A **Weather Forecast App using the OpenWeatherMap API** -how it aligns with your syllabus:

✅ UNIT I: Spring Boot Basics & Project Structure

- **Setting up project** using Maven or Gradle
- Creating a **basic Spring Boot application**
- Understanding the @SpringBootApplication annotation
- Managing configuration with application.properties

✅ *Covered when students initialize the Spring Boot project and configure API keys and endpoints.*

✅ UNIT II: RESTful Services

- **Creating REST Controllers** to fetch and display weather data
- **Using annotations** like @RestController, @GetMapping, @RequestParam
- **Consuming REST APIs** using RestTemplate or WebClient

✅ *Students consume OpenWeatherMap REST API, parse responses, and serve data via their own API endpoints.*

✅ UNIT III: Data Persistence & Security

- Save **search history** or **user preferences** using **Spring Data JPA**
- Use an embedded database like H2 or connect to MySQL/PostgreSQL
- Implement basic **Spring Security** (e.g., user login to store preferences securely)

✅ *Students learn CRUD operations, data relationships, and simple login functionality.*

✅ UNIT IV: Microservices & WebFlux

- Optional: Break the app into **microservices** (one for weather, another for user settings)
- Use **WebClient** for reactive API calls
- Implement **Spring Boot testing** with @WebMvcTest and @DataJpaTest

✅ *Stretch goal: add reactive programming with WebFlux and basic actuator endpoints.*

✅ UNIT V: Reactive Persistence (Optional/Advanced)

- Save reactive data using MongoDB or Cassandra if going for a **fully reactive version**
- Use **Spring Data MongoDB** to store weather logs

✅ *This can be offered as an optional enhancement for advanced students.*

✅ Course Outcomes (COs) Mapping

- **CO1**: Understanding Spring Boot → *Setting up app and dependencies*
- **CO2**: Developing RESTful services → *Fetching/displaying weather data*
- **CO3**: JPA and Security → *User preferences, authentication*
- **CO4**: Reactive APIs → *Use WebClient, test reactive components*
- **CO5**: Reactive persistence → *Optional if MongoDB is used*

🔧 Enhancement Ideas

- Add **JWT authentication** for advanced security.
- Create a **dashboard UI** using Thymeleaf or React (optional).
- Implement **caching** using Spring Cache for repeated weather queries.
- Add **unit & integration tests** for controllers and services.