Group 5
Chaeri Son , Daechan Yang, Seungwoo Cho

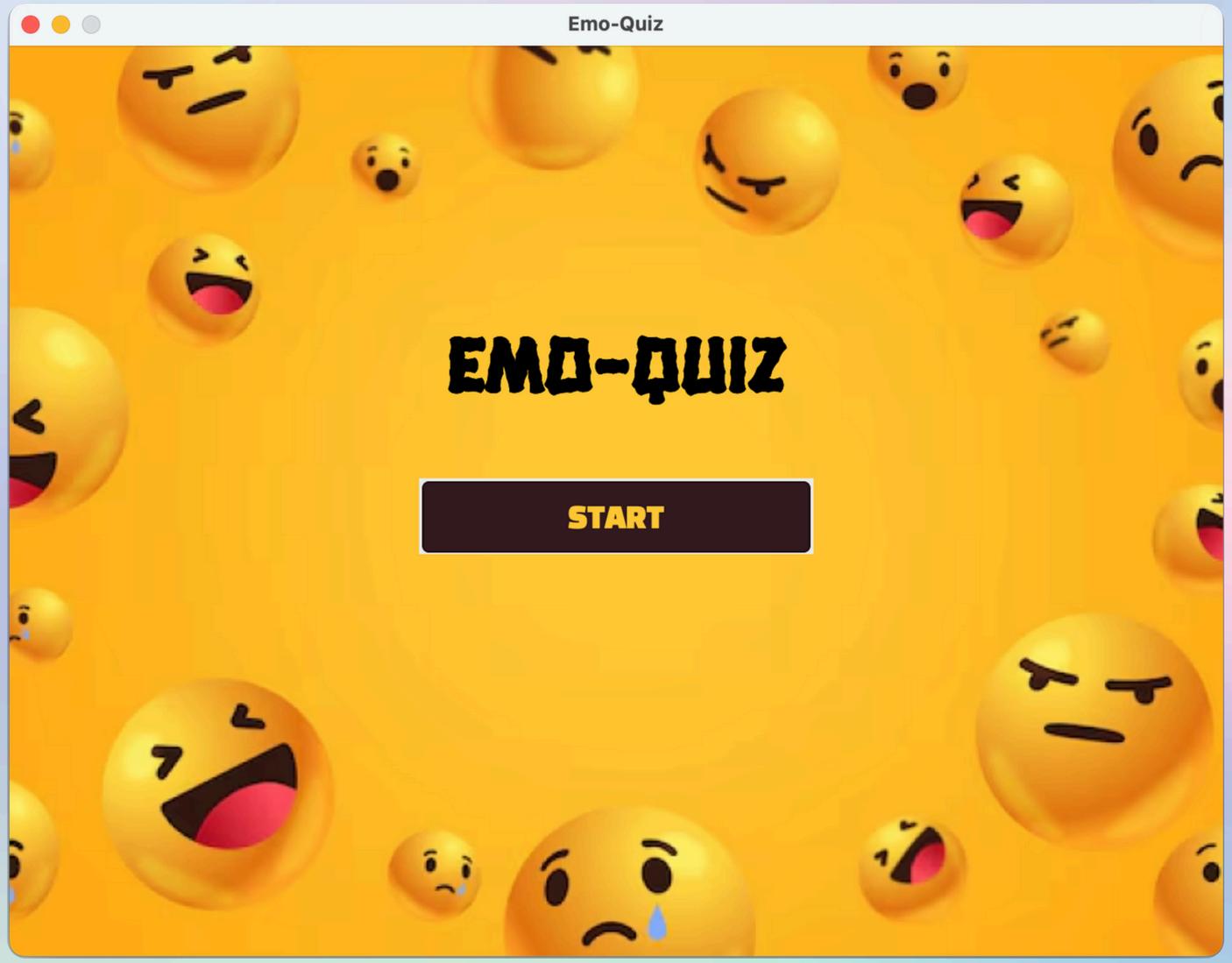
Fall 2024 RDBMS

Update) Emo-Quiz

Duolingo?
Noooooooo
✧Emoji Quiz✧
ง(• ᴗ •)

Index

Introduction



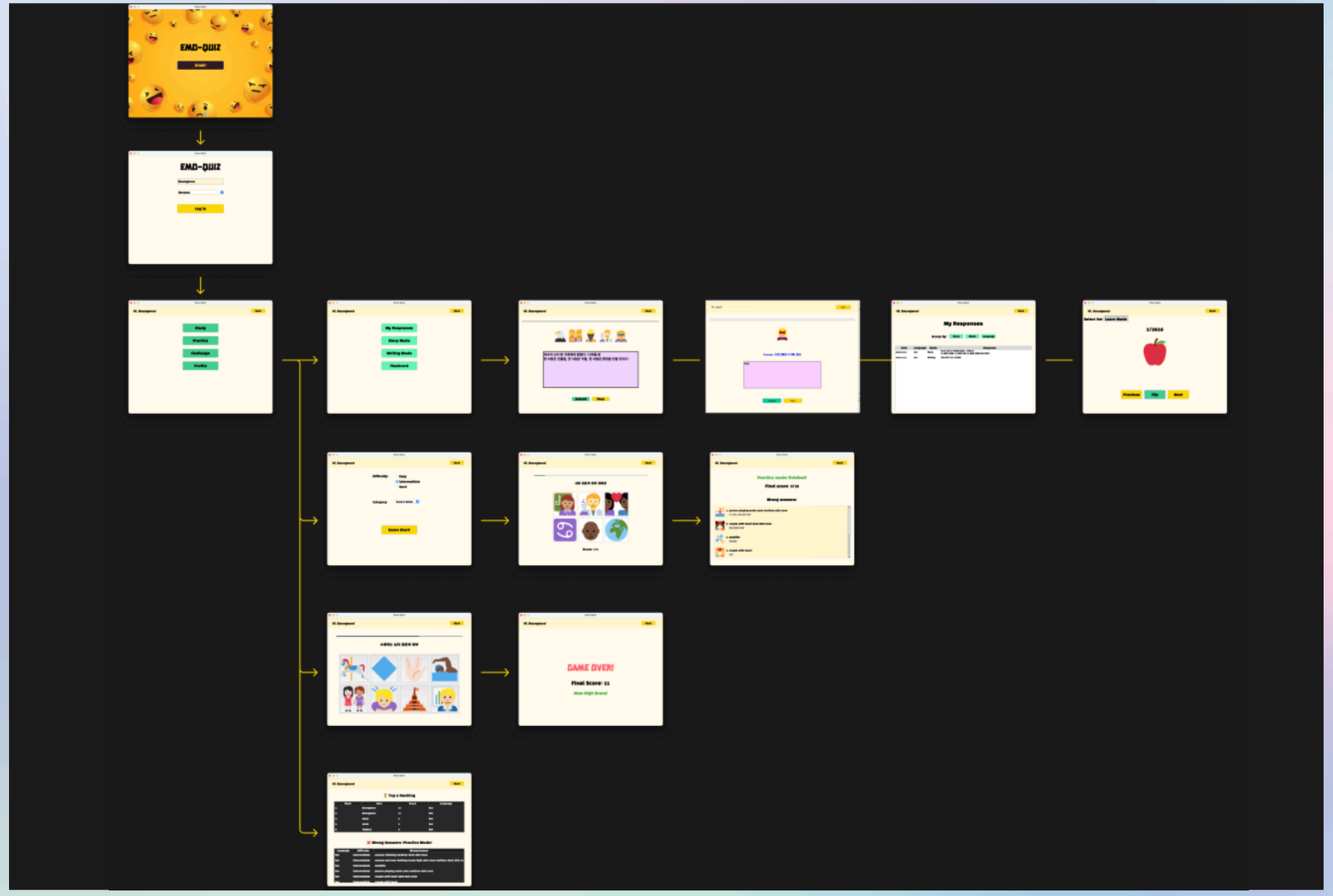
Target Users of Application

Everyone who needs an easy way
to learn foreign language

Goal

To provide a visualized, effective and fun
language learning system to global users

Update - Introduction



Main Functions

- Study mode
- Practice mode
- Challenge mode
- Profile
- Link : (https://www.figma.com/design/jui3ralrcUG8JXH8WwNWg5/Relational-DB-Team-Project?node-id=231-228&t=8YnO35esDOiZpSjX-1)

Update - Introduction

```
import tkinter as tk
from tkmacosx import *

import random
from tkinter import ttk, messagebox
from PIL import ImageTk, Image

import mysql.connector

import os
import json
from datetime import datetime
```

index int64 0 5.03k	emoji string · lengths 1 10	category string · classes 10 values	description_eng_Latn string · lengths 2 76	description_deu_Latn string · lengths 2 88	description_spa_Latn string · lengths 2 97
0	🍎	food	red apple	roter apfel	manzana roja
1	🍐	food	pear	birne	pera
2	🍊	food	tangerine	mandarine	mandarina
3	🍋	food	lemon	zitrone	limón
4	🍌	food	banana	banane	plátano
5	🍉	food	watermelon	wassermelone	sandía
6	🍇	food	grapes	trauben	uvas
7	🍓	food	strawberry	erdbeere	fresa
8	🫐	food	blueberries	blaubeeren	arándanos

Language / Packages

- **GUI Develop**: Tkinter, PIL
- **Functions/OS**: MySQL, tkmacosx, datetime, os
- **Image/Game**: json, random

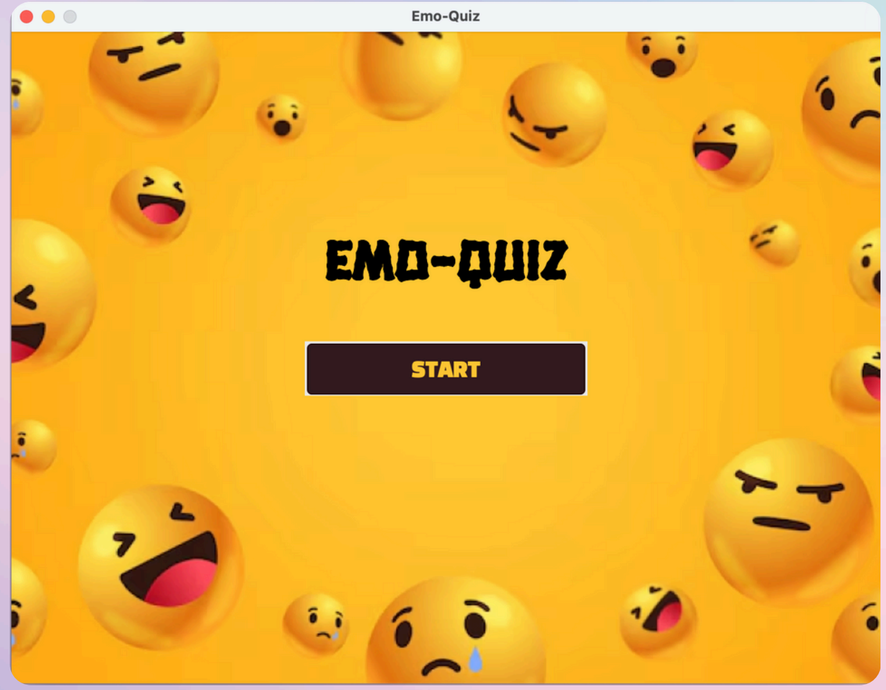
Dataset

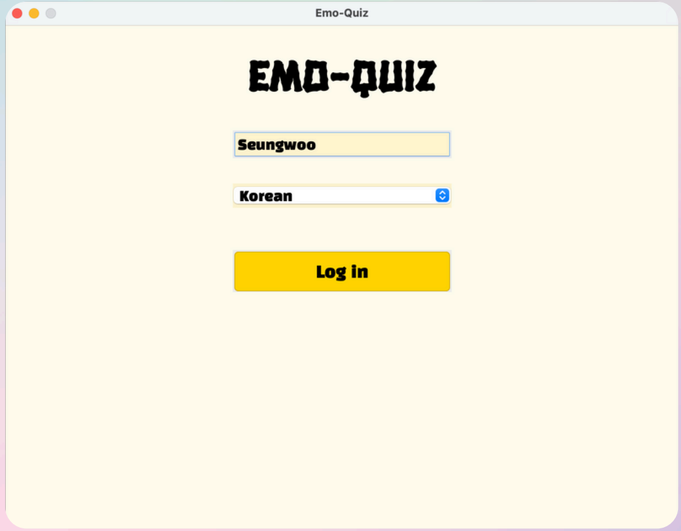
- Emoji dataset from Huggingface
- (Link: Kamali, Omar. Emoji Map Dataset. Hugging Face,https://huggingface.co/datasets/omarkamali/emoji-map/viewer. Accessed 10 Dec. 2024)
- Had to organize tables for Emo-Quiz

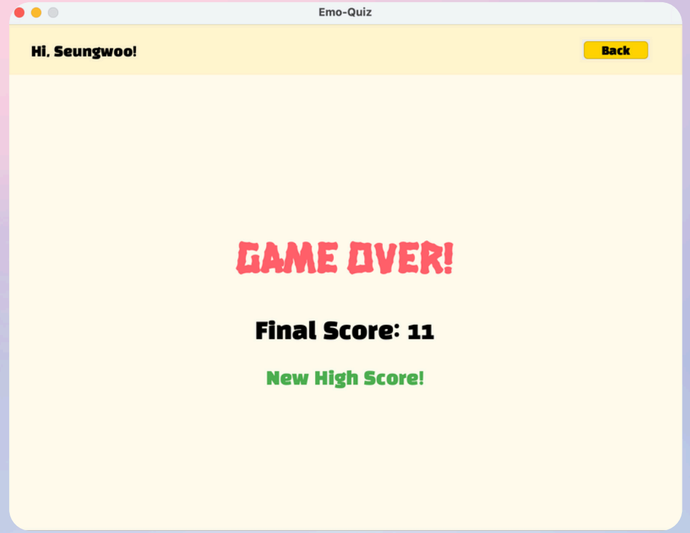
GUI Design - Mood & Concept

It's a Game!

Used game-friendly fonts, colors, images to create game mood







GUI Design – Controls

Buttons

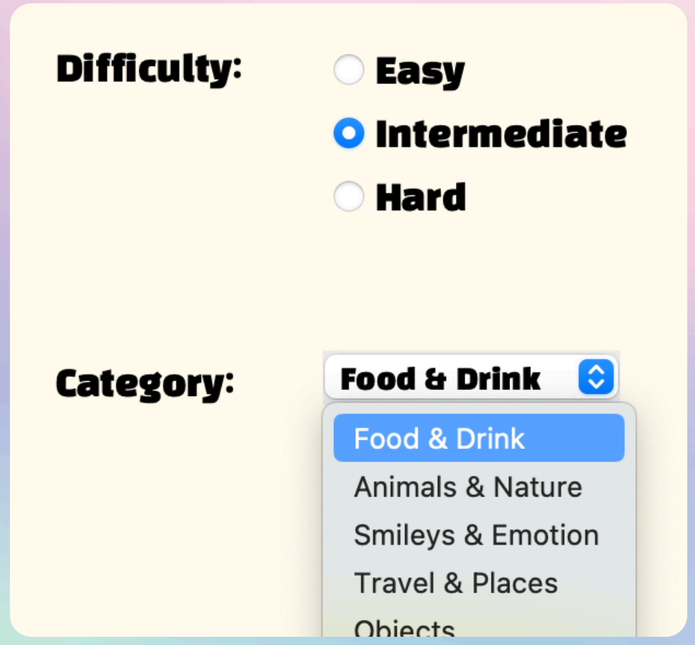
Buttons have different colors based on their functions and hierarchy

