Software Engineering

Lecture 10

RIPHAH INTERNATIONAL UNIVERSITY

Agenda

- Software Architecture

Software architecture

- Software architecture is a high-level structure of a software system.
- Each structure comprises software elements, relations among them, and properties of both elements and relations.
- Its documentation is also called High-Level Design.

What is an architectural pattern?

- An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context.
- This is also known as Software Design Pattern.
- In the context of any software that architectural pattern is also called software architecture.

Difference Software Architectures

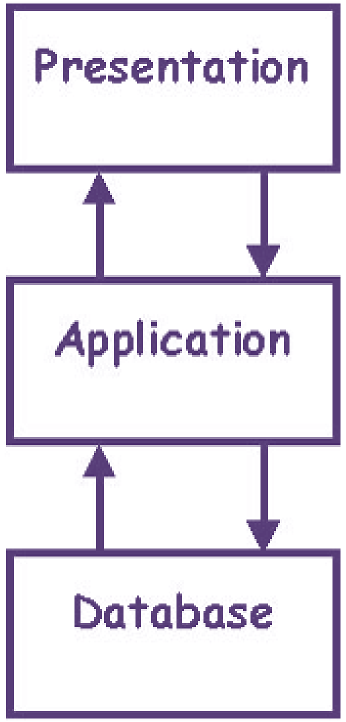
- Three-tier
- Model-view-controller
- Domain Driven Design
- Micro-Kernel
- Blackboard pattern
- Sensor-controller-actuator
- Presentation-abstraction-control

Selection

- The choice of architecture depends on various factors such as the nature of the application, scalability requirements, performance goals, team size, development budget, and deployment environment. Here are some common scenarios where different types of architectures are typically used:

N-Tier Architecture

- A diagrammatic representation of an n-tier system depicts here – presentation, application, and database layers.



- These three layers can be further subdivided into different sub-layers depending on the requirements.
- Some of the popular sites that have applied this architecture are
  - Indian Railways
  - Amazon

