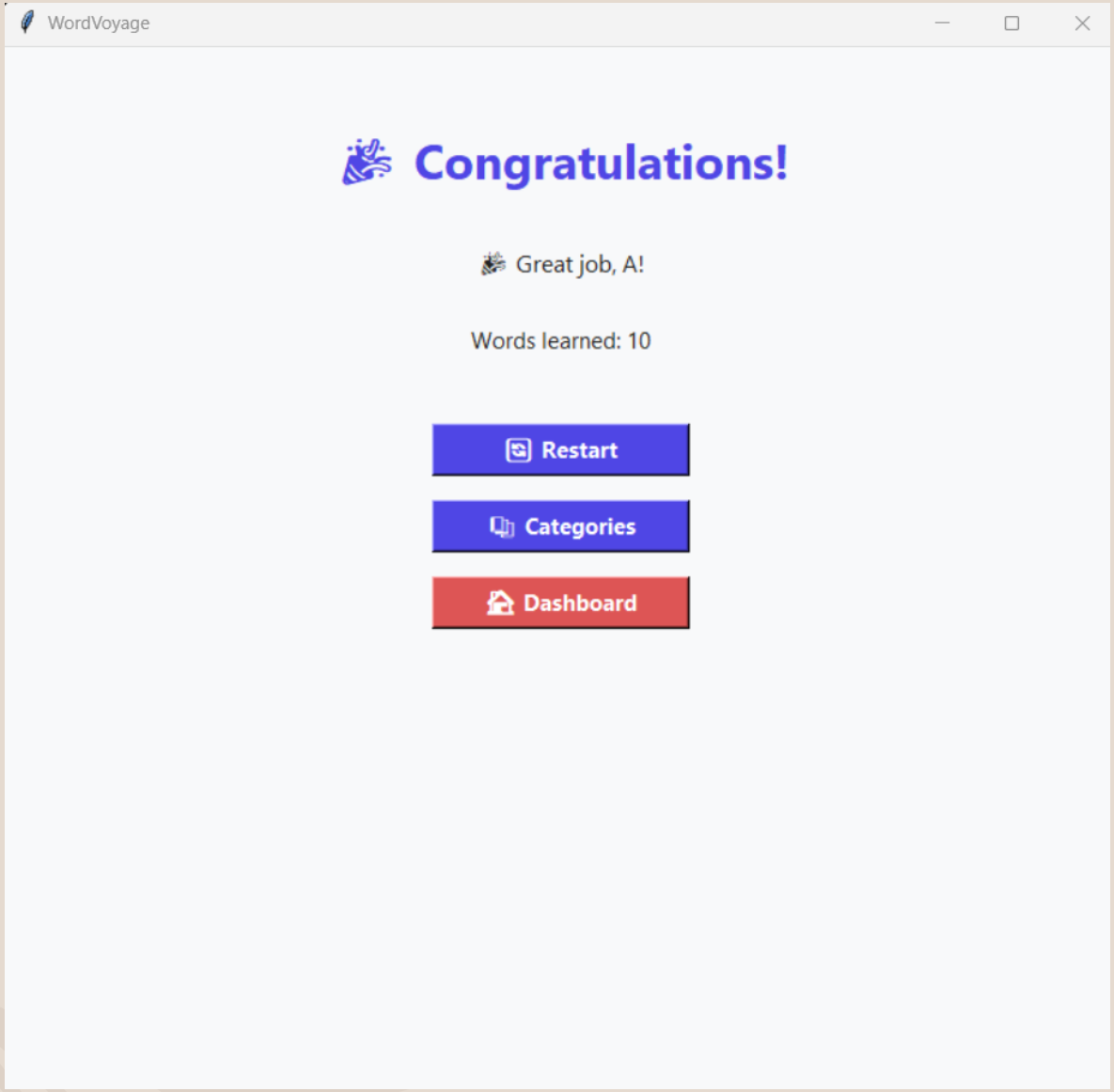
4. Below the actual script is the pronunciation.

5. The "Show Answer" is a toggle button which changes to "Hide Answer" once revealed (the word translation shows up below Quit Flashcards button.

6. The previous button is disabled when it's the first card, and the next button changes to Finish button when it's the 10th card.
7. The Quit button was added for a better user experience.

Learn/Quiz

4. Learn Window (Completion interface)

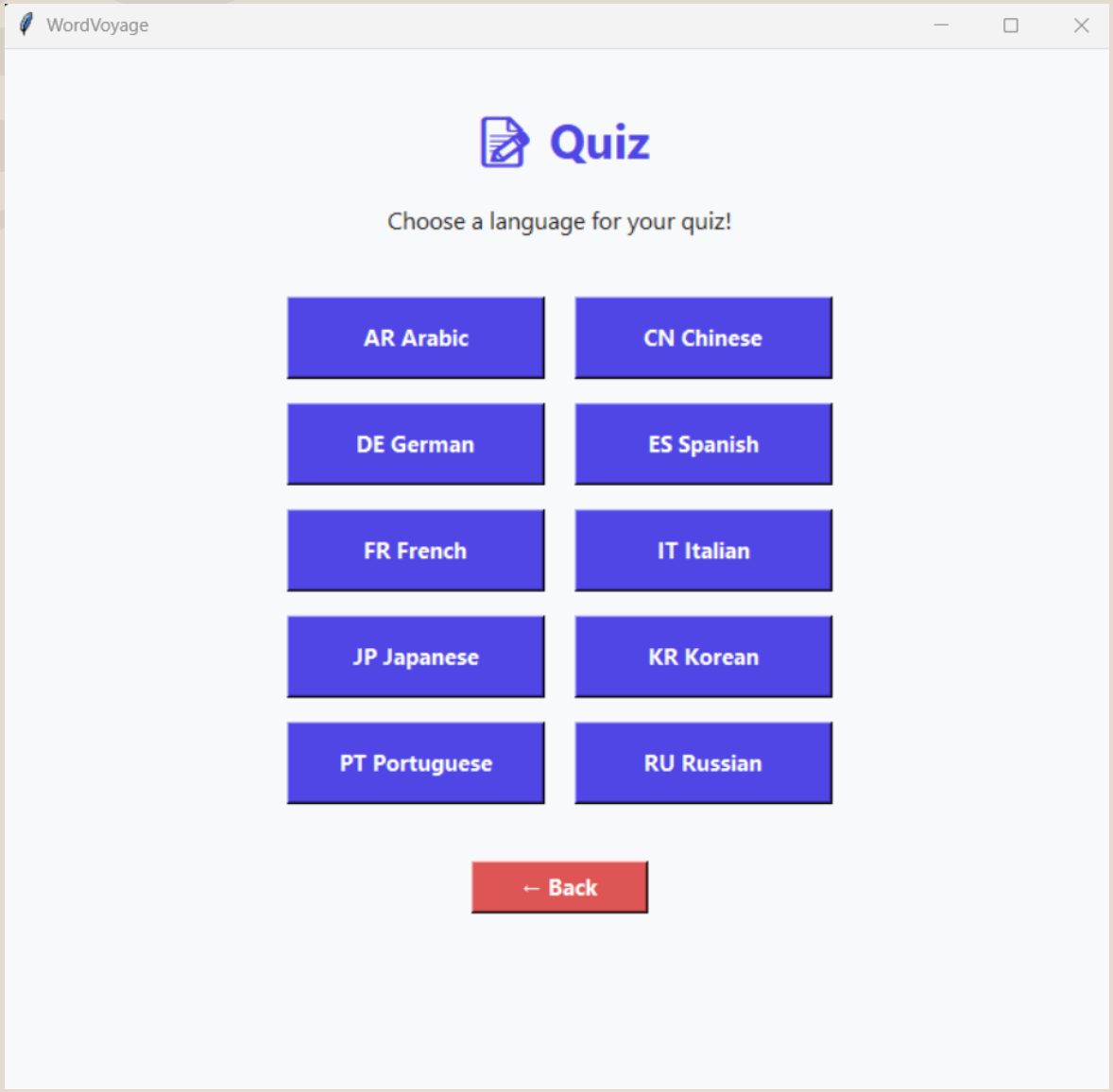


(ref. completion.py)

There's nothing to explain much here because its mostly tkinter coding with a bit of mysql query.

Learn/Quiz

1. Quiz Window (choose a language)

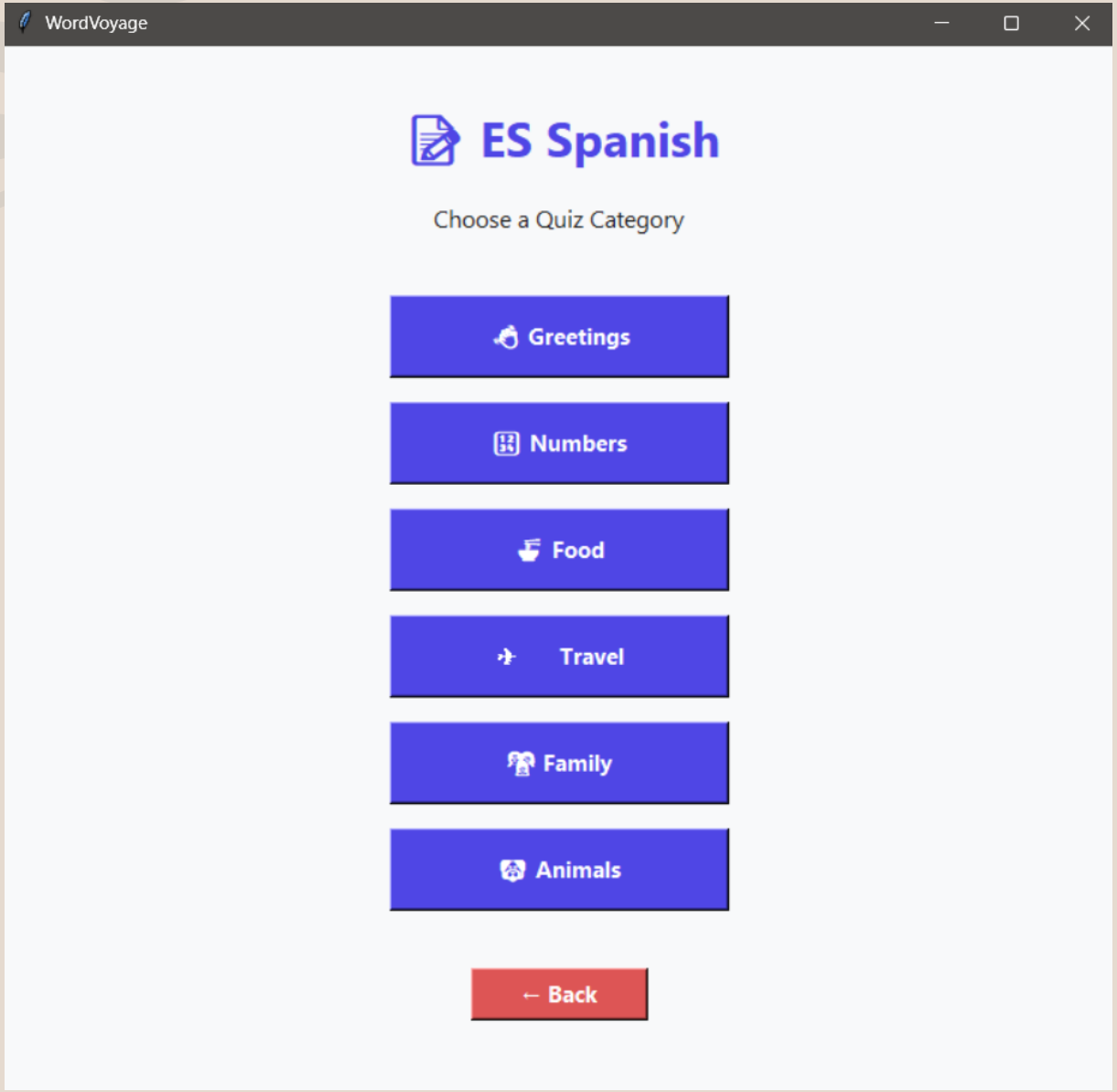


(ref quiz_languages.py)

Very similar to learn.py

Learn/Quiz

2. Quiz Window (choose a category)



(ref. quiz_categories.py)

Again similar to that of learn window's categories

Learn/Quiz

3. Quiz Window (ref. quiz.py)

NOTE: Cooldown time is 2.5 seconds





Learn/Quiz

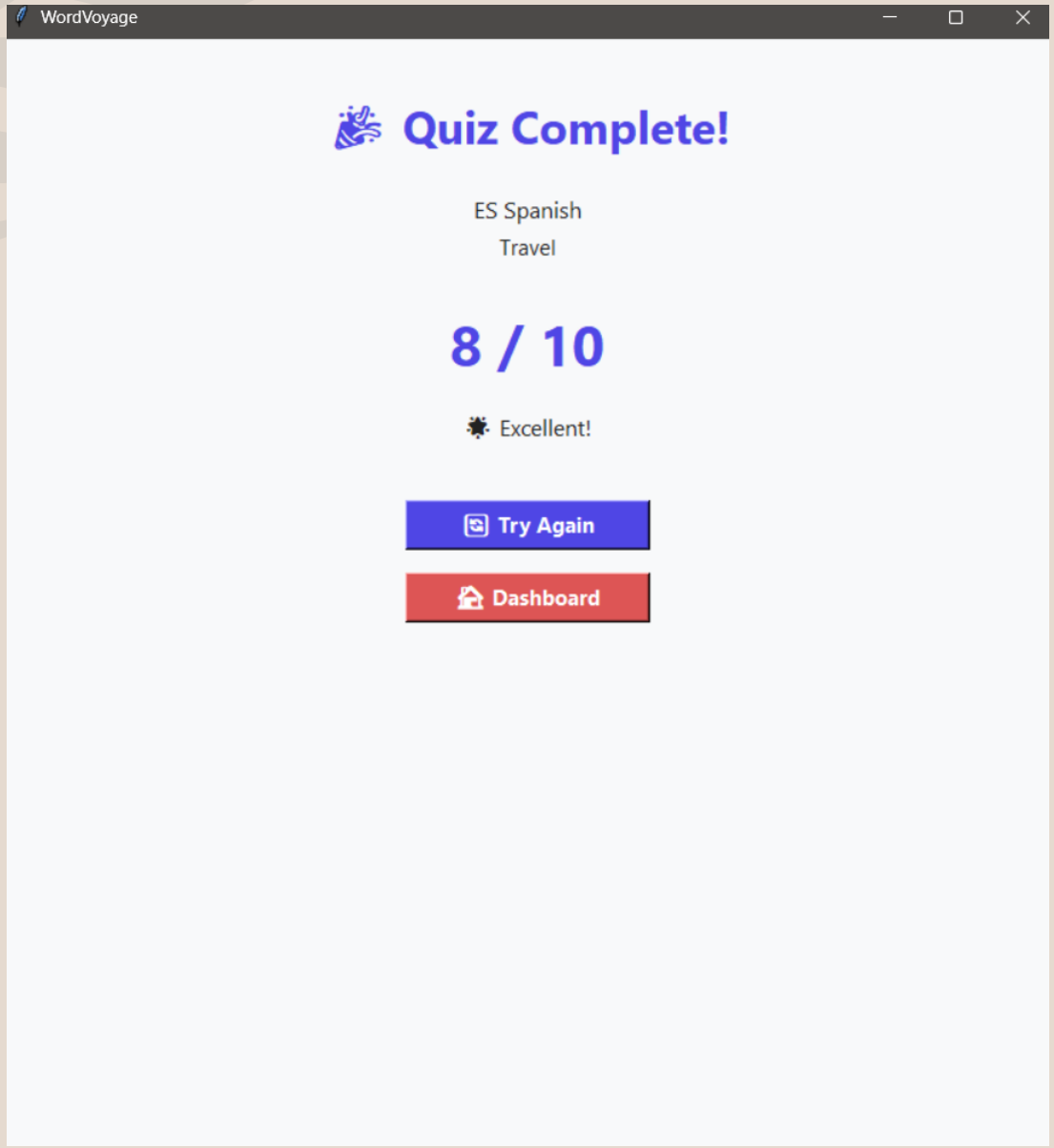
4.Quiz Window (ref quiz.py)





Learn/Quiz

5. Quiz Window (ref. quiz_completion.py)



Its pretty similar to learn window's completion page, but here, the quiz data is stored in the "quiz_results" table in the database.

This data is then used in places like progress window and also in profile window, again using queries written within the python code at various places.

Learn/Quiz (logic behind removing hard-coding

In our WordVoyage project, we reduced hardcoding by retrieving language and category names dynamically from our MySQL database using Python's mysql.connector module.

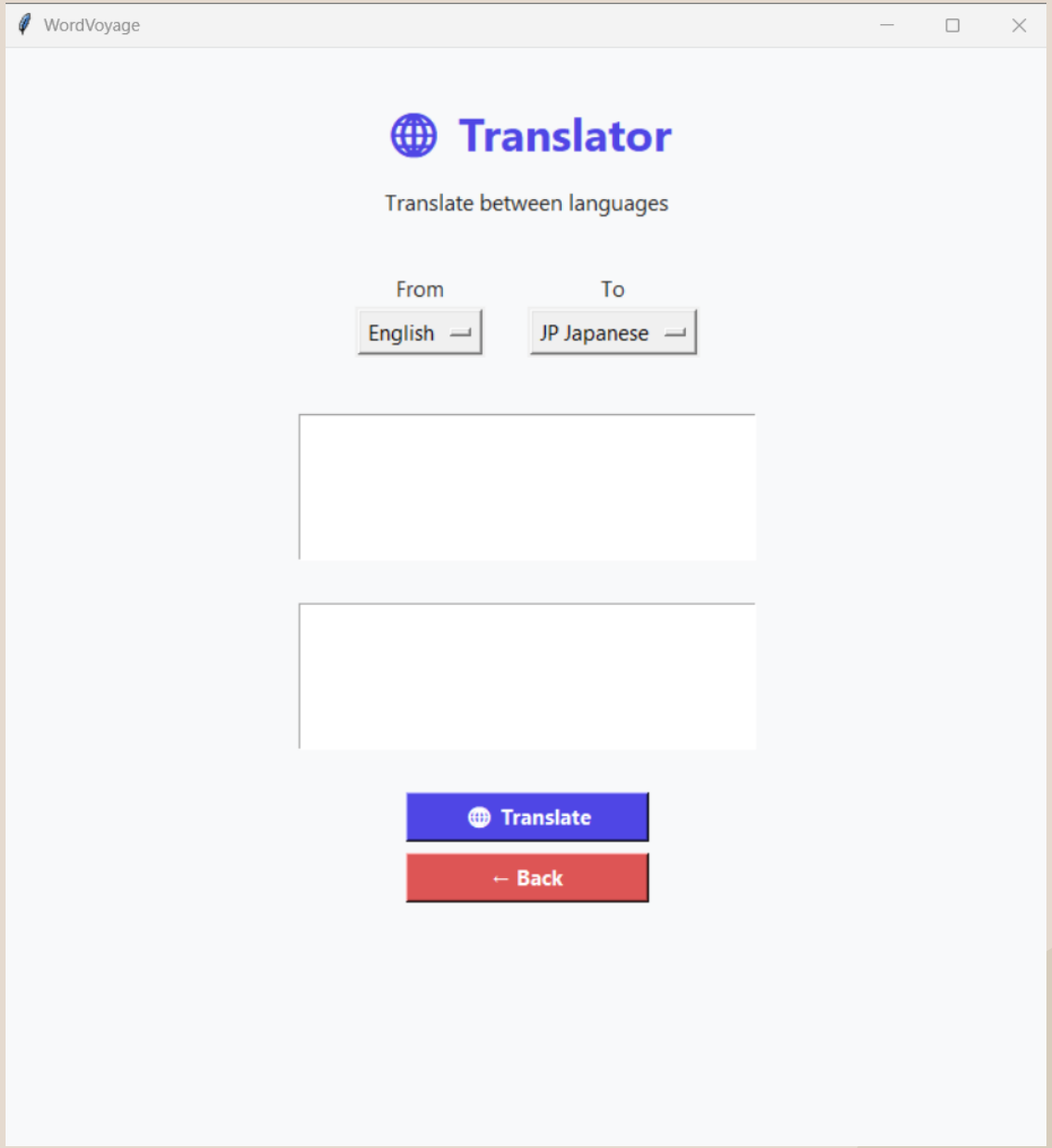
In learn.py, we use an SQL query with SELECT DISTINCT to fetch the available language names from the flashcards table. In categories.py, we retrieve category names from the categories table.