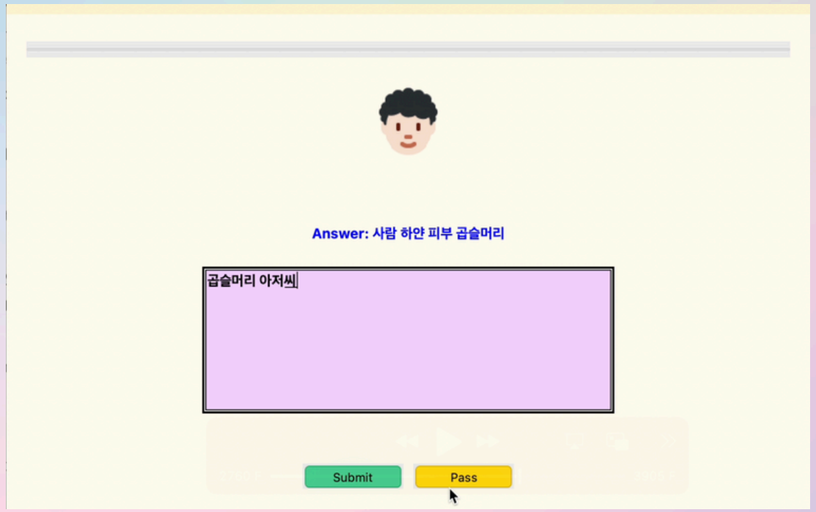
Radio btn & Combo box

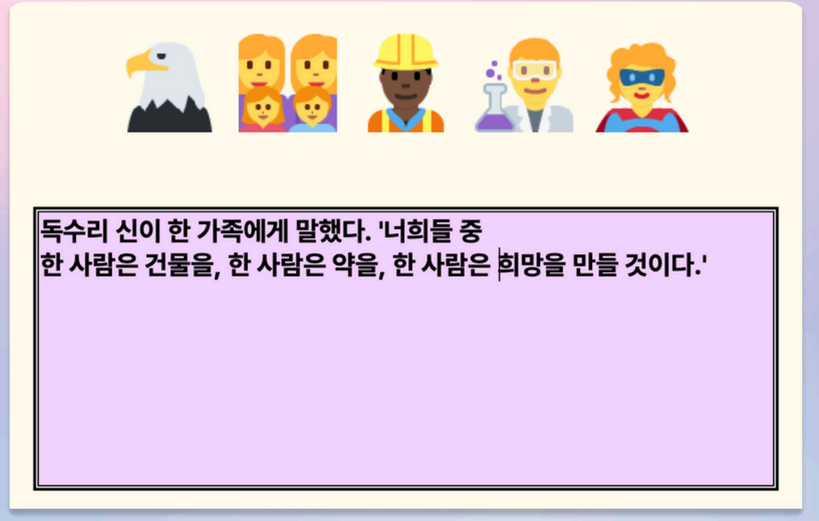
User can easily control their options with radio buttons and combo box

Textfields

Textfields are clearly distinguished from background,
by using its complementary colors







