Assignment 1: RuleCrafter

Overview

RuleCrafter is a web application designed to manage and evaluate rules defined in a SQL-like syntax. The application allows users to create, store, display, and evaluate rules against user-provided data. The backend is powered by SQLite for data storage, while the frontend is built using Streamlit, providing a simple and intuitive user interface.

Features

1. Rule Management:

- Users can add new rules in SQL format, which are then parsed and stored in a SQLite database.
- Existing rules can be viewed, and users have the ability to remove any rule they no longer need.

2. Rule Evaluation:

- Users can input their data (age, experience, department, salary) and evaluate existing rules against this data.
- The application interprets the rules and determines whether the provided data satisfies each rule.

3. Abstract Syntax Tree (AST) Representation:

- Rules are parsed into an Abstract Syntax Tree (AST) to facilitate logical evaluations.
- The application employs a set of classes to represent different components of the rules, such as conditions and logical operations.

4. User Interface:

- The UI is designed with a clean layout, featuring sections for adding new rules, displaying existing rules, removing rules, and evaluating them based on user input.

Technical Implementation

1. Database Setup

The SQLite database is initialized, and a table named `rules` is created if it does not already exist. This table stores the rule strings along with their unique identifiers.

2. Abstract Syntax Tree (AST)

The application defines an AST using various classes to represent different elements of the rules:

- ASTNode: A base class for all AST nodes.
- BinaryOperation: Represents binary operations like comparisons (e.g., `age = 30`).
- Literal: Represents constant values.
- Field: Represents fields from the user data (e.g., `age`, `department`).
- LogicalOperation: Represents logical operations (AND, OR) between conditions.

3. SQL to JSON Conversion

The function `sql_to_json` is responsible for converting SQL-like rule strings into a JSON format that can be easily processed. This function uses regular expressions to extract conditions and handles both single conditions and logical operations.

4. Rule Parsing and Evaluation

Once a rule is added, it is parsed into an AST for evaluation:

- The `parse_rule` function constructs the AST based on the JSON representation.
- The `evaluate_ast` function recursively evaluates the AST against the user-provided data, returning a boolean indicating whether the conditions are met.

5. User Interface with Streamlit

The frontend of RuleCrafter is developed using Streamlit, which allows for rapid deployment of web applications with Python. Key components of the UI include:

- A text area for rule input.
- Displays for existing rules.
- Selectors for removing rules.
- Input fields for user data (age, experience, department, salary).
- A button to evaluate rules.

6. Rule Evaluation Results

When the user clicks the "Evaluate Rules" button, the application retrieves all stored rules, evaluates them against the provided user data, and displays the results, indicating which rules are satisfied.

User Experience

- Adding a Rule: Users can easily input rules in a friendly SQL format. Upon submission, they receive immediate feedback on whether the rule was successfully added.
- Viewing Existing Rules: A clear list of existing rules helps users keep track of their configurations.
- Removing Rules: The application simplifies rule management by allowing users to remove outdated rules.
- Evaluating Rules: Users can quickly check if their data meets the defined criteria, receiving clear visual feedback on their results.

Error Handling

The application includes error handling to catch issues during rule processing and database operations. Users receive feedback if:

- The rule string is empty.
- There are errors in parsing the SQL statement.
- There are any issues during rule evaluation.

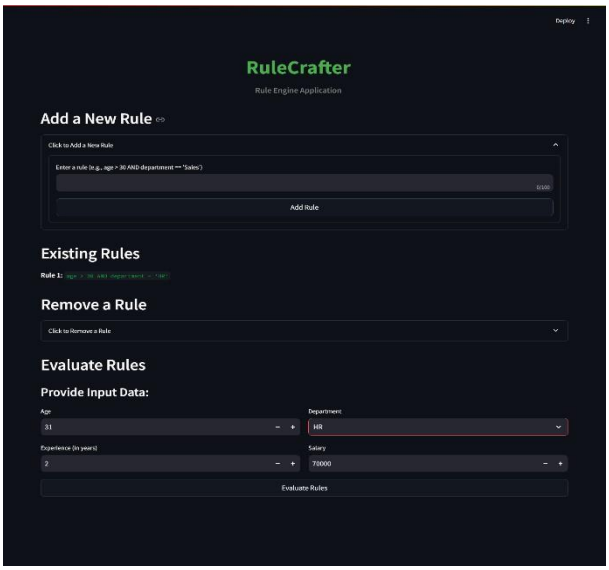
Conclusion

RuleCrafter is an innovative and user-friendly application that simplifies the management and evaluation of rules in a SQL-like format. With its intuitive interface and robust backend, it serves as a powerful tool for users needing to create and enforce rules based on dynamic data inputs. Future improvements could include:

- Enhanced parsing capabilities to support more complex SQL syntax.
- A more extensive user interface for managing large sets of rules.
- Integration with external databases or APIs for dynamic data evaluation.

This project demonstrates a practical application of abstract syntax trees and provides valuable insights into rule-based logic evaluation, making it an excellent resource for developers and data analysts alike.

Screenshots of the RuleCrafter



(GitHub Repository) Code Link: https://github.com/Ayushverma135/ZeotapAssignment

Assignment 2: WeatherPro

Overview

WeatherPro is a web application designed for real-time weather monitoring, allowing users to check current weather conditions and five-day forecasts for various cities. The application provides a simple and intuitive user interface built using Streamlit and fetches weather data from the OpenWeatherMap API. Users can add cities to their watchlist and receive timely updates on weather conditions.

Features

1. Current Weather Updates

- Users can enter city names to view the current weather conditions, including temperature, humidity, and weather description.
- Weather icons are displayed based on the current weather conditions, enhancing visual representation.

2. Five-Day Forecast:

- The application provides a five-day weather forecast, including daily temperature highs and lows, humidity levels, and conditions.
- Each forecast entry is clearly labeled with the corresponding weekday (e.g., Monday, Tuesday).

3. Add City Functionality:

- Users can easily add cities to their watchlist by entering the city name and clicking the "Add City" button.
- The added cities are stored in memory and can be displayed together with their weather updates.

4. Responsive Design:

- The user interface is designed to be visually appealing and responsive, ensuring a seamless experience across various devices.

5. Scheduled Updates:

- Weather data is fetched automatically at regular intervals (every 5 minutes) to keep users informed.

Technical Implementation

1. API Integration:

- The application integrates with the OpenWeatherMap API to fetch weather data. Users must sign up for a free API key to access the service.
- The current weather and five-day forecast are retrieved using separate API calls.

2. Data Processing:

- The retrieved data is parsed to extract relevant information such as temperature, humidity, weather conditions, and icon codes.
- The application organizes this data into a user-friendly format for display.

3. User Interface with Streamlit:

- The frontend is built using Streamlit, which facilitates rapid deployment of web applications with Python. Key components of the UI include:
    - An input field for city names.
    - A button to add cities to the watchlist.
    - Display sections for current weather and five-day forecasts.
    - Weather icons corresponding to the weather conditions.

4. Scheduler Setup:

- APScheduler is utilized to periodically fetch weather updates every 5 minutes. This ensures that users always have access to the most current weather data.

5. Weather Widget Display:

- The current weather and forecasts are displayed in a widget-style format, making it easy for users to read and interpret the information.

User Experience

- Adding a City: Users can easily input city names, and upon submission, the application fetches and displays the corresponding weather data.
- Viewing Weather Updates: Current weather conditions and the five-day forecast are displayed in a clean and organized manner, allowing for quick reference.
- Weather Icons: Icons representing the weather conditions enhance the visual appeal and provide immediate context for users.

Error Handling

The application includes error handling to manage issues that may arise during data fetching and user input. Users receive feedback if:

- The city name input is empty or invalid.
- There are errors in fetching data from the OpenWeatherMap API (e.g., city not found).
- Network issues prevent data retrieval.

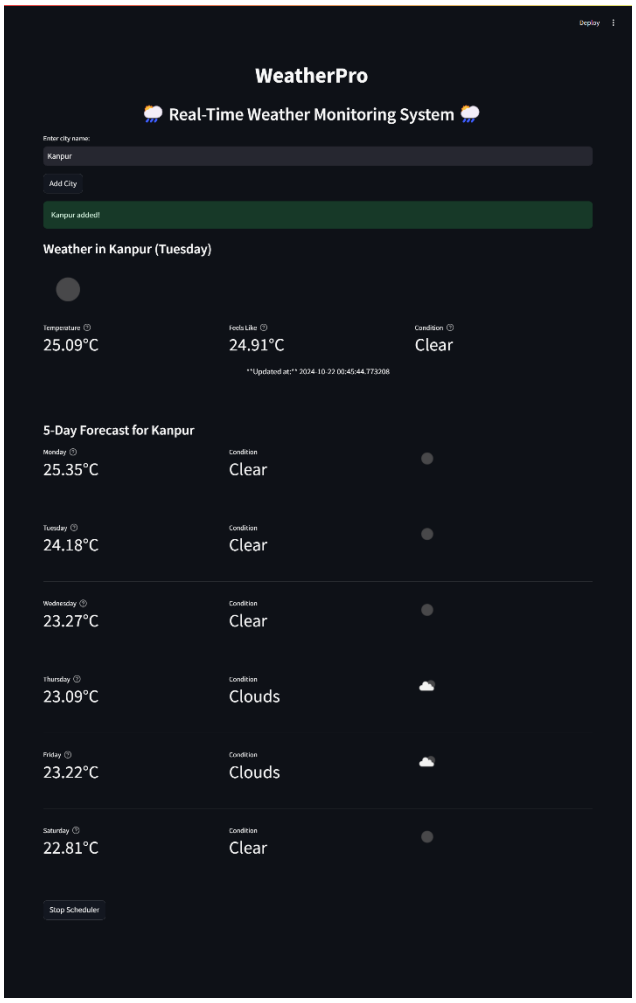
Conclusion

WeatherPro is an innovative and user-friendly application that simplifies real-time weather monitoring. With its intuitive interface and reliable backend, it serves as a powerful tool for users who want to stay informed about weather conditions in their chosen cities. Future improvements could include:

- User accounts to save preferred cities across sessions.
- Enhanced visualizations, such as temperature trends over time.
- Support for additional weather metrics and historical data.

This project demonstrates the practical application of web technologies and APIs, providing valuable insights into real-time data handling and user interface design.

Screenshots of WeatherPro



(GitHub Repository) Code Link: https://github.com/Ayushverma135/ZeotapAssignment