Pattern Mapping

| **Use Case** | **Design Pattern** | **Justification** | **Application in Architecture** |
| --- | --- | --- | --- |
| Set Reminder Type | Strategy | Multiple reminder behaviors (audio, visual, vibration) must be interchangeable without altering task logic. | A ReminderStrategy interface is defined with concrete implementations like AudioReminder, VibrationReminder, etc. The appropriate strategy is selected at runtime. |
| Trigger Reminders | Observer | Reminder system needs to notify users at the correct time without tightly coupling with specific notification mechanisms. | A ReminderClock acts as the subject and notifies registered observers (e.g., audio or visual handlers) when it's time for a task reminder. |
| Add Task with Priority & Accessibility | Decorator | Task attributes like priority, font size, or color need to be added dynamically without modifying the base task class. | A base Task class is extended at runtime with decorators such as PriorityTask, LargeFontTask, or HighContrastTask to apply extra behaviors or UI styles. |
| Save/Load Task Data | Singleton | Application requires a single point of access for managing local data storage consistently across different modules. | A TaskStorageManager class follows the Singleton pattern, ensuring only one instance handles SQLite interactions throughout the application lifecycle. |

# Task Matrix

| **Task** | **Responsible** |
| --- | --- |
| All Use Cases | All team members |