Functional Programming

Group Project - Team 12

WeatherWander Application Report

Introduction :

WeatherWander is a Haskell-based command-line application that offers personalised recommendations for places to visit based on current weather conditions. This report details how to compile, run, and use WeatherWander, explains its data source and information extraction process, and highlights features implemented in the application.

Compilation and Execution:

Prerequisites:

- Install the Haskell toolchain, including GHC and Stack, a build tool for Haskell.

Compilation Steps:

1. **Clone the Repository**: Clone the WeatherWander repository from GitHub:
   ```sh
   git clone https://github.com/shashankyadav03/WeatherWander.git
   ```
2. **Navigate to Project Directory**:
   ```sh
   cd WeatherWander/haskell-project
   ```
3. **Build the Project**: Use Stack to compile the project:
   ```sh
   stack build
   ```

Running the Application:

- To run WeatherWander, execute the following command in the terminal:
  ```sh
  stack run show <city>
  ```
- Replace `<city>` with the desired city's name to get weather-based activity recommendations. If the city name is more than one word use double quotes.

Usage:

WeatherWander operates on an intelligent flow to provide the best user experience while optimising resource usage:

1. City Input: The user inputs a city name.
2. Database Check: The application checks if the city's weather data is already present in the local database and whether it was updated within the last hour.
    - If the data is recent, it is used directly from the database.
    - If not, WeatherWander fetches the latest weather data from the WeatherStack API.
3. Database Update/Insertion: Weather data is then updated or inserted into the database.
4. Activity Suggestions: Based on the current weather conditions (temperature, chance of rain, etc.), the application queries the database for suitable activities and locations in the specified city.

Web Source and Information Extraction:

WeatherWander utilises the WeatherStack API for real-time weather data. This API offers comprehensive weather information, including temperature, precipitation chances, and more. The application makes HTTP requests to the API endpoint, passing the city name as a parameter. The response, a JSON object, is then parsed to extract relevant weather data. This data is used to update the local database and to determine suitable activities based on the current weather.

Extra Features and Innovations:

Database Optimization:

WeatherWander intelligently uses a local database to store recent weather data. This optimization reduces the frequency of API calls, ensuring efficient operation and minimising the risk of exceeding the API's rate limits.

Error Handling and User Feedback:

Robust error handling mechanisms are in place to ensure the application runs smoothly. These include handling HTTP exceptions, JSON parsing errors, and database-related issues. Clear and informative messages are displayed to the user in case of any errors.

Enhanced User Experience:

The application features a user-friendly command-line interface (CLI) that simplifies interaction. It provides clear instructions and feedback, making it accessible even to those unfamiliar with CLI applications.

Beyond Specifications:

WeatherWander goes beyond basic requirements by implementing features like database caching, comprehensive error handling, and an intuitive user interface. These additions enhance the application's performance, reliability, and user experience.

Conclusion:

WeatherWander exemplifies the effective use of Haskell in developing a practical and user-friendly application. Its integration with real-time weather data, efficient use of a local database, and thoughtful user interface design make it a valuable tool for planning outdoor activities. The additional features and optimizations ensure that WeatherWander not only meets but exceeds the expected functionalities, showcasing the potential of Haskell in real-world applications.

Team 12 Details:

Student Name	:	Shashank Yadav
Student ID	:	220961493
Student Email	:	ec23633@qmul.ac.uk
Student Name	:	Sloan Dcunha
Student ID	:	230792506
Student Email	:	ec23771@qmul.ac.uk
Student Name	:	Pierre Colaco
Student ID	:	230768361
Student Email	:	ec23609@qmul.ac.uk
Student Name	:	Atharv K Sawant
Student ID	:	230825343
Email.	:	ec23615@qmul.ac.uk