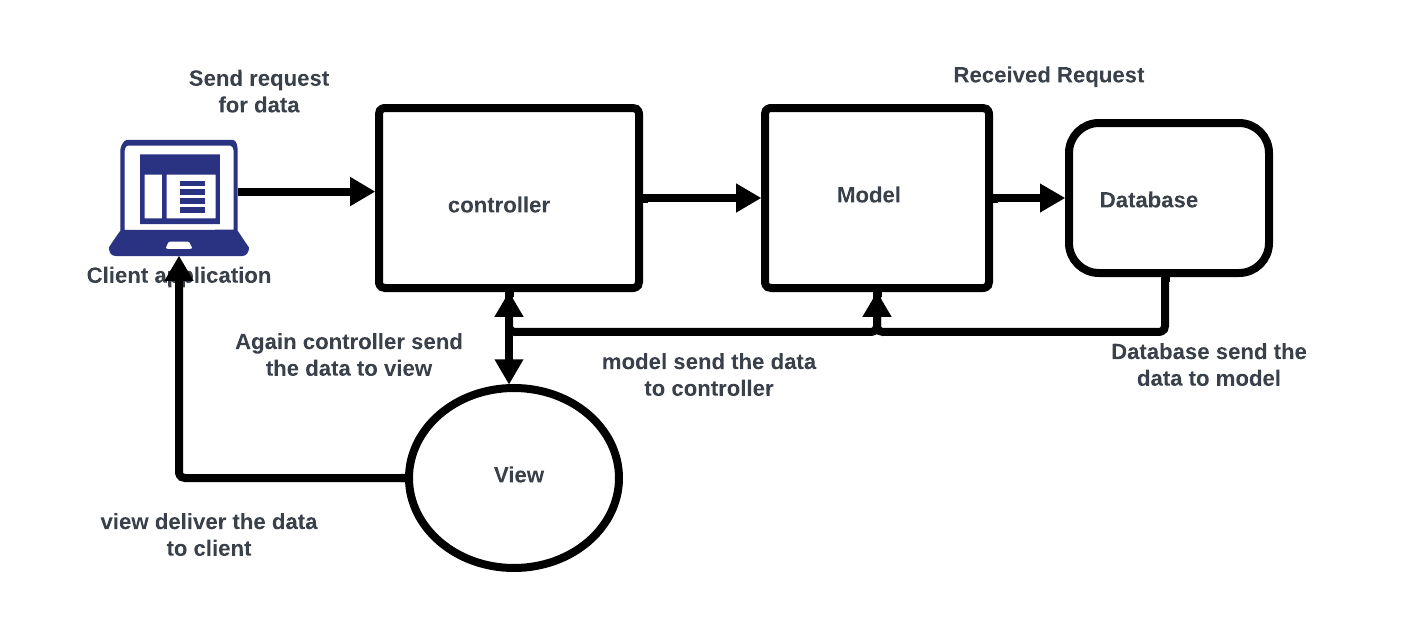
**1.Design Pattern Explanation - Prepare a one-page summary explaining the MVC (Model-View-Controller) design pattern and its two variants. Use diagrams to illustrate their structures and briefly discuss when each variant might be more appropriate to use than the others.**



1. Model

The Model component in the MVC (Model-View-Controller) design pattern represents the data and business logic of an application. It is responsible for managing the application’s data, processing business rules, and responding to requests for information from other components, such as the View and the Controller.

2. View

Displays the data from the Model to the user and sends user inputs to the Controller. It is passive and does not directly interact with the Model. Instead, it receives data from the Model and sends user inputs to the Controller for processing.

3. Controller

Controller acts as an intermediary between the Model and the View. It handles user input and updates the Model accordingly and updates the View to reflect changes in the Model. It contains application logic, such as input validation and data transformation.

**2.Principles in Practice - Draft a one-page scenario where you apply Microservices Architecture and Event-Driven Architecture to a hypothetical e-commerce platform. Outline how SOLID principles could enhance the design. Use bullet points to indicate how DRY and KISS principles can be observed in this context**.

**Microservices Architecture:**

* Inventory Service: Responsible for managing product inventory and availability.
* Order Service: Handles order processing, payment, and shipment tracking.
* User Service: Manages user authentication, profiles, and preferences.
* Recommendation Service: Suggests products based on user behaviour and preferences.
* Analytics Service: Gathers and analyses user activity for business insights.

**Event-Driven Architecture:**

* Inventory Update Events: Sent by the Inventory Service to notify the Order Service and Recommendation Service about changes in product availability.
* Order Placed Events: Triggered by the Order Service to update inventory, process payment, and notify users about order status.
* User Activity Events: Generated by the User Service to update user profiles and trigger recommendation updates.

**SOLID Principles:**

*Single Responsibility Principle (SRP):*

Each microservice has a single responsibility, ensuring clear and focused functionality.

*Open/Closed Principle (OCP):*

Microservices are designed to be open for extension but closed for modification.

*Liskov Substitution Principle (LSP):*

Microservices should be interchangeable without affecting the correctness of the system.

*Interface Segregation Principle (ISP):*

Microservices expose specific interfaces tailored to their functionalities.

*Dependency Inversion Principle (DIP):*

Microservices depend on abstractions rather than concrete implementations, facilitating flexibility and testability.

**DRY (Don't Repeat Yourself) Principle:**

Shared Libraries: Common functionalities like authentication and logging are implemented once and shared across microservices.

Code Generators: Use tools to generate boilerplate code for similar functionalities, reducing duplication.

**KISS (Keep It Simple, Stupid) Principle:**

*Minimalistic Design:* Microservices are kept lightweight, focusing on essential functionalities to avoid complexity.

*Clear Communication:* Emphasize clear communication protocols between services to minimize integration complexities.

**3. rends and Cloud Services Overview - Write a three-paragraph report covering: 1) the benefits of serverless architecture, 2) the concept of Progressive Web Apps (PWAs), and 3) the role of AI and Machine Learning in software architecture. Then, in one paragraph, describe the cloud computing service models (SaaS, PaaS, IaaS) and their use cases**

*Trends and Cloud Services Overview:*

Serverless architecture offers numerous benefits, revolutionizing the way applications are developed and deployed. By abstracting away server management, developers can focus solely on writing code, leading to faster time-to-market, reduced operational overhead, and cost optimization through pay-per-execution pricing models. Serverless also enables automatic scaling, allowing applications to handle varying workloads seamlessly without manual intervention, thus enhancing scalability and resilience.

*Progressive Web Apps (PWAs)*: represent the next evolution of web applications, combining the best features of web and mobile experiences. PWAs provide offline capabilities, push notifications, and fast performance, delivering a native app-like experