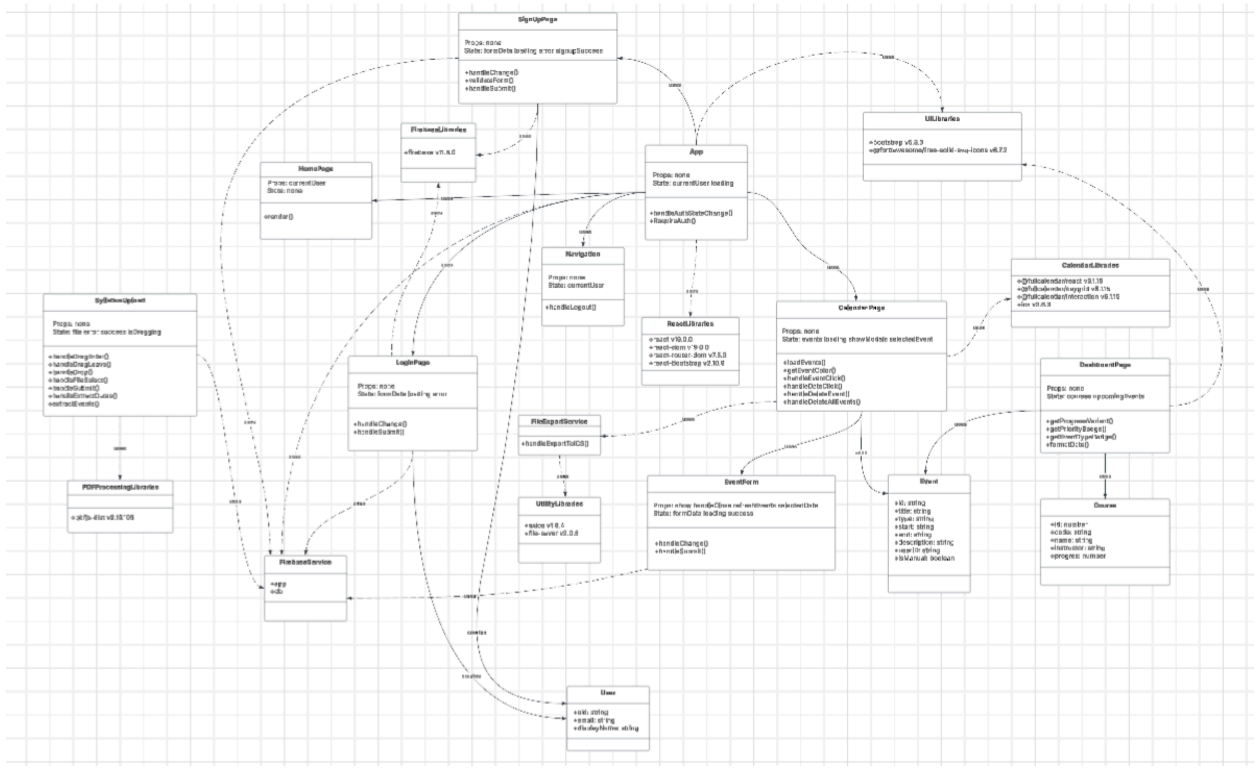
Design “Class” Diagram

DCD



The design diagram illustrates the structured architecture of the Align application through clearly defined component modules organized by functionality. The core components (App, Navigation, HomePage) form the foundation, with App serving as the central container that manages authentication state and renders all other interface elements. Authentication components (SignUp, Login) handle user registration and access control through Firebase Authentication services, while feature components (DashboardPage, SyllabusUpload, CalendarPage) implement the main application functionality including course tracking, PDF processing, and calendar visualization. The diagram highlights two key design patterns: the Observer Pattern implemented in CalendarPage, which maintains synchronization with Firestore for real-time data updates, and the Facade Pattern through FirebaseConfig, which encapsulates and simplifies Firebase interactions to provide a cleaner API for other components. Domain models (User, Course, Event) represent the data structures that flow through the application, creating a cohesive system where SyllabusUpload extracts events from documents, CalendarPage displays them, and DashboardPage summarizes upcoming deadlines. The relationship arrows clearly demonstrate how components interconnect, with solid lines showing rendering relationships and dashed lines indicating dependencies, creating a comprehensive yet maintainable architecture that promotes separation of concerns and component reusability.

Design Patterns

Singleton

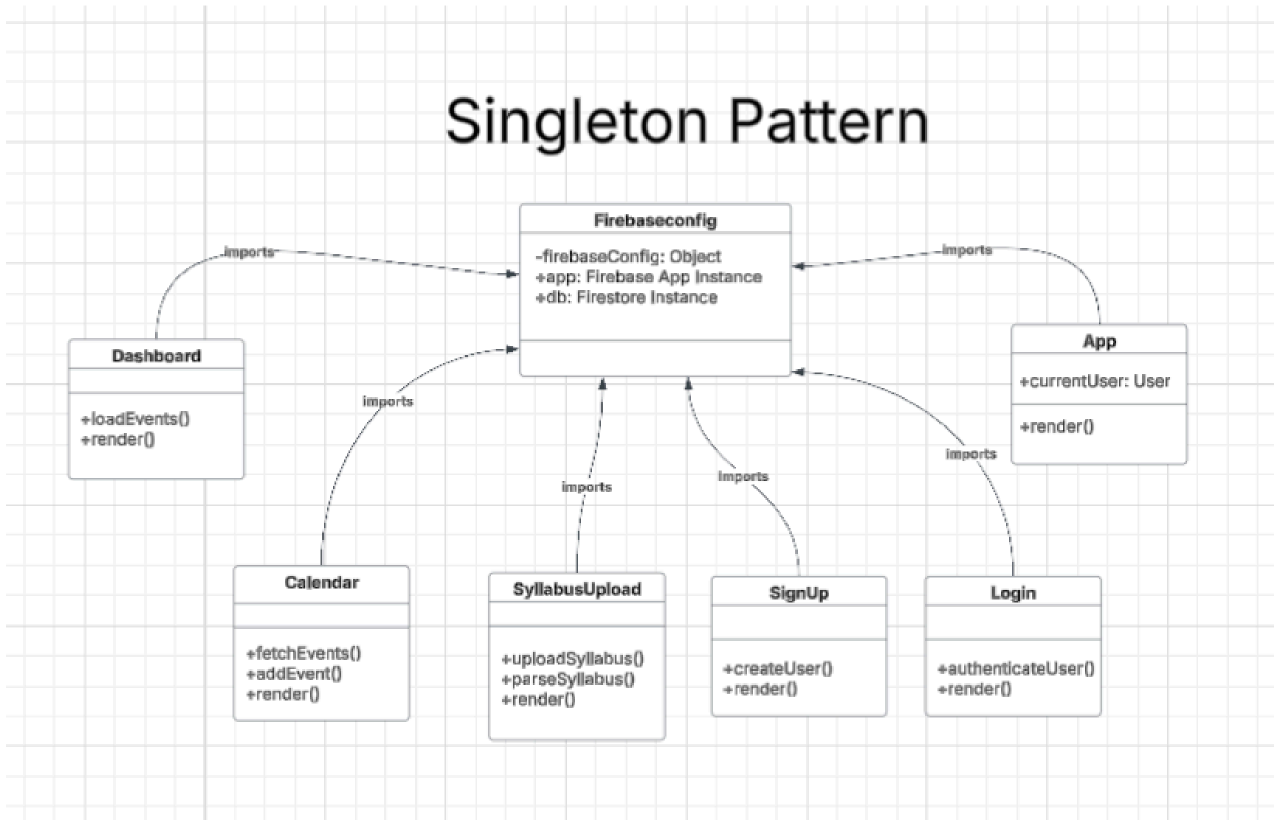
The Singleton design pattern for Firebase integration ensures consistent, application-wide access to resources. By creating single instances of the Firebase app and Firestore database in the firebase-config.js module and exporting these instances, it becomes the centralized point of access while guaranteeing only one connection exists throughout the application. This approach prevents redundant database connections, reduces resource consumption, and eliminates potential synchronization issues that could arise from multiple database instances.

Align App implements the Singleton pattern through JavaScript modules. Although it doesn't use the traditional class-based approach with private constructors and a getInstance method, it achieves the same goal by:

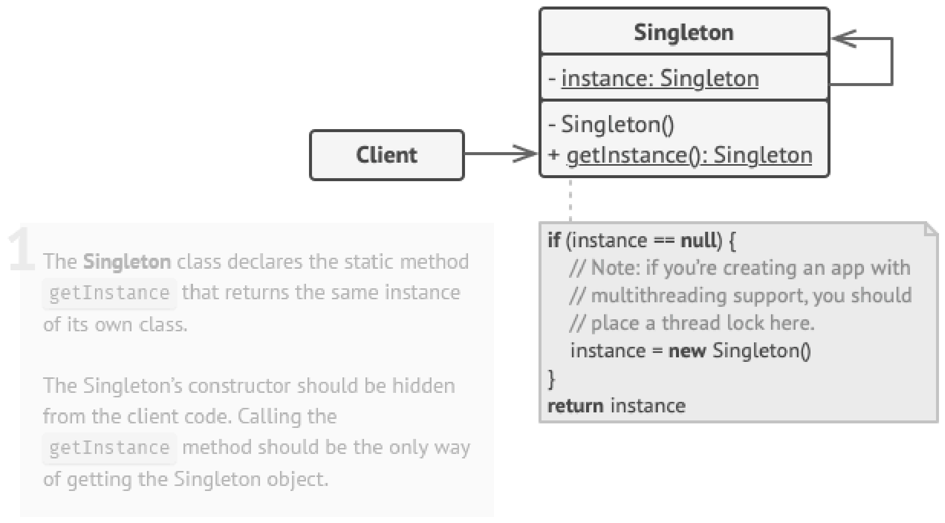
1. Creating single instances of Firebase app and Firestore database
2. Exporting these instances for global use throughout the application
3. Using the module system to ensure only one instance exists

This design decision was used for maintaining data integrity and optimizing performance in our application. Since multiple components (authentication, syllabus uploading, calendar management, and dashboard) all require database access, the Singleton pattern provides a clean architecture that avoids code duplication and ensures consistent database operations. Also, this implementation simplifies future maintenance as any changes to Firebase configuration only need to be made in a single location.

Singleton Creational Pattern



Singleton Pattern Example:



```
if (instance == null) {
   // Note: if you're creating an app with
   // multithreading support, you should
   // place a thread lock here.
   instance = new Singleton()
}
return instance
```

The **Singleton** class declares the static method `getInstance` that returns the same instance of its own class.

The Singleton's constructor should be hidden from the client code. Calling the `getInstance` method should be the only way of getting the Singleton object.