ASSIGNMENT 1

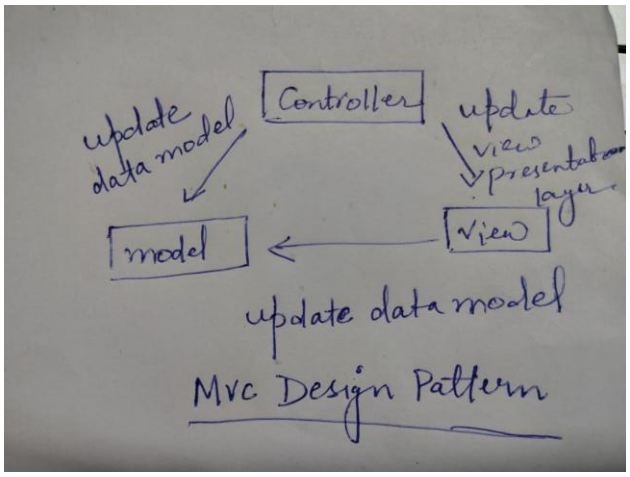
MVC(MODEL-VIEW-CONTROLLER)

What is MVC?

MVC (Model-View-Controller) is a software architectural pattern that separates an application into three interconnected components:

- **Model**: Manages data and business logic.
- **View**: Displays data (UI).
- **Controller**: Handles user input and updates Model/View accordingly

MVC Diagram:



```
[User]
    |
    V
[Controller] <-------> [Model]
   |                  |
   v                  |
  [View] <----------------
```

- **Flow**: User interacts with the **Controller**, which updates the **Model**. The **Model** notifies the **View** to refresh the display

Variants of MVC

1. MVP (Model–View–Presenter)

- **Presenter** replaces Controller and contains the presentation logic.
- The **View** is passive; the **Presenter** directly updates the **View**.
- Used in desktop/mobile applications.

MVP Diagram:

```
[User]
  |
  v
 [View] <--------> [Presenter] <--------> [Model]
```

- **Presenter** handles all logic and directly modifies the View.

Use When:

- You want easier unit testing (Presenter is easier to test).
- The View should be passive.

---

2. MVVM (Model–View–ViewModel)

- Common in frameworks like Angular, React, or WPF.
- **ViewModel** acts as a binder between Model and View.
- Uses data binding to sync View and ViewModel.

MVVM Diagram:

```
[View] <------ Data Binding ------> [ViewModel] <------> [Model]
```

- ViewModel exposes data streams/observables to which the View binds.

Use When:

- You have strong data-binding support in your framework.
- You want less boilerplate and reactive UI updates

When to Use Which:

Pattern	**Best Used When**
MVC	Web apps with simple input/output cycles.
MVP	Desktop/mobile apps with complex UI and need for testability.
MVVM	Applications using reactive or data-binding frameworks (e.g., Angular, React, WPF).