The application then generates buttons dynamically based on the retrieved data. Therefore, when we add a new language with flashcards or add a new category to the database, the corresponding buttons can appear without modifying the Python code.

However, the category emojis are still hardcoded in a Python dictionary. Thus, the language and category names are dynamically loaded, while some presentation elements remain hardcoded.

In this project, **the enumerate() function** is used in learn.py, quiz_language.py, to access both the index and language details while iterating through the list of languages retrieved from the database. The index is used to calculate the row and column positions using floor division (//), which calculates the quotient, and modulo (%), which calculates the remainder, respectively. This allows the language buttons to be arranged automatically in a two-column grid without manually specifying the position of each button.

TRANSLATE/PROGRESS



Well, this feature of the project has the most potential to be better but wrt 12th grade level, it shows another pretty solid use of python connectivity with mysql.

1. Uses the "Flashcards" table for translation.

2. It can translate from the 10 languages included to English, or the other way and even translation between two languages by using english as the bridge between the two languages.

3. For this to work, the word given in the first box has to be in the table and that too in its original script format and not in its English format (unless both English and that language have the same script ofc)

4. About using English as the bridge, all the 60 words used (10 words for each of the 6 categories) are the exact same for all the languages, the English word used is always the same for convenience.
5. It cannot translate a group of words if the don't exist exactly like that in the table, all cuz it heavily depends on the table

TRANSLATE/PROGRESS

```
mysql> select * from flashcards;
+--------------+---------------+---------------+---------------------------+---------------------------------+------------------+
| flashcard_id | language_name | category_name | word                      | pronounciation                  | english          |
+--------------+---------------+---------------+---------------------------+---------------------------------+------------------+
|            1 | JP Japanese   | Greetings     | こんにちは                | Konnichiwa                      | Hello            |
|            2 | JP Japanese   | Greetings     | おはよう                  | Ohayou                          | Good Morning     |
|            3 | JP Japanese   | Greetings     | こんばんは                | Konbanwa                        | Good Evening     |
|            4 | JP Japanese   | Greetings     | さようなら                | Sayounara                       | Goodbye          |
|            5 | JP Japanese   | Greetings     | ありがとう                | Arigatou                        | Thank You        |
|            6 | JP Japanese   | Greetings     | すみません                | Sumimasen                       | Excuse Me        |
|            7 | JP Japanese   | Greetings     | ごめんなさい              | Gomen nasai                     | Sorry            |
|            8 | JP Japanese   | Greetings     | はい                      | Hai                             | Yes              |
|            9 | JP Japanese   | Greetings     | いいえ                    | Iie                             | No               |
|           10 | JP Japanese   | Greetings     | はじめまして              | Hajimemashite                   | Nice to meet you |
|           11 | JP Japanese   | Numbers       | 一                        | Ichi                            | One              |
|           12 | JP Japanese   | Numbers       | 二                        | Ni                              | Two              |
|           13 | JP Japanese   | Numbers       | 三                        | San                             | Three            |
|           14 | JP Japanese   | Numbers       | 四                        | Yon                             | Four             |
```

TRANSLATE/PROGRESS

10	JP Japanese	Greetings	はじめまして	Hajimemashite	Nice to meet you
70	FR French	Greetings	Enchanté	Enchanté	Nice to meet you
130	DE German	Greetings	Freut mich	Freut mich	Nice to meet you
250	IT Italian	Greetings	Piacere di conoscerti	Pya-cheh-reh dee ko-no-sher-tee	Nice to meet you
310	KR Korean	Greetings	만나서 반가워요	Mannaseo bangawoyo	Nice to meet you
370	CN Chinese	Greetings	很高兴认识你	Hěn gāoxìng rènshi nǐ	Nice to meet you
430	RU Russian	Greetings	Очень приятно	Ochen priyatno	Nice to meet you

And so on….

