Project Title: Weather Application

Architecture Name: Model View Controller Architecture

Reasoning:

The Weather Application employs the Model-View-Controller (MVC) architecture to provide a well-structured, maintainable, and scalable solution for presenting weather information to users. MVC separates the application into three interconnected components, each with distinct responsibilities. This separation enhances code organization, promotes reusability, and facilitates collaboration among development teams. The model represents the data and business logic, the view handles the presentation of information to the user, and the controller manages user interactions and updates to the model and view.

Architecture Design:

1. Model:

- The model component encapsulates the application's data and business logic related to weather information.

- It includes classes, data structures, and methods for retrieving, processing, and storing weather data obtained from external sources such as APIs or databases.

- The model notifies registered observers (including the controller and view) of changes to the weather data, ensuring synchronization between components.

2. View:

- The view component is responsible for presenting the weather information and user interface elements to the user.

- It includes graphical elements, such as charts, maps, and text displays, to visualize weather data in an intuitive and informative manner.

- The view subscribes to updates from the model and refreshes its display to reflect changes in the weather data.

- User interactions with the view, such as selecting a location or adjusting settings, are forwarded to the controller for handling.

3. Controller:

- The controller acts as an intermediary between the model and view, handling user inputs and coordinating updates.

- It receives input from the user via the view and translates these actions into operations on the model or view.

- Based on user interactions, the controller may update the model with new data, trigger data retrieval from external sources, or modify the view's presentation.

- The controller ensures that changes to the model are reflected in the view and vice versa, maintaining consistency throughout the application.

Communication Flow:

1. User interacts with the view by providing input or triggering actions.

2. The view forwards user inputs to the controller.

3. The controller processes the inputs, updates the model if necessary, and instructs the view to update accordingly.

4. The view retrieves data from the model and presents it to the user.

5. Any updates or changes to the data are propagated from the model to the view via the controller, ensuring synchronization.

Benefits of MVC:

- Modularity: MVC separates concerns, making it easier to manage and maintain individual components.

- Reusability: Components like models and views can be reused across different parts of the application, promoting code reuse and reducing redundancy.

- Testability: The clear separation of responsibilities enables easier unit testing of individual components, leading to improved code quality and reliability.

- Scalability: MVC's modular structure facilitates scalability, allowing components to be added, modified, or replaced without affecting other parts of the application.

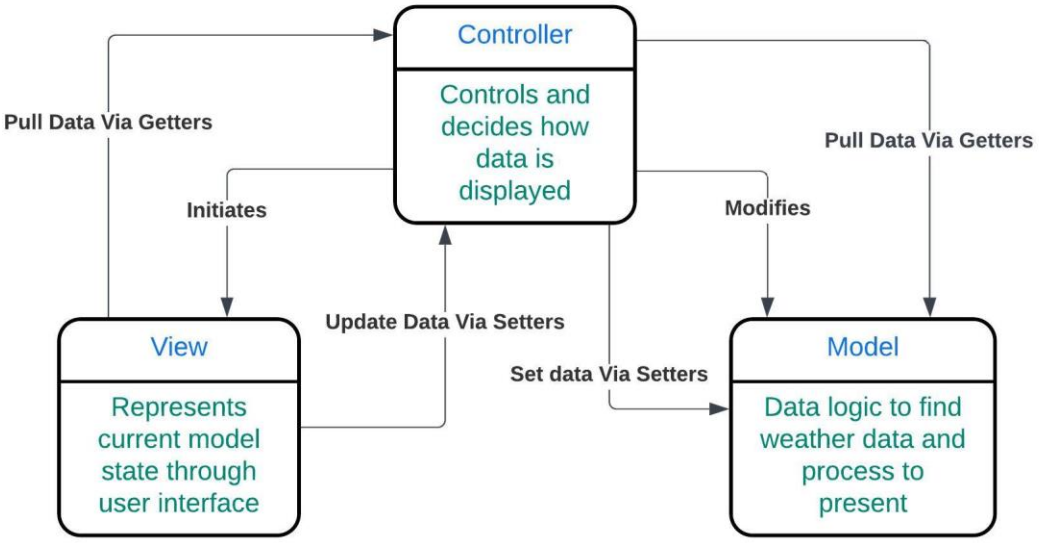

Fig : MVC Pattern

By adopting the MVC architecture, the Weather Application achieves a clean separation of concerns, enabling efficient development, maintenance, and scalability while delivering a user-friendly and reliable weather forecasting experience.

.