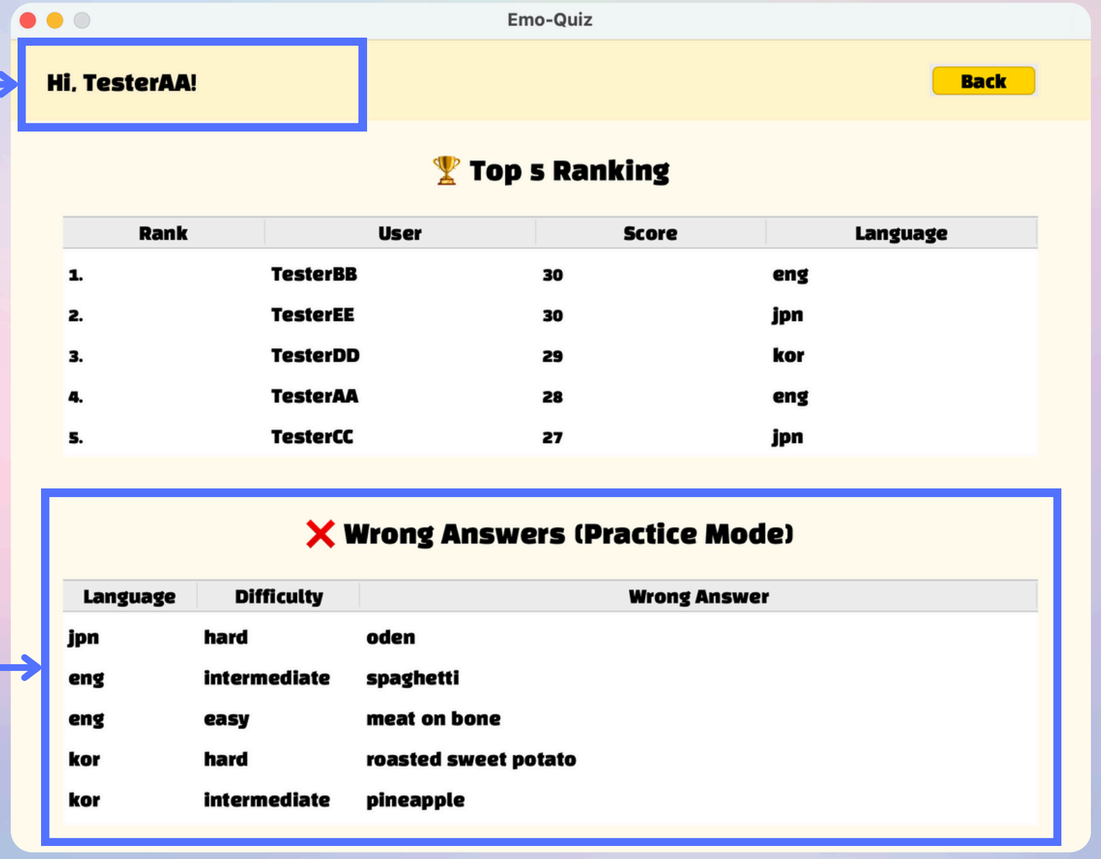
GUI Design - User Interaction

Real User Experience

- User can create their own ID and play the game with it
- GNB and welcome message are always displayed at the top of the screen
- User can confirm their wrong answers from practice mode

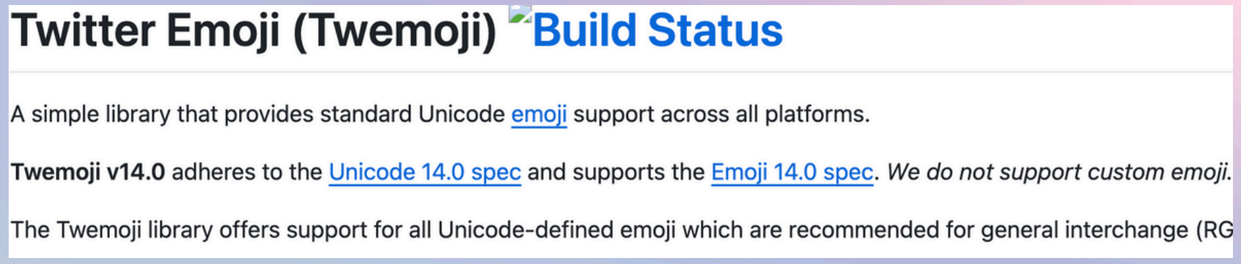
Log In view

Profile view



Update - Data cleaning

index int64	emoji string · lengths	category string · classes	description_eng_Latn string · lengths	description_deu_Latı string · lengths
0 – 5.03k	1 – 10	10 values	2 – 76	2 – 88
0	🍎	food	red apple	roter apfel
1	🍐	food	pear	birne
2	🍊	food	tangerine	mandarine
3	🍋	food	lemon	zitrone
4	🍌	food	banana	banane
5	🍉	food	watermelon	wassermelone
6	🍇	food	grapes	trauben
7	🍓	food	strawberry	erdbeere



Twitter Emoji (Twemoji) Build Status

A simple library that provides standard Unicode emoji support across all platforms.

Twemoji v14.0 adheres to the Unicode 14.0 spec and supports the Emoji 14.0 spec. *We do not support custom emoji.*

The Twemoji library offers support for all Unicode-defined emoji which are recommended for general interchange (RG

```
# Seperate the table for our task
emoji_basic = df[['index', 'emoji', 'group', 'codepoints']]
description_cols = ['index'] + [col for col in df.columns if col.startswith
('description_')]
emoji_description = df[description_cols]
```

Emoji Map Dataset:
Kamali, Omar. Emoji Map Dataset. Hugging Face, https://huggingface.co/datasets/omarkamali/emoji-map/viewer. Accessed 10 Dec. 2024

Twemoji:
Twitter. Twemoji. GitHub, https://github.com/twitter/twemoji. Accessed 10 Dec. 2024.

1) **What was our initial dataset?**

- First Dataset: 5027 Emojis with 18 languages description

(Link: Kamali, Omar. Emoji Map Dataset. Hugging Face, https://huggingface.co/datasets/omarkamali/emoji-map/viewer. Accessed 10 Dec. 2024)

- Second Dataset: Emojis with Unicode

(Link: TheDevastator. Analyzing Emoji Characteristics Through Unicode. Kaggle, 2023, https://www.kaggle.com/datasets/thedevastator/analyzing-emoji-characteristics-through-unicode. Accessed 13 Dec. 2024)

- => Merged Dataset : The First Dataset + Second Dataset

2) **Extract the Emoji Images from Twemoji**

- P : Tkinter can't render emojis well, need alternative
- S : Extract the Twemoji images with the Unicode
- Drop the emoji that can't align with the Twemoji Images.

3) **Divide the Merged dataset into Two Datasets**

- emoji_basic –> Real Emoji file!
- emoji_description –> Emoji descriptions with 15 languages

+More information is in our data cleaning file!

Update - Special algorithms

```
:lass practice_mode:

    def generate_questions(self):
        self.cursor.execute("DELETE FROM practice_log")
        self.db.commit()

        for _ in range(self.total_rounds):
            category_condition = "" if self.category == 'All categories' else f"AND eb.category = '{self.category}'"

            # emoji choice - correct one
            query_correct = f"""
            SELECT eb.idx, ed.description_{self.language} as description
            FROM rdbmsfinal.emoji_basic eb
            JOIN rdbmsfinal.emoji_description ed ON eb.idx = ed.idx
            WHERE ed.description_{self.language} IS NOT NULL
            {category_condition}
            ORDER BY RAND()
            LIMIT 1
            """
            self.cursor.execute(query_correct)
            correct = self.cursor.fetchone()

            # emoji choice - wrong ones
            wrong_count = self.current_options - 1
            query_wrong = f"""
            SELECT eb.idx
            FROM rdbmsfinal.emoji_basic eb
            JOIN rdbmsfinal.emoji_description ed ON eb.idx = ed.idx
            WHERE eb.idx != {correct['idx']}
            AND ed.description_{self.language} IS NOT NULL
            {category_condition}
            ORDER BY RAND()
            LIMIT {wrong_count}
            """
            self.cursor.execute(query_wrong)
            wrong = self.cursor.fetchall()
            wrong_ids = [str(w['idx']) for w in wrong]

            # practice_log
            insert_query = """
            INSERT INTO practice_log
            (correct_idx, wrong_idx, correct, correct_description, difficulty)
            VALUES (%s, %s, FALSE, %s, %s)
            """
            self.cursor.execute(insert_query,
                (correct['idx'], json.dumps(wrong_ids), correct['description'], self.difficulty))
            self.db.commit()
```

Had to think about game algorithms especially.
In practice mode, there was chance of delay in time during the game if the question was made one by one.
To prevent this, all the questions were made once, and uploaded to ‘practice_log’ table. And printed these questions one by one.

We picked one right answer, and made to choose other wrong answers except the one picked for answer.

And when the practice mode starts, it is told to clear the table, because there could be the case when the game is quit before finishing all the questions, there will be log of previous game, not being cleared.

ER Diagram

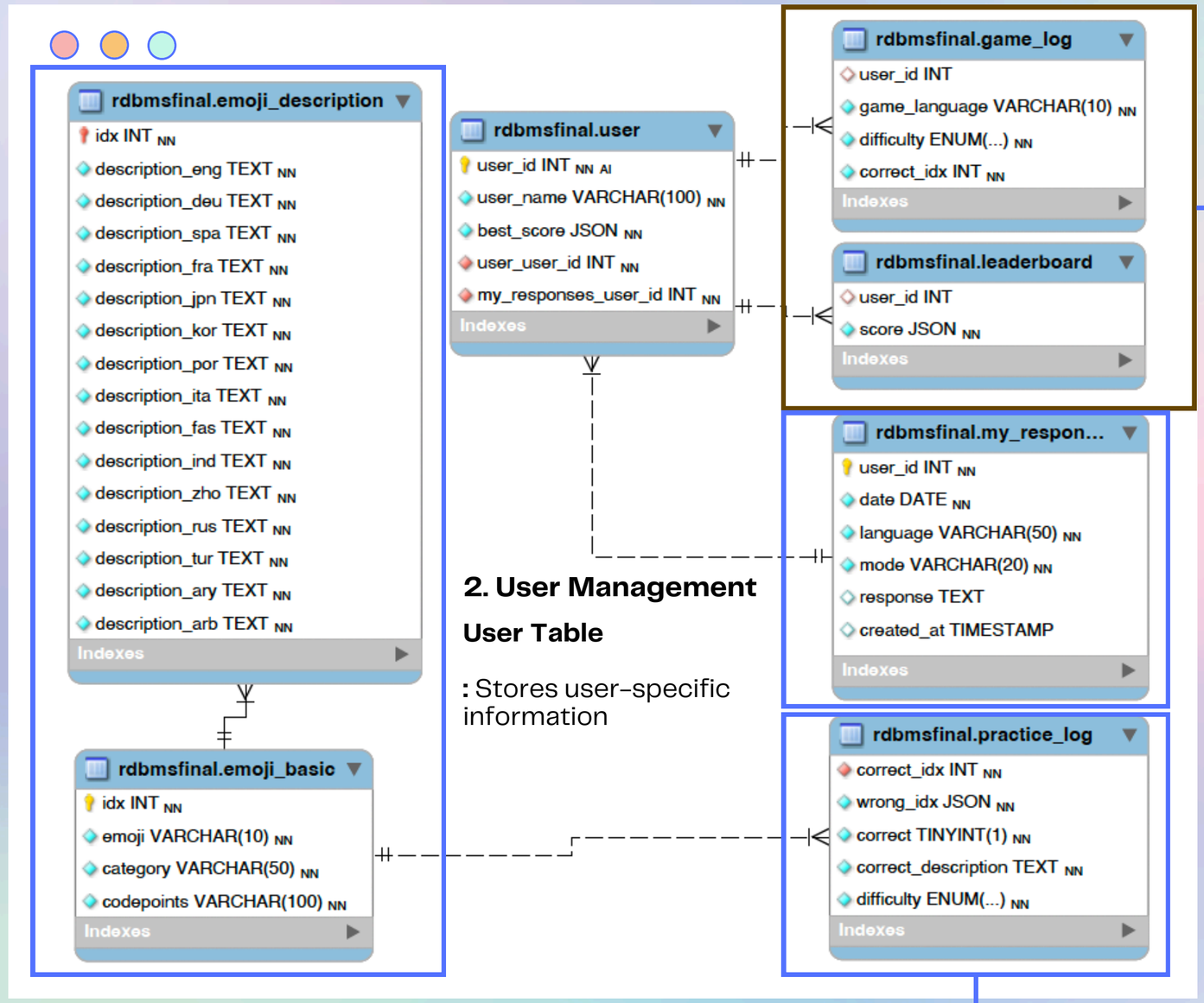
1.Core Table

1-1
Emoji_description Table

:No emoji! Only contains the description of emoji in 15 languages

1-2
Emoji_basic Table

:Has real emojis, categories, and unicode!



