CSC 496 Fall 2023
Lab 2: Pokédex version 3

Dr. Si Chen

October 12, 2023

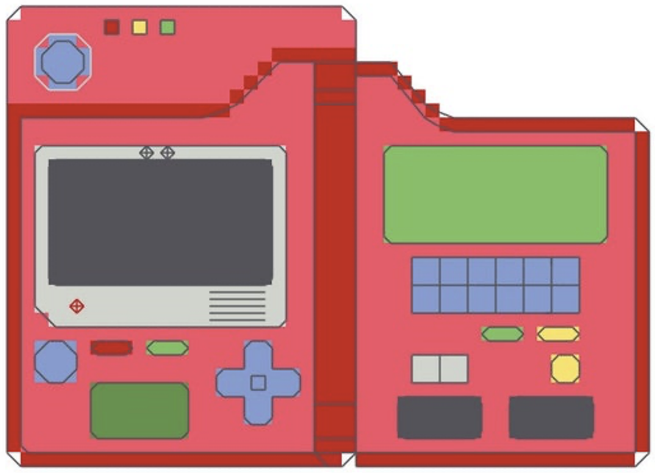

Figure 1: Pokédex

Introduction

The goals of this lab are:

- Learn to integrate external APIs in application development.
- Implement a user-friendly Pokédex application displaying various attributes of Pokémon.

Our course webpage: https://www.cs.wcupa.edu/schen/ios23/

Lab Instructions

Objective

Develop a Pokédex Application showcasing the details of Pokémon (No.1 – 151). The application should effectively display important attributes of each Pokémon, accompanied by a profile picture.

Key Concepts to Cover

- API integration
- User interface design

Requirements

1. Make use of the provided API Address: https://pokemon.wcpc.fun/id/:pokemon_id
2. Integrate **at least six attributes** from the Pokémon API above, with "name" being a mandatory field. Other possible attributes can include weight, height, base experience, etc.
3. Display the Pokémon's profile picture alongside its attributes in a user-friendly manner.
4. Implement a search feature allowing users to input a Pokémon ID and fetch the corresponding information.
5. For every Pokémon, request data from https://pokemon.wcpc.fun/gpt/:pokemon_id to obtain a description provided by ChatGPT. Display this description within the application.
6. Prioritize aesthetics and user experience. The interface should be polished, intuitive, and visually appealing.

Evaluation Criteria

- Successful API integration and data retrieval (60%)
- Proper UI/UX design reflecting both aesthetics and user experience (10%)
- Functional and efficient search feature (10%)
- Code readability and comments (20%)

Hints

- **Fetching Pokémon Data:** To fetch Pokémon data, the APIData class uses a method called fetchAPIData(). This method takes a Pokémon ID as an argument, makes an API call, and decodes the received JSON data into a PokemonData struct.
- **Updating UI with Fetched Data:** After fetching the Pokémon data, the ‘updateState(with:)‘ method is used to update the ‘@Published‘ state variable ‘pokemonName‘.

  ```
  private func updateState(with PokemonData: PokemonData) {
      self.pokemonName = PokemonData.name
  }
  ```

- **Displaying Pokémon Image:** In the ‘ContentView‘, the fetched Pokémon’s image is displayed using the Pokémon ID formatted as a 3-digit string.

  ```
  let pokemonID_string = String(format: "\%03d", self.pokemonID)
  Image(pokemonID_string)
  ```

- **Searching for a Pokémon:** The ‘ContentView‘ provides a TextField where users can input a Pokémon ID. Pressing the ”Search” button then triggers the ‘fetchNew(PokemonID:)‘ method of the ‘APIData‘ class.

  ```
  Button("Search") {
      self.pokemonID = Int(pokemonID_String) ?? 0
      apiData.fetchNew(PokemonID: self.pokemonID)
  }
  ```

- **Observable and ObservedObject:** Note the use of ‘ObservableObject‘ in the ‘APIData‘ class and ‘@ObservedObject‘ in the ‘ContentView‘. This allows the view to observe and react to changes in the ‘APIData‘’s state.

  ```
  class APIData: ObservableObject { ... }
  @ObservedObject var apiData = APIData(PokemonID: 1)
  ```

Deliverables

Submit your whole project via compressed zip file to D2L under Lab 2. Your folder should include both code and project setting files (FetchAPI.xcodeproj).

Submission

- Check the lab due date on the course website. Late submissions will not be accepted.
- Submit your assignment to D2L directly.

- **No copy or cheating is tolerated**. If your work is based on others' or AI, please give clear attribution. Otherwise, you **WILL FAIL** this course.