TRANSLATE/PROGRESS

```
# -------------------------
# English → Other
# -------------------------

if from_language == "English":

    query = """
    SELECT word
    FROM flashcards
    WHERE english=%s
    AND language_name=%s
    LIMIT 1
    """

    cursor.execute(
        query,
        (
            text,
            to_language
        )
    )

    result = cursor.fetchone()

    if result:
        translation = result["word"]
```

```
# -------------------------
# Other → English
# -------------------------

elif to_language == "English":

    query = """
    SELECT english
    FROM flashcards
    WHERE word=%s
    AND language_name=%s
    LIMIT 1
    """

    cursor.execute(
        query,
        (
            text,
            from_language
        )
    )

    result = cursor.fetchone()

    if result:
        translation = result["english"]
```

```
# Other → Other
# -------------------------

else:

    query = """
    SELECT english
    FROM flashcards
    WHERE word=%s
    AND language_name=%s
    LIMIT 1
    """

    cursor.execute(
        query,
        (
            text,
            from_language
        )
    )

    english_result = cursor.fetchone()

    if english_result:

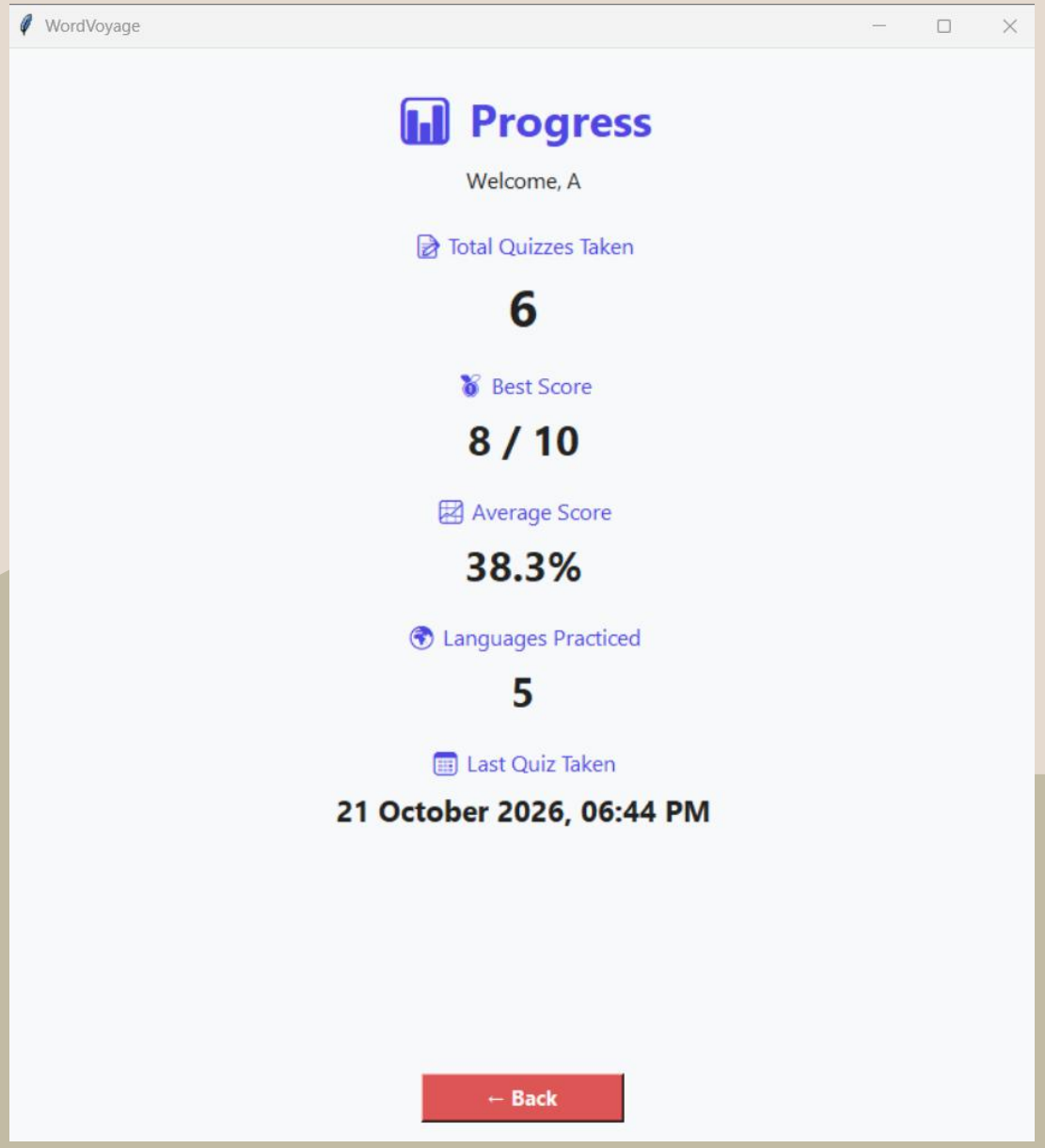
        english_word = english_result["english"]

        query = """
        SELECT word
        FROM flashcards
        WHERE english=%s
        AND language_name=%s
        LIMIT 1
        """

        cursor.execute(
            query,
            (
                english_word,
                to_language
            )
        )

        final_result = cursor.fetchone()
```

