Design Pattern

The design pattern our system follows and suits our implementation the most is the Model View Controller (MVC). The MVC pattern divides our software part into three main components:

* **Model**: Includes the classes that represents stored data. The Model can only see and interact with the Controller.
* **View**: Includes the interface that the user can view and interact with
* **Controller**: Includes the system logic and interacts with both the View and Model components by retrieving inputted data from user and passing it to the model after processing any required logic with it.

The MVC structure explains the main steps our system follows. The user interacts with the user interface of our system (application or website) and whenever the user requests to input or output a certain data from the view, the view sends that request to the controller along with any inputted data. Afterwards, the controller processes the request and retrieves the needed data from the model, and then sends the appropriate output (results) back to the view to be shown to the user.

View

Controller

Model

Advantage of using this pattern:

* Separation of Concerns:

The MVC pattern separated our system into components that are independent from each other, this is called “separation of concerns”. A failure can easily be found because of the components independency, and an edit can be done on any of the components without disturbing other components.

* Re-usability and Flexibility:

The MVC pattern can increase the scalability of our system, this is because a function can easily be added or changed in a component without changing other components. Hence, our system is easily flexible to scalability. Additionally, different component of our system can be reused in other systems to give the same service.

* High Cohesion and Low Coupling:

The MVC model automatically makes the system have high cohesion and low coupling. This is because each component in the system must be related in functionality and serve the system with a specific job, which makes the functions in one component highly cohesive. Also, the different components in the system are lowly coupled, meaning changing one component, whether a major or minor change, does not interrupt the work of other components.