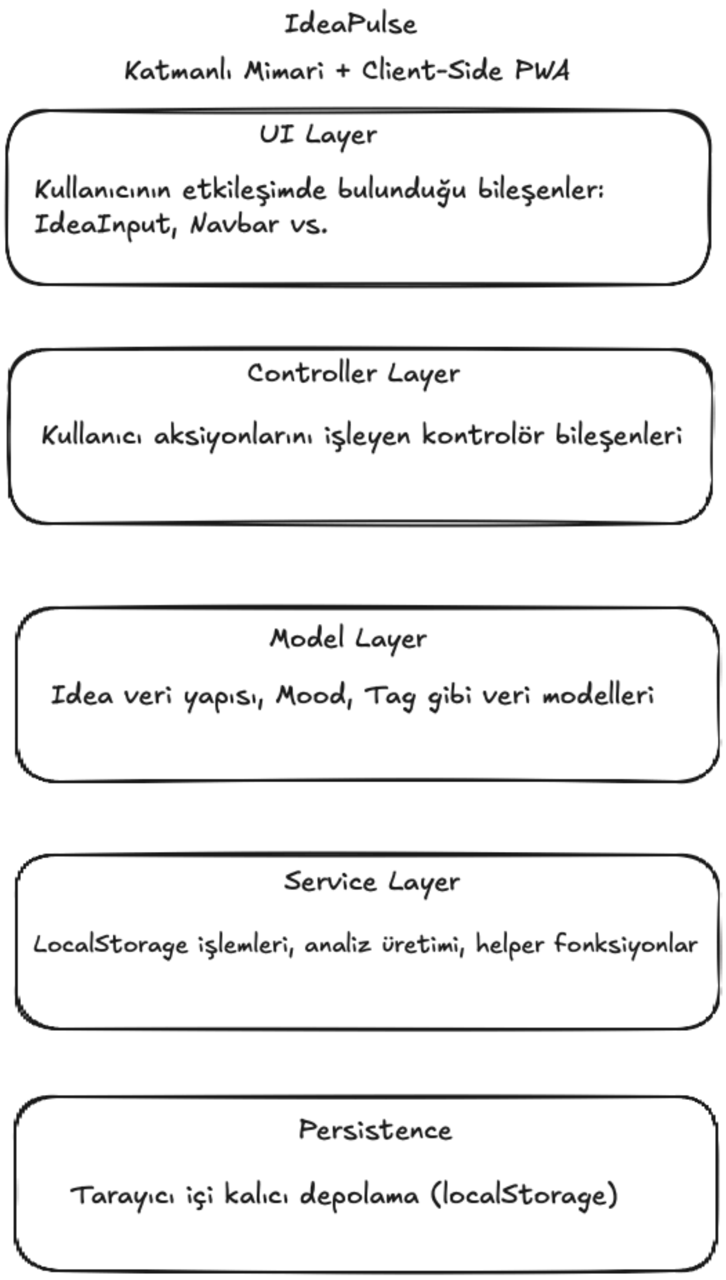
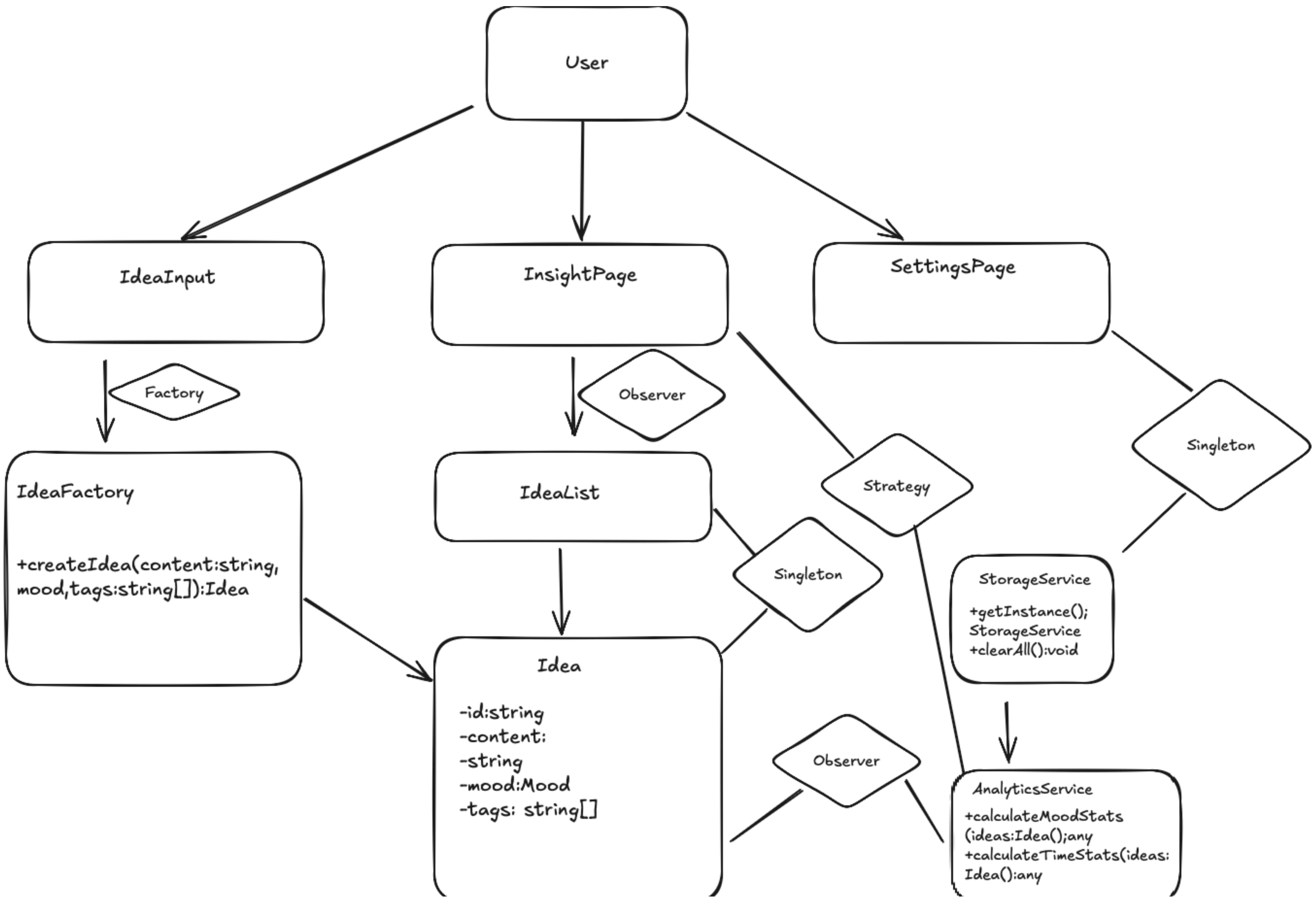
Software Architecture Overview + Design Patterns

1. Software Architecture Diagram


IdeaPulse
Katmanlı Mimari + Client-Side PWA
UI Layer
Kullanıcının etkileşimde bulunduğu bileşenler: IdeaInput, Navbar vs.
Controller Layer
Kullanıcı aksiyonlarını işleyen kontrolör bileşenleri
Model Layer
Idea veri yapısı, Mood, Tag gibi veri modelleri
Service Layer
LocalStorage işlemleri, analiz üretimi, helper fonksiyonlar
Persistence
Tarayıcı içi kalıcı depolama (localStorage)

2. Component Architecture Diagram



3. Pattern–Use Case Mapping Table

Use Case	Design Pattern	Description
Add Idea	Factory Pattern	The creation of `Idea` objects based on user input (e.g., title, content, mood) is managed via a centralized factory. This enables scalable and maintainable object creation, as new input sources or formats can be supported by modifying only the factory logic.
List Ideas	Observer Pattern	The UI's idea list updates automatically when state changes using React's state management. This decouples the data from the UI and ensures real-time synchronization without manual intervention.
Mood Analysis	Strategy Pattern	Different analysis methods (such as mood-based or tag-based) are encapsulated into individual strategies. This allows easy swapping or extension of analysis logic without modifying the core system.
Delete Data	Singleton Pattern	Application-wide resources like `AppSettings` or `StorageManager` are implemented as singletons, ensuring consistent access and state management across all components.

4. Pattern Implementation Explanations

Factory Pattern

Different ideas with various moods and tags are created using the **IdeaFactory**.

Observer Pattern

By using **useState** and **useEffect**, the **IdeaList** automatically updates whenever a new idea is added.

Strategy Pattern

In the future, alternative analysis types—such as time-based analysis—can be added alongside mood analysis. Each type is implemented as a separate strategy.

Singleton Pattern

The **AppSettings** structure, which manages the application's configuration, is created as a single instance and accessed globally throughout the app.