TRANSLATE/PROGRESS



Well, the progress part highly uses tkinter coding for structure, taking info. from quiz_results table, a bit of math module, and also curdate to keep a track of both date and time of last quiz.

DATABASE

```
+---------------------+
| Tables_in_polylingo |
+---------------------+
| categories          |
| flashcards          |
| languages           |
| quiz_results        |
| users               |
+---------------------+
```

Categories, languages tables were used before hardcoding was removed for languages and categories, we can remove them but its better to show it instead.

```
mysql> select * from users;
+---------+-----------+---------------+-------------+
| user_id | username  | password      | date_joined |
+---------+-----------+---------------+-------------+
|       1 | admin     | admin123      | 2026-07-30  |
|       2 | alex      | 1234          | 2026-07-30  |
|       3 | WittyBFool | 321899       | 2026-07-31  |
|       4 | A         | A             | 2026-08-04  |
|       5 | test      | 123           | 2026-08-11  |
|       6 | S         | S             | 2026-08-11  |
|       7 | D         | 1             | 2026-10-02  |
|       8 | AD        | 12345678Aa@   | 2026-10-03  |
|       9 | An        | 1TheElrics819$ | 2026-10-17 |
+---------+-----------+---------------+-------------+
9 rows in set (0.00 sec)
```

I was thinking of removing many of the records here, its too messy

DATABASE

```
mysql> select
    -> * from quiz_results;
+-----------+------------+---------------+---------------+-------+-----------------+---------------------+
| result_id | username   | language_name | category_name | score | total_questions | date_taken          |
+-----------+------------+---------------+---------------+-------+-----------------+---------------------+
|         1 | WittyBFool | JP Japanese   | Greetings     |     6 |              10 | 2026-08-06 20:04:53 |
|         2 | A          | JP Japanese   | Numbers       |     8 |              10 | 2026-08-06 20:13:13 |
|         3 | WittyBFool | JP Japanese   | Numbers       |     8 |              10 | 2026-08-15 16:22:46 |
|         4 | WittyBFool | JP Japanese   | Numbers       |     0 |              10 | 2026-08-15 16:23:07 |
|         5 | WittyBFool | FR French     | Food          |     7 |              10 | 2026-08-15 16:24:10 |
|         6 | A          | JP Japanese   | Greetings     |     0 |              10 | 2026-08-17 17:35:11 |
|         7 | A          | IT Italian    | Numbers       |     7 |              10 | 2026-08-19 13:24:59 |
|         8 | A          | KR Korean     | Greetings     |     0 |              10 | 2026-08-19 13:36:25 |
|         9 | A          | AR Arabic     | Greetings     |     0 |              10 | 2026-08-19 19:04:14 |
|        10 | D          | JP Japanese   | Numbers       |     5 |              10 | 2026-10-02 12:56:17 |
|        11 | D          | FR French     | Animals       |     6 |              10 | 2026-10-02 13:24:11 |
|        12 | A          | ES Spanish    | Travel        |     8 |              10 | 2026-10-21 18:44:10 |
+-----------+------------+---------------+---------------+-------+-----------------+---------------------+
12 rows in set (0.00 sec)
```

SOME AUXILIARY FILES:

1. *main.py:* we use this file to run the program, connects to app.py

2. *app.py:* this connects the whole project to main.py, has all the def functions for each of the windows, its like the bridge between the execution and source of the project.

3. *db_connection.py:* just the mysql connector import statement, and the connection itself (user, database, passwd, etc)

4. *theme.py:* maintains homogeneity in colour code (so that writing the same hex code for frequently used colours would be reduced.)

5. *splash.py:* code related the initial splash screen (inspired from gaming applications)