2. User Management

User Table

: Stores user-specific information

3. User Interaction

Game_ log Table(Wrong Answer Log)

: Table that collects all wrong questions from practice_log

Leaderboard Table

Shown in the Profile – Leaderboard
: Store the best score from the survival mode by user id.

4. Response Tracking

My_response Table

Shown in Study Mode – My Response
:Store all the responses. User can filter them by language, users, or time

5. Quiz Tracking

Practice_log Table(Quiz Sheet)

Shown in Practice Mode
: In Practice mode, 10 questions are generated and displayed on the game, this table store the questions to display on it

DEMO

Let's Start DEMO!

SQL Statement

Writing Mode

:Select 5 Random emojis

Practice Mode

:Bring problem from practice_log

Survival Mode

: Get several distractors (Wrong Options for the question)

SELECT

```
SELECT idx FROM emoji_basic
ORDER BY RAND()
LIMIT 5
```

```
SELECT *
FROM practice_log
LIMIT {self.current_round-1}, 1
```

GROUP BY

```
SELECT
    user_id,
    DATE_FORMAT(date, '%Y.%m.%d') as date,
    language,
    mode,
    GROUP_CONCAT(response SEPARATOR ', ') as response
FROM my_responses
WHERE mode != 'survival'
GROUP BY mode, user_id, date, language
ORDER BY date DESC, mode ASC
```

JOIN

```
SELECT eb.idx
FROM rdbmsfinal.emoji_basic eb
JOIN rdbmsfinal.emoji_description ed ON eb.idx = ed.idx
WHERE eb.idx != {}
AND ed.description_{} IS NOT NULL
ORDER BY RAND()
LIMIT 7
```

UPDATE/INSERT/ DELETE

```
DELETE FROM game_log
WHERE user_id = %s
AND correct_idx IN (SELECT correct_idx FROM practice_log)
```

```
INSERT INTO practice_log
(correct_idx, wrong_idx, correct, correct_description, difficulty)
VALUES (%s, %s, FALSE, %s, %s)
```

```
UPDATE practice_log
SET correct = TRUE
WHERE correct_idx = %s
```

My response

: Collect responses from writing mode and story mode, filtering them by mode and language.

Practice Mode

DELETE : Delete the overlapped question that has been in that round.

INSERT INTO: Insert the new questions into the practice_log so that the questions that hasn't been in the previous round could be pop up.

UPDATE: Update the practice_log where

Challenges

Emoji Rendering

Mac Versus Windows

Database Connection

Python Version

GUI Integrating

Update - challenge - version problems

Mac Versus Windows

- tkmacosx package
- size difference
- font system

Python Version

- difference in virtual environment
- python version (v3.12.1-->v3.13.0)

Database Connection

- Setting directions of files set to each person
- Should have merged these up in the code

GUI Integrating

- Had to think of some additional settings (ex. frame, grid, size including menu system...)

Update - challenge - emoji rendering

832	👩‍❤️‍💋‍👩	1F468 200D 2695 FE0F
833	👨‍❤️‍💋‍👨	1F469 200D 2695 FE0F
834	👩‍❤️‍👩	1F468 200D 1F393
835	👨‍❤️‍👨	1F469 200D 1F393
836	👩‍❤️‍💋‍👨	1F468 200D 1F3EB
837	👨‍❤️‍💋‍👩	1F469 200D 1F3EB

emojis with complex unicode

```
# Process only complex emojis
complex_emojis = df[df['codepoints'].apply(lambda x: is_complex_emoji(str(x)))]
total = len(complex_emojis)
print(f"Processing {total} complex emojis.")

for i, row in complex_emojis.iterrows():
    index = row['index']
    code = format_code(str(row['codepoints']))

    url = f"https://raw.githubusercontent.com/twitter/twemoji/master/assets/72x72/{code}.png"
    save_path = os.path.join(save_dir, f"twemoji_{index}.png")

    # Print progress every 50 items
    if len(failed_indices) % 50 == 0 and len(failed_indices) > 0:
        print(f"Processing... Failed count so far: {len(failed_indices)}")

    try:
        response = requests.get(url)
        if response.status_code == 200:
            with open(save_path, 'wb') as f:
                f.write(response.content)
        else:
            failed_indices.append(index)

    except Exception as e:
        failed_indices.append(index)

# Save failed indices to txt file
if failed_indices:
    with open(failed_txt, 'w') as f:
        f.write('\n'.join(map(str, failed_indices)))
    print(f"\nFailed to download {len(failed_indices)} complex emojis.")
    print(f"Failed indices have been saved to {failed_txt}")
```

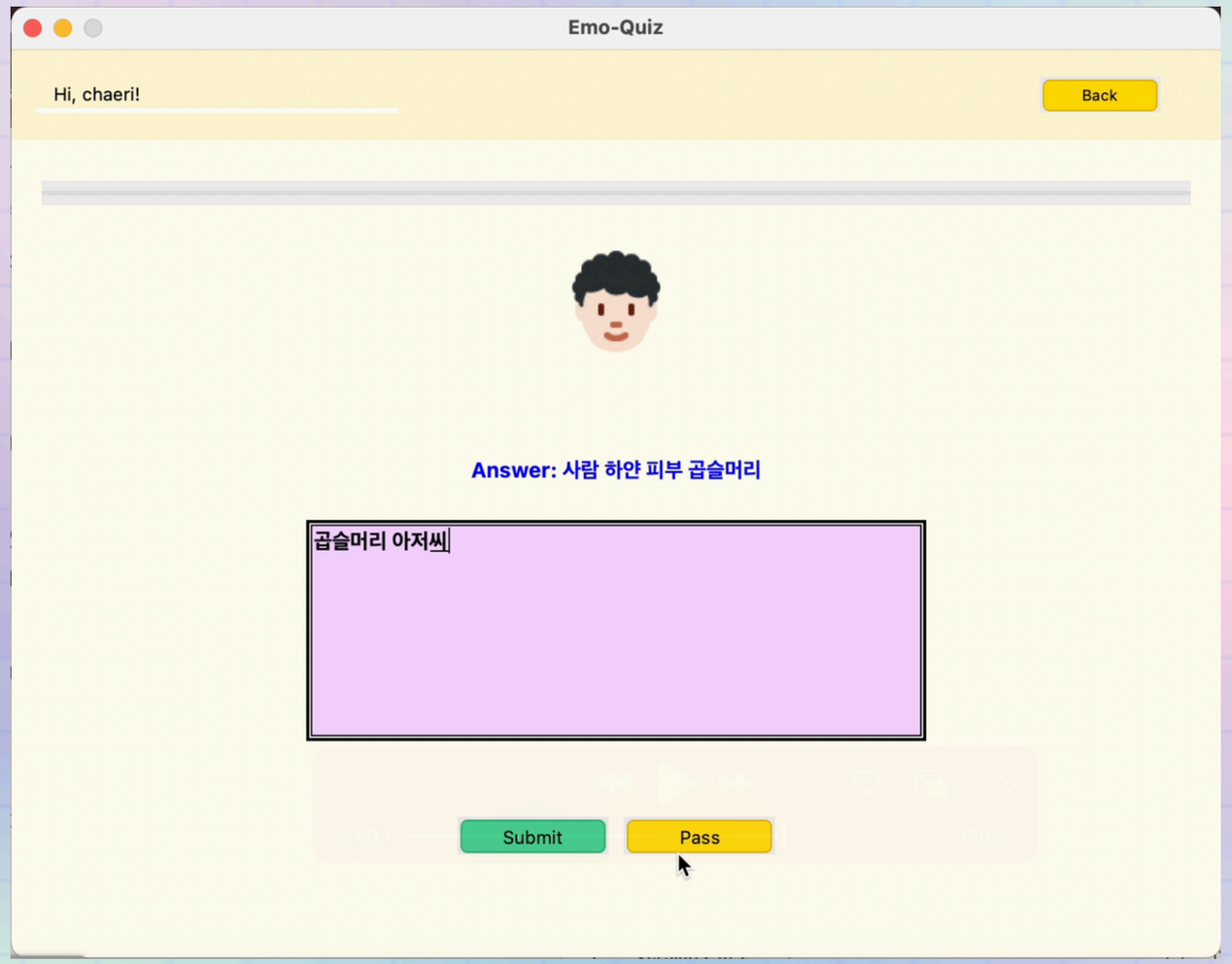
twemoji downloading (by using unicode)

writing text file of emoji indexes that have failed to download

Deciding to use twemoji, we had to download it.
Here's specific step

1. merge previous table with just emojis and descriptions) with a table that shows each emoji's unicode.
2. Each twemojis were named by unicodes, so had to download using unicode.
3. Lots of emojis were not downloaded, because there was new emojis with complex multiple unicode.
4. Download emojis that has complex unicode
5. Check the emojis that were not on twemoji image set, drop them from the table because we cannot use that for our game.

Update Writing Mode



Update) Timeline

Oct 25th
Dataset Decision & Topic Ideation

Finalize dataset selection and ideate the project topic

Nov 2nd
Game Logic, Framework decided

Decide on the logic and framework for implementation

Nov 10th
Twemoji Data Download & Cleaning

Download Twemoji dataset and clean the existed dataset

Nov 16th
Complete the Data Structure

Completed the database schema design in SQL workbench

Nov 24th
Study Mode Completed

Finalize development of the study mode

Dec 1st
Practice Mode Completed

Finalize development of the practice mode

Dec 8th
Practice Mode Completed

Finalize development of the survival mode

Dec 10th
GUI Merging Completed

Complete GUI integration and merging

Group 5
Chaeri Son , Daechan Yang, Seungwoo Cho

Thank you for Listening!