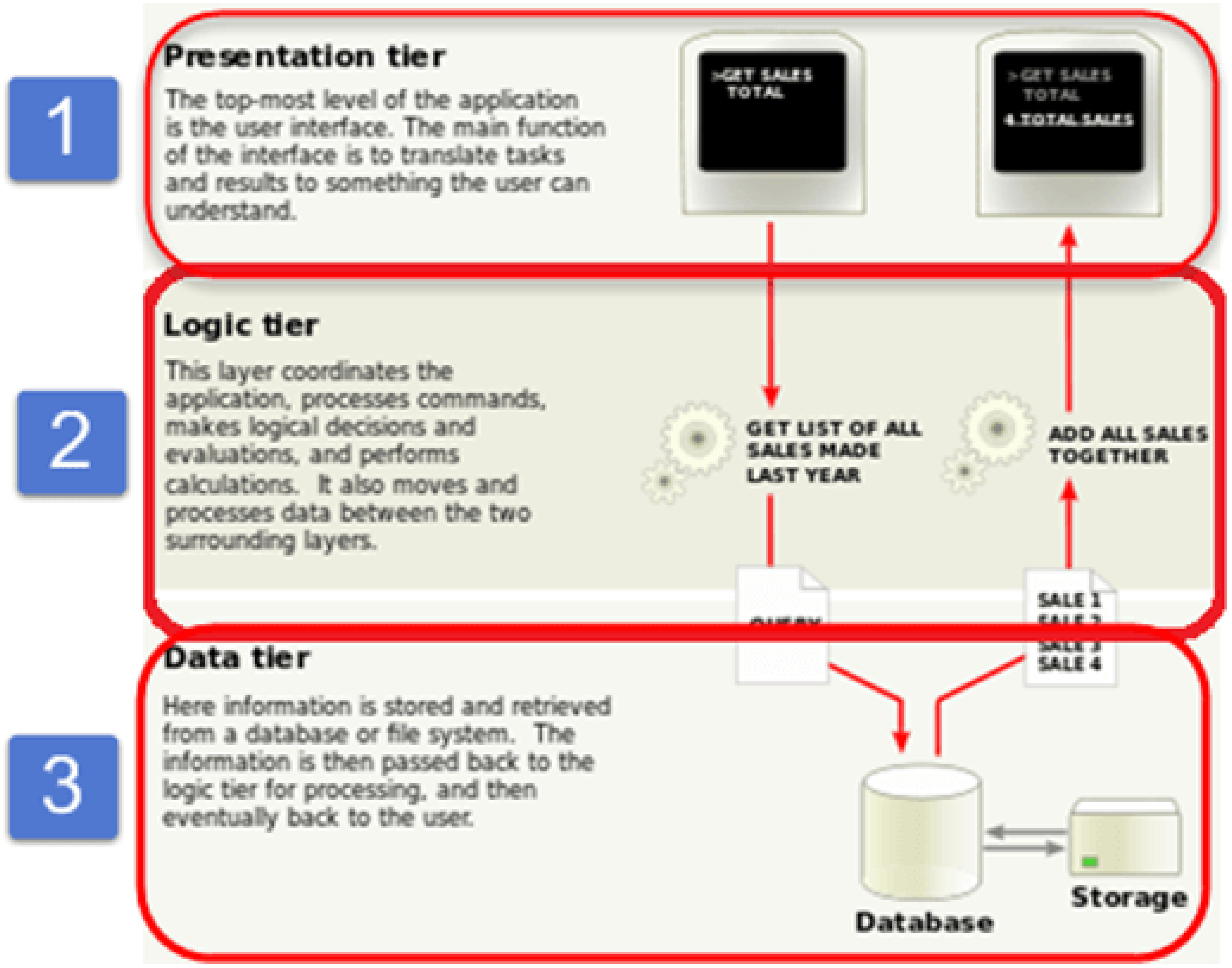
Types of N-Tier Architectures

- There are different types of N-Tier Architectures
  - 3-tier Architecture
  - 2-Tier Architecture
  - 1- Tier Architecture

3-Tier Architecture

- By looking at the diagram, you can easily identify that 3-tier architecture has three different layers.
    1. Presentation layer
    2. Business Logic layer
    3. Database layer

3-Tier Architecture



Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

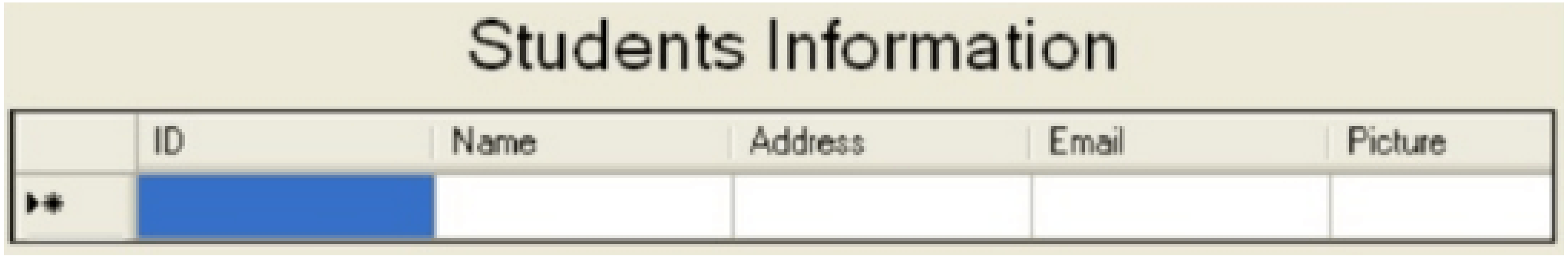
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

3-Tier Architecture

- Here we have taken a simple example of student form to understand all these three layers. It has information about a student like – Name, Address, Email, and Picture.
- User Interface Layer or Presentation Layer


Students Information

	ID	Name	Address	Email	Picture
▶✱					

Presentation Layer

```
private void DataGrid1_SelectedIndexChanged(object sender, System.EventArgs e)
{
// Object of the Property layer
clsStudent objproperty=new clsStudent();
// Object of the business layer
clsStudentInfo objbs=new clsStudentInfo();
// Object of the dataset in which we receive the data sent by the business layer
DataSet ds=new DataSet();
// here we are placing the value in the property using the object of the
//property layer
objproperty.id=int.Parse(DataGridl.SelectedItem.Cells[1].Text.ToString());

// In this following code we are calling a function from the business layer and
// passing the object of the property layer which will carry the ID till the database.
ds=objbs.GetAllStudentBsIDWise(objproperty);

// What ever the data has been returned by the above function into the dataset
//is being populate through the presentation laye.
txtId.Text=ds.Tables[0].Rows[0][0].ToString();
txtFname.Text=ds.Tables[0].Rows[0][1].ToString();
txtAddress.Text=ds.Tables[0].Rows[0][2].ToString();
txtemail.Text=ds.Tables[0].Rows[0][3].ToString();
```

