NAME: ABBAS KHAN

REG NO: SP23-BSE-115

***Model-View-Controller MVC***

The app has a registration form (View), input handling (Controller), and data storage (Model). MVC separates these concerns for maintainability.

Implementation

**Model:** Stores kid/parent data (e.g., Kid, Parent classes).

**View:** UI components (e.g., registration form).

**Controller:** Handles user inputs (e.g., RegistrationController).

***Observer Pattern***

**Why**: Parents need real-time updates on their child’s activities (e.g., location, app usage).

**Implementation**

* Parents subscribe as observers to kid-related events (e.g., Parent implements Observer).
* The app notifies parents when predefined events occur (e.g., KidSubject triggers updates).

***Factory Method Pattern***

**Why:** Different types of monitoring features (e.g., location tracking, screen time) may need dynamic instantiation.

**Implementation:**

* MonitoringFeatureFactory creates specific features (e.g., LocationTracker, AppUsageMonitor).

***Singleton Pattern :***

**Why:** A single instance of the monitoring service ensures consistent data across the app.

**Implementation:**

* MonitoringService.getInstance() manages all kid data centrally.

***SOLID Principles***

* **Single Responsibility:** Each class handles one concern (e.g., AuthService for login).
* **Open/Closed:** Extend features (e.g., new monitors) without modifying existing code.
* **Dependency Inversion:** Use abstractions (e.g., IMonitoringFeature) for flexibility.

***Decorator Pattern***

**Why:** Dynamically add monitoring features (e.g., adding "geofencing" to basic tracking).

* Implementation:
* Extend BasicMonitor with GeofenceDecorator.

***Strategy Pattern***

**Why:** Switch monitoring algorithms (e.g., strict vs. relaxed tracking) at runtime.

* Implementation:
* Define ITrackingStrategy with implementations like StrictTracking.



