Homework #2
CS300/500

Programming Assignment: Event Management System

Objective:
Create an Event Management System that utilizes inheritance and design patterns to manage different types of events (e.g., workshops, concerts, and conferences). The system should allow users to create, manage, and search for events efficiently.

Requirements:

1. **Base Class**:
   - Define a base class Event with common attributes and methods for all events. Attributes might include eventName, eventDate, location, and methods like getDetails() and isUpcoming().
2. **Derived Classes**:
   - Create at least two (CS300) or three (CS500) derived classes: Workshop, Concert, and Conference. Each class should have its own specific attributes (e.g., Workshop might have duration and instructor, Concert might have bandName and genre, Conference might have speakers and topics) and override the getDetails() method to display relevant information.
3. **Vector of Pointers**:
   - Use a std::vector to store pointers to Event objects.
4. **Design Patterns**:
   - Implement the **Factory Method** pattern to create instances of Event subclasses. This should facilitate the easy addition of new event types in the future.
   - Use the **Strategy Pattern** to allow users to search for events using different criteria (e.g., by date, by location, by type). Create separate strategies for each search criterion and allow the user to select one at runtime.
5. **User Interface**:
   - Design a console-based user interface that allows users to:
     - Create new events.
     - Search for events based on different criteria (using the selected strategy).
     - Display details of specific events.
     - List all upcoming events.
6. **Additional Features**
   - (Bonus for CS300, **Required for CS500**) Add a feature that allows users to RSVP for events and track attendance.
   - (Bonus for CS300 and CS500) Implement a notification system that sends alerts for upcoming events based on user preferences (e.g., daily or weekly reminders).

7. **MakeFile:**
    - Create a MakeFile with a default target that compiles the project, and a target called “run” that compiles then runs the project. If your Makefile is platform specific or includes paths that are only relevant to your computer, that is fine.
8. **Documentation**:
    - Write documentation explaining your design choices, including:
        - How inheritance was utilized in your design.
        - Why use of a vector of pointers.
        - Why using design patterns is beneficial.
        - Any challenges encountered during development.

Submission:

- Submit your code on GitHub.
- Include your documentation in the repository.
- Provide instructions on how to run your program.

Evaluation Criteria:

- Correctness of the implementation (40%)
- Effective use of inheritance and design patterns (30%)
- Code quality and readability (20%)
- Documentation and usability (5%)
- Makefile (5%)