**SIR SYED UNIVERSITY OF ENGINEERING & TECHNOLOGY SOFTWARE ENGINEERING DEPARTMENT**

**Fall 2022**

**Web Engineering (SWE-315) Assignment# 2**

Semester: 6th Batch: 2020

Submission Date: 30/12/2022 Marks: 10

# Note:

1. **This is a group assignment with each group having maximum of 4 members.**
2. **Each member will present the information provided in the submitted document in class.**

Q1) Differentiate between following Architecture patterns with respect to Web Applications:

1. MVC (Model-view-controller)
2. MVVM (Model-view-view-model)
3. MVP (Model-view-presenter)
4. VIPER (View, Interactor, Presenter, Entity, and Router Model)

Consider any application scenario to show the implementation of above given architecture patterns.

**ANSWER:**

1. **MVC (Model-view-controller)**

Model, View, and Controller are the three primary logical components that are divided into an

application by the MVC Framework, an architectural pattern. MVC is hence an abbreviation.

Model View Controller is the acronym for this phrase.

In this design, a component is created to manage particular application development aspects. The business logic layer and presentation layer are separated by MVC. Its primary application is in desktop graphical user interfaces (GUIs).

The following three MVC elements are crucial:

**Model:** It contains all of the data and the logic that behind it.

**View:** Handles user interaction or displays info to the user.

**Controller:** A link between the Model and View subsystems.

**Model**

Data and associated logic are stored in the model component. It represents any connected business logic or data that is being moved across controller components.

A Controller object, for instance, enables you to obtain client information from the database. It modifies data and either uses it to render the same data or sends it back to the database.

**View**

A view is a representation of how data is presented within an application. The information received from the model data is used to construct views. In order to send the output to the user, a view asks the Model for information.

**Controller**

The component of the application that manages user interaction is called the Controller. The user's mouse and keyboard inputs are interpreted by the Controller, which then instructs the Model and the View to adjust as necessary.

The Model receives commands from a Controller to update its state (E.g., Saving a specific document.



1. **MVVM (Model-view-view-model)**

Clean, well-organized code is always preferred by developers for projects. The maintenance of the software is aided by organizing the codes in accordance with a design pattern.

Adding and removing app features is simpler. Additionally, design patterns ensure that all of the scripts are tested in unit tests without the interference of other classes.

The industry-recognized software architectural pattern known as Model-View-View Model (MVVM) fixes all the problems of MVP and MVC design patterns. The MVVM architecture recommends isolating the application's essential business logic from the views or user interface (UI) that present data.

**MVVM's various code layers are as follows:**

Model: The abstraction of the data sources is done at this layer. Together, the model and view model retrieve and store the data.

View Model is informed about the user's action by this layer, which serves this purpose. Without any application logic, this layer just observes the View Model.

Data streams that are pertinent to the view are exposed via the view model. It also acts as a connection between the Model and the View.

**Model**

Data and pertinent logic are stored in the model. It represents any connected business logic or data that is being moved across controller components.

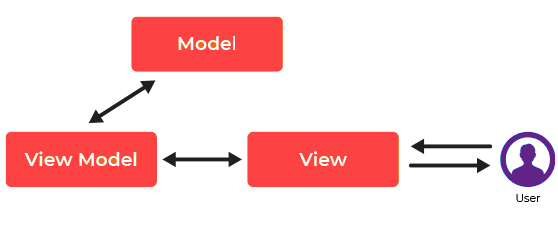
For instance, a Controller object will get student information from the institution's database. It modifies data and either uses it to render the same data or sends it back to the database.

**View**

The term "View" refers to user interface elements like HTML, CSS, jQuery, etc. As a result, the MVVC pattern view is held accountable for presenting the data that was received from the Controller. Additionally, this View changes the Model (s) into the User Interface (UI).

**View Model:**

To support the state of the View, the view model is in charge of displaying functions, commands, and methods. It is also responsible for running the model and turning on the View's events.



|  |
| --- |
| **KEY DIFFERENCE BETWEEN MVC & MVVM**  Model, View, and Controller are the three basic logical components that make up an application using the MVC framework, which is an architectural pattern. However, **MVVM** makes it possible to separate the creation of the graphical user interface using markup language or GUI code.  In MVC, the controller serves as the application's entry point, but in MVVM, the view serves in same capacity.  While MVVM is simple for separate unit testing since the code is event-driven, the MVC Model component may be tested independently from the user.  In contrast to MVVC design, which has "one to many" relationships between View and View Model, MVC architecture has "one to many" interactions between the Controller and View. |

1. **MVP (Model-view-presenter)**

MVP (Model — View — Presenter) architecture is one of the most popular architecture patterns and is valid in organizing the project. MVP (Model — View — Presenter) comes into the picture as an alternative to the traditional MVC (Model — View — Controller) architecture pattern. The MVP pattern **allows for easier mocking of the view and more efficient unit testing of applications**. In the MVP pattern, the presenter manipulates the model while simultaneously updating the view.

* **Model:** Layer for storing data. It is responsible for handling the domain logic(real-world business rules) and communication with the database and network layers.
* **View:** UI(User Interface) layer. It provides the visualization of the data and keep a track of the user’s action in order to notify the Presenter.
* **Presenter:** Fetch the data from the model and applies the UI logic to decide what to display. It manages the state of the View and takes actions according to the user’s input notification from the View.

1. **VIPER (View, Interactor, Presenter, Entity, and Router Model)**

VIPER (View, Interactor, Presenter, Entity and Router) is a [design pattern](https://www.techtarget.com/searchsoftwarequality/definition/pattern) for software development that develops [modular code](https://www.techtarget.com/searchsoftwarequality/definition/structured-programming-modular-programming) based on clean design architecture. The modules in VIPER are [protocol](https://www.techtarget.com/searchnetworking/definition/protocol)-oriented and each function, property input and output is performed by way of specific sets of communication rules.

VIPER is often used to develop smartphone apps and is heavily used in [iOS](https://www.techtarget.com/searchmobilecomputing/definition/iOS) along with [Swift](https://www.techtarget.com/whatis/definition/Apple-Swift), Apple’s programming language. VIPER is based on [SOLID](https://www.techtarget.com/whatis/definition/SOLID-software-design-principles) design principles, specifically the single

responsibility principle.The words represented in the VIPER acronym note the types of modules in the design pattern:

* **View** (also known as the view controller): Both displays information for the user and detects user interaction. The Presenter is the only module the View module has contact with.
* **Interactor**: Manipulates entities or models and fetches and stores data.
* **Presenter**: Contains the user interface and prepares data for presentation. The Presenter contacts the Interactor for data requests, the View to present prepared data to the user and the Router in order to hand off objects.
* **Entity**: The data models that are manipulated by the Interactor.
* **Router** (also known as [wireframe](https://www.techtarget.com/whatis/definition/wireframe)): Handles navigation in the module or application. It creates the View and wires the Presenter to act as output to the Interactor. Contacts the Presenter in order to route requests.

**KEY DIFFERENCE BETWEEN MVP & VIPER**

MVP model is easier to understand as it has a separate application model.There is no conceptual relationship between application components.Whereas in VIPER there is lack of availability of resources to learn the VIPER design,it’s still not common in [iOS developers](https://www.techaheadcorp.com/hire/ios-developers/) and requires a certain level of knowledge and expertise to use it.

It’s necessary to have its own controllers for a model. Hence, an application with increased models can’t withstand MVP if the number of controllers is not increased as well.On the other hand Viper makes the process scalable and enables developers to work simultaneously on a project.