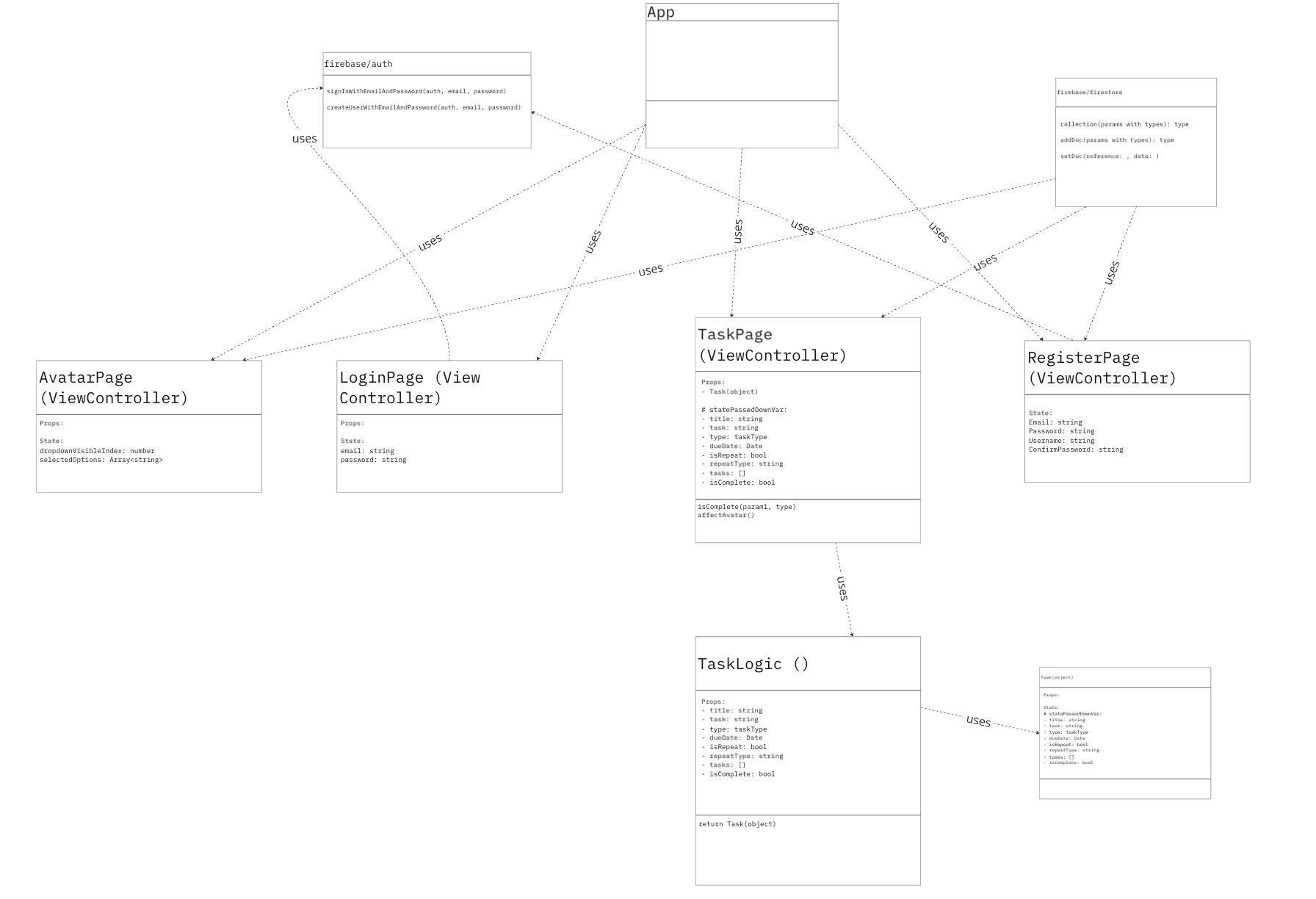
**Design Class Diagram**

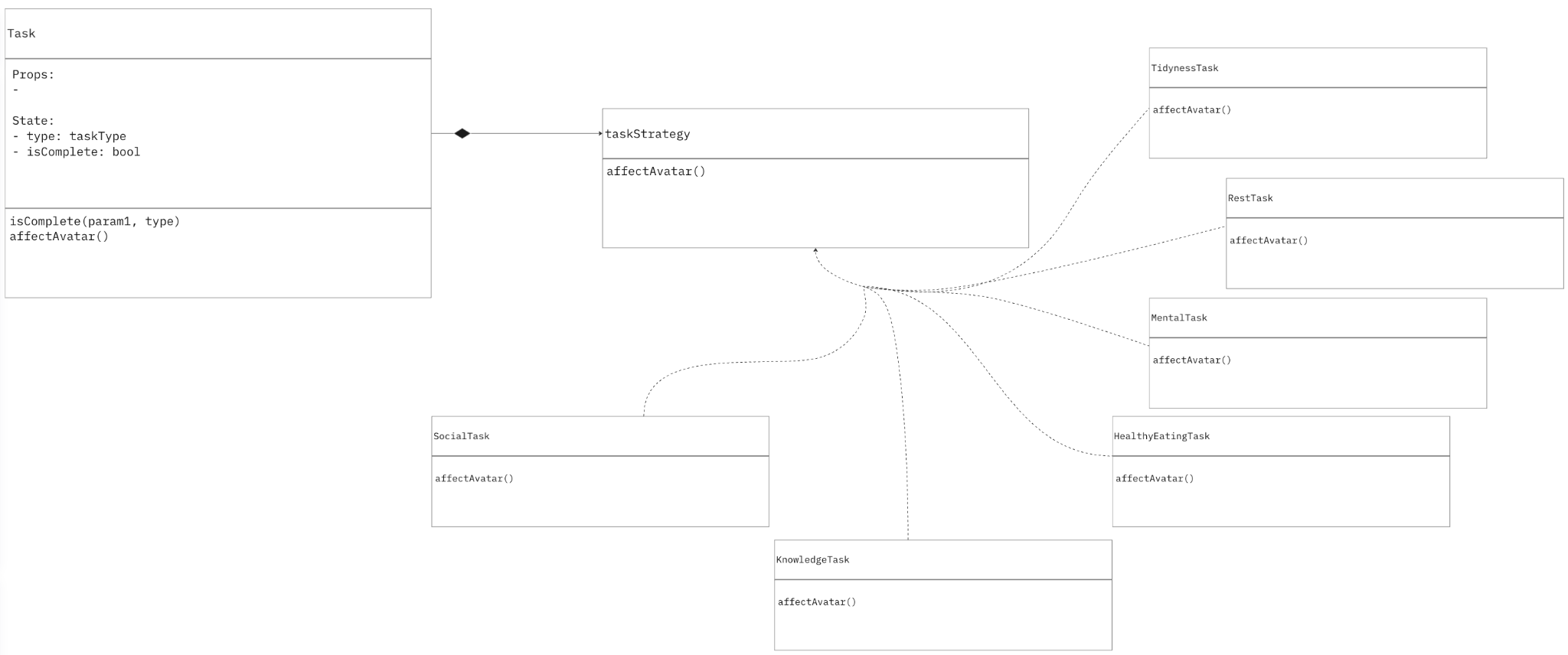
<https://miro.com/app/board/uXjVLEpzz4A=/> 

This diagram shows the relationship between the main components of our application. The main App holds the location of all the other view controllers for the purpose of page navigation. Login and register use FireBase authentication for creating and verifying user accounts. The TaskPage also uses FireBase for storing user tasks.

The TaskPage view controller utilizes a TaskLogic component, all task data gathered from the user on the view controller, sent to TaskLogic, and TaskLogic returns a filled Task component. Based on user task completion, the tasks affect the avatar on the avatarPage.

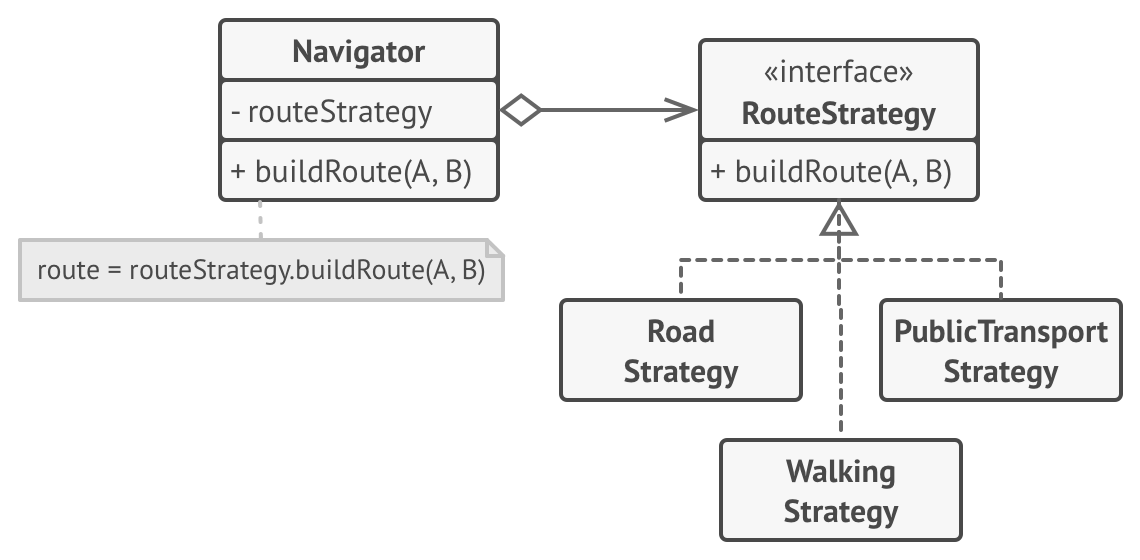
**Design Pattern**

Strategy Pattern -



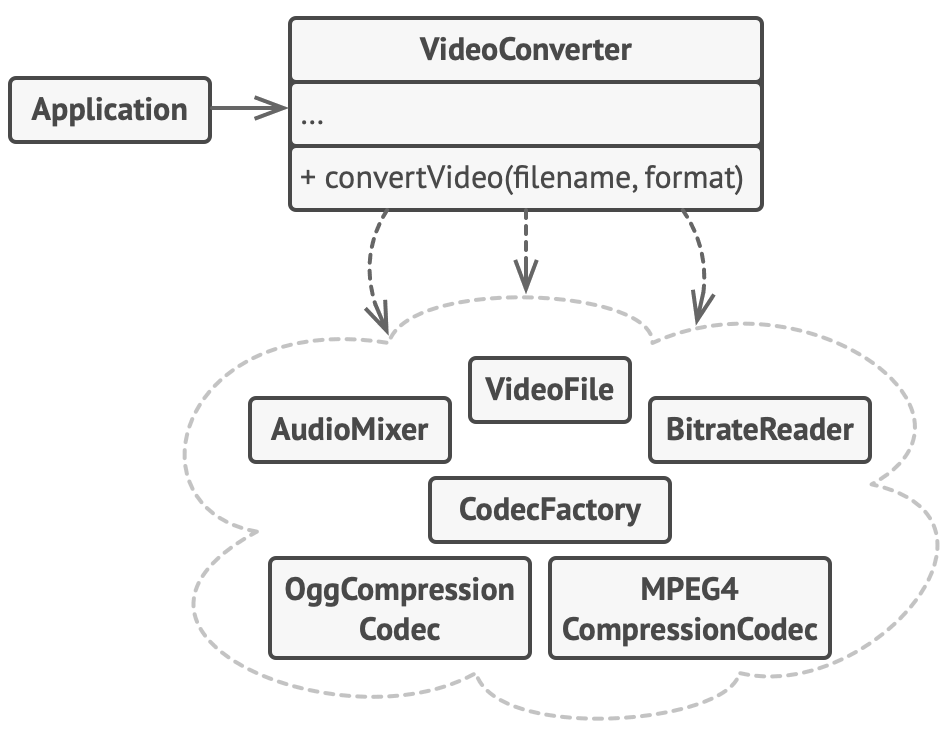
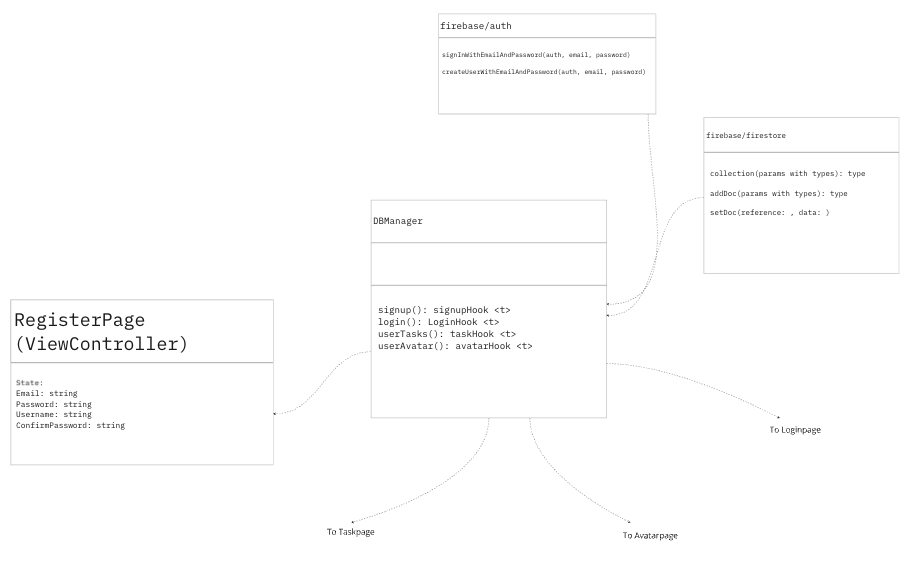
For implementing some task logic we followed the Strategy Design Pattern shown below.

<https://refactoring.guru/design-patterns/strategy>



The strategy pattern will be implemented on the different tasks types as each task type will affect the avatar differently. By having the strategy design implemented, the app should have an easier time applying the proper effects once a task is completed (or if a task is not completed). The plan is to have status bars on the avatar, and as tasks in a particular category are completed, that status bar goes up, but over time, if no tasks in a category are completed, the bar goes down.

Facade Pattern -



<https://refactoring.guru/design-patterns/facade>

For database management, we also plan to implement a facade design pattern. The database encapsulates all of the user’s tasks, information, and preferences from each view controller in the application. This allows for simple interfacing with the external Firebase library which handles storage and user verification.