Business Access Layer

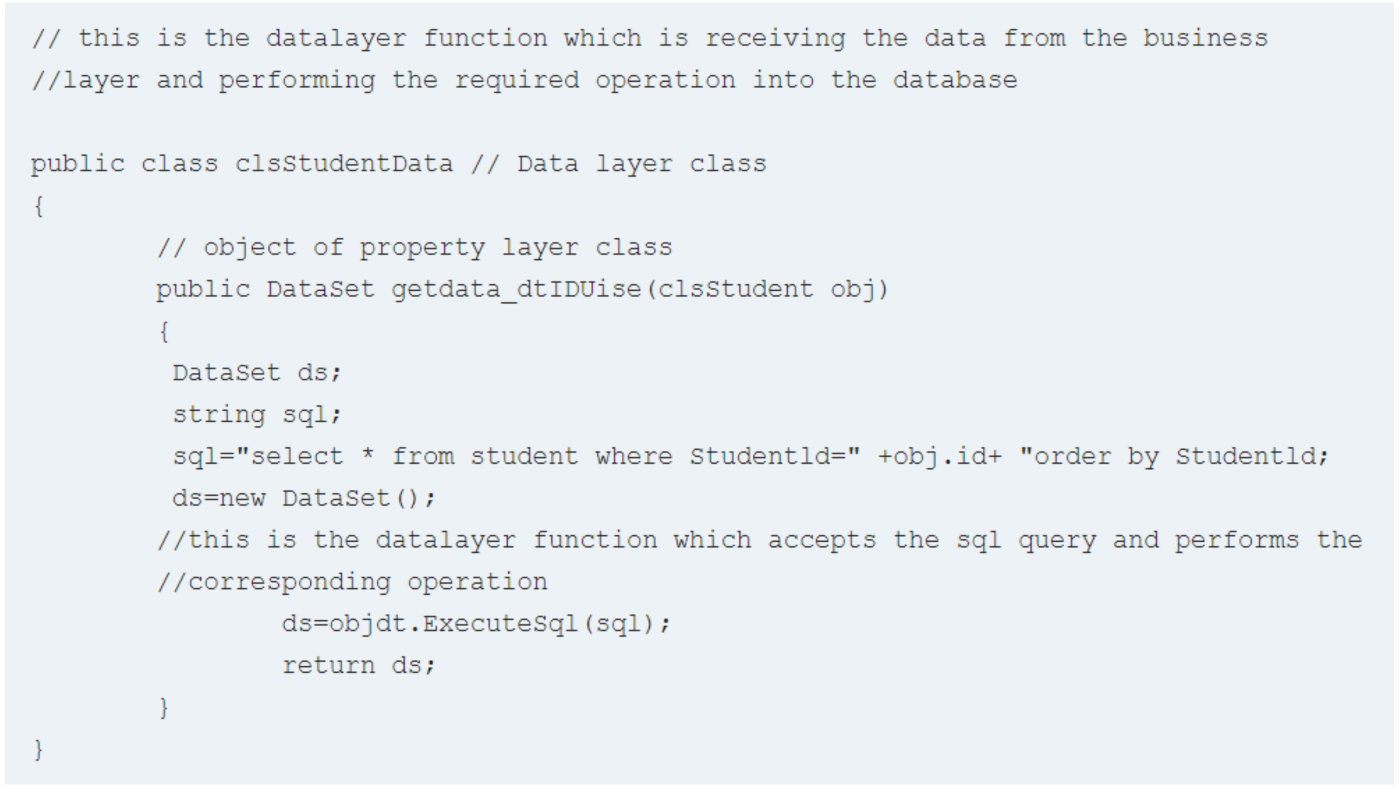
- This is the function of the business layer which accepts the data from the application layer and passes it to the data layer.
  - Business logic acts as an interface between Client layer and Data Access Layer
  - All business logic – like validation of data, calculations, data insertion/modification are written under business logic layer.
  - It makes communication faster and easier between the client and data layer
  - Defines a proper workflow activity that is necessary to complete a task.

