|  |  |  |
| --- | --- | --- |
|  | Pros | Cons |
| Layered Architecture | * Lower layers can be reused * Suitable for interactive system | * Difficult to structure a system in layered architecture |
| Pipe & Filter Architecture | * Allows designer to simply understand input/output behavior of a system in terms of individual filters. | * Not suitable for interactive system |
| Client & Server Architecture | * Centralized Data * Better Security * Faster development process * Support Multiple views * Easier to maintain or modify | * Multiple clients lead to increased workload hence reducing network speed. * All clients will not work when the server fails. |

## **Reason for the candidate architecture**

We are selecting the Model-view-controller (MVC) architecture. MVC allows us to separate our development into 3 interconnected parts the model, the view, and the controller. The controller will be handling all requests of the application and then forwards instructions to the model to prepare information required. The view will be using the data provided by the controller and the model and displaying it.

Here are some reasons which made us chose MVC over other architecture.

1. Faster development process

With MVC, we can do rapid and parallel development. This means one developer can work on the view while the other works on the controller. This eliminates the need to wait for pre-requisites. In this way, we can theoretically complete the application 3 times faster compared to another candidate architecture.

1. Multiple views

With MVC, we can create many views for a single model. This allows us to reduce code duplication because it separates the data and business logic from the view.

1. Modularity

With many applications, the user interface tends to be changed to most. With this idea in mind, MVC allows us to change views without affecting the backend logic. For example, if the controller and model is set up correctly, we can fetch data from the database independently from the view. This is a strong point for using MVC and for future scalability.

1. Simple data passing

With MVC, we can pass data from component to component in a simple data format, JSON. With JSON, data is unformatted, and the client can then decide what to do with the data. Formatting JSON is also quite simple.

With the following 4 reasons, we strongly believe in the MVC architecture and it have also been proven to be a good development architecture by many others.

Thus, after weighing the cons and pros of using Model-view-controller (MVC) architecture and considering its ease and high suitability for interactive systems, we concluded that MVC candidate architecture is best suited architecture design for our system.