Business Access Layer

```
// this is the function of the business layer which accepts the data from the
//application layer and passes it to the data layer.
public class clsStudentInfo
{
        public DataSet GetAllStudentBsIDWise(clsStudent obj)
        {
         DataSet ds=new DataSet();
         ds=objdt.getdata_dtIDWise(obj);// Calling of Data layer function
         return ds;
        }
}
```

Data Access Layer

- This is the data layer function, which receives the data from the business layer and performs the necessary operation into the database.



```
// this is the datalayer function which is receiving the data from the business
//layer and performing the required operation into the database

public class clsStudentData // Data layer class
{
        // object of property layer class
        public DataSet getdata_dtIDUise(clsStudent obj)
        {
         DataSet ds;
         string sql;
         sql="select * from student where Studentld=" +obj.id+ "order by Studentld;
         ds=new DataSet();
        //this is the datalayer function which accepts the sql query and performs the
        //corresponding operation
                ds=objdt.ExecuteSql(sql);
                return ds;
        }
}
```

2-Tier Architecture

- It is like Client-Server architecture, where communication takes place between client and server.
- In this type of software architecture, the presentation layer or user interface layer runs on the client side while dataset layer gets executed and stored on server side.
- There is no Business logic layer or immediate layer in between client and server.

Single Tier or 1-Tier Architecture

- It is the simplest one as it is equivalent to running the application on a personal computer. All of the required components for an application to run are on a single application or server.
- Presentation layer, Business logic layer, and data layer are all located on a single machine.

Two-Tier Architecture

- Also known as Client-Server Architecture.
- Divides the application into two main components: the client and the server.
- The client is responsible for the presentation layer and user interface, while the server handles the application processing and data storage.
- Suitable for small to medium-sized applications and often used in client-server database systems.

Two-Tier Architecture

- **Example:**
- A simple desktop application for managing a library. In this case:
- **Presentation Layer** (Client): The desktop application itself, where a librarian interacts with the user interface to perform tasks like searching for books, adding new entries, or checking in and checking out books.
- **Data Layer** (Database Server): The database, which stores information about the books, their availability, and details of library members. The desktop application directly interacts with the database to retrieve or update information.

One-Tier Architecture

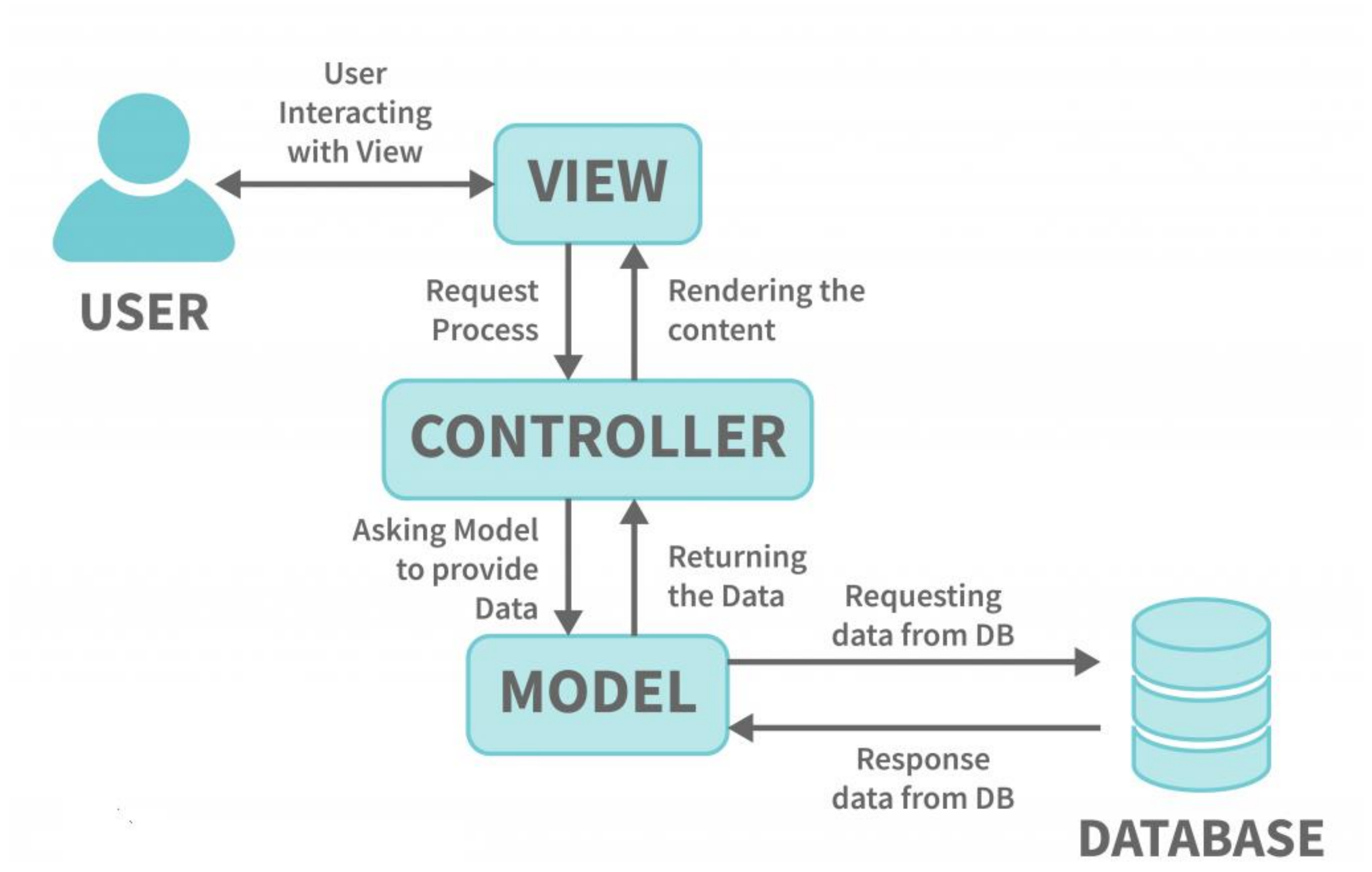
- Also known as Single-Tier Architecture.
- In this model, all the components (presentation, application processing, and data management) are housed in a single system or machine.
- Commonly found in small-scale applications where simplicity is prioritized over scalability.
- **Example**: Standalone Desktop Application
  - A simple calculator application installed on your computer could be an example of 1-tier architecture. All components, including the user interface, application logic, and data storage, are present on the same machine.

Model-view-controller

MVC Framework

- The Model-View-Controller (MVC) framework is an architectural pattern that separates an application into three main logical components
  - Model
  - View,
  - Controller.
- Hence the abbreviation MVC. Each architecture component is built to handle specific development aspects of an application.
- MVC separates the business logic and presentation layer from each other.
  - It was traditionally used for desktop graphical user interfaces (GUIs). Nowadays, MVC architecture in web technology has become popular for designing web applications as well as mobile apps.

MVC Architecture



Model

- Manages the application's data, business logic, and rules. It represents the core of the application's functionality.
  - It's like the brain, handling information and rules behind the scenes.
- **Example:** In a to-do list application, the Model would handle tasks, their status (completed or not), and any associated details.

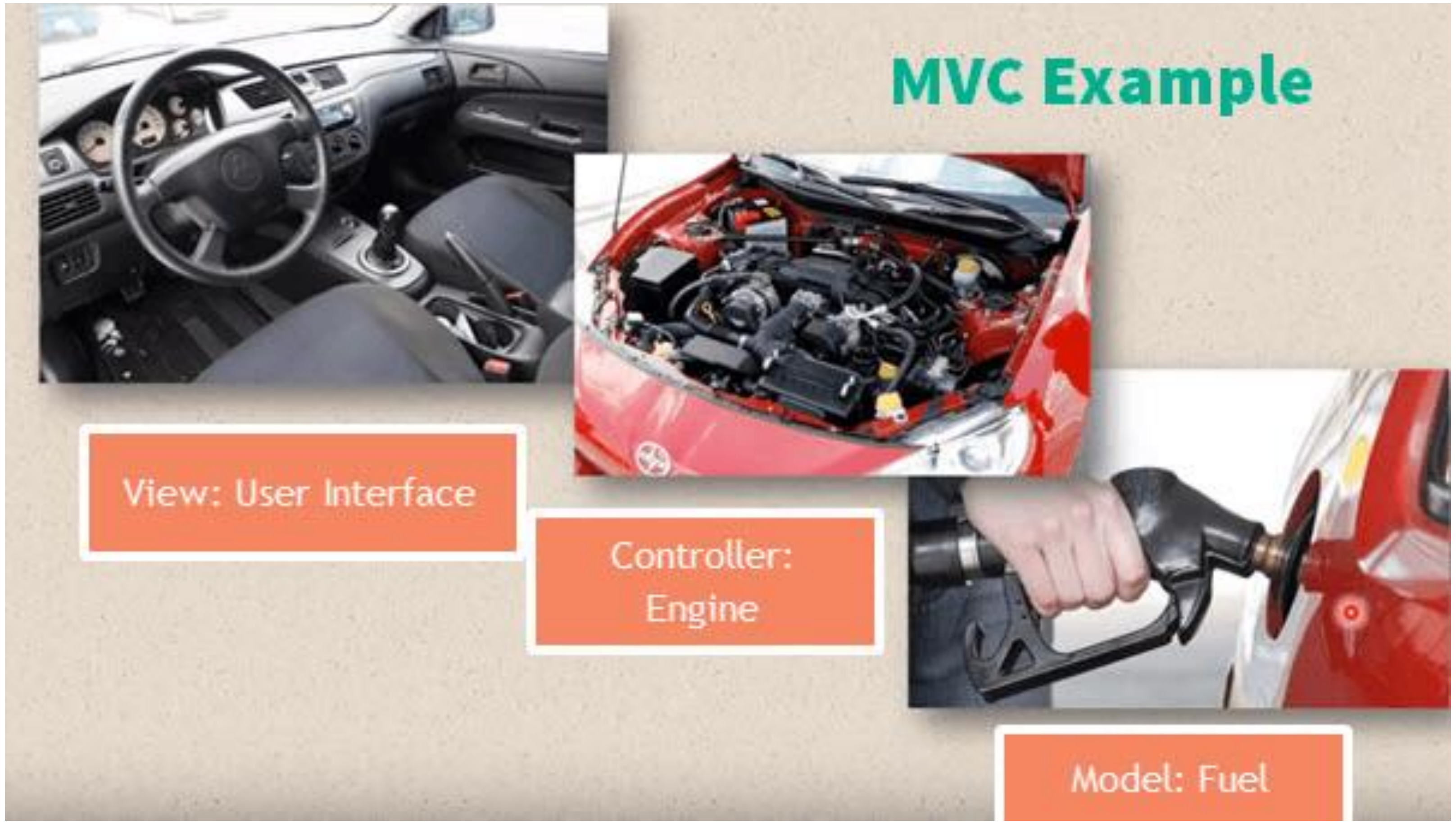
View

- Deals with the user interface and how information is presented to the user. It's responsible for displaying data from the Model and capturing user input.
- **Example**: In the same to-do list application, the View would be the actual visual representation of the tasks on the screen, allowing the user to see and interact with their to-do items.

Controller

- Acts as an intermediary between the Model and the View. It handles user input, updates the Model accordingly, and ensures that the View reflects any changes in the data.
- **Example**: In the to-do list application, the Controller would manage actions like adding a new task, marking a task as completed, or deleting a task. It translates user interactions into changes in the underlying data (Model) and updates the display (View) accordingly.

Example


MVC Example
View: User Interface
Controller: Engine
Model: Fuel

Popular MVC web frameworks

- Ruby on Rails
- Django
- CakePHP
- Yii
- CherryPy
- Spring MVC
- Catalyst
- Rails
- Zend Framework
- CodeIgniter
- Laravel
- Fuel PHP
- Symphony

3-tier Architecture vs. MVC Architecture

Parameter	3-Tier Architecture	MVC Architecture
Communication	This type of architecture pattern never communicates directly with the data layer.	All layers communicate directly using triangle topology.
Usage	3-tier: widely used in web applications where the client, data tiers, and middleware a run on physically separate platforms	Generally used on applications that run on a single graphical workstation.

It's worth noting that MVC can be implemented within the presentation layer of a three-tier architecture, where the MVC components reside in the presentation layer, and the other layers (application/business logic layer and data layer) follow the three-tier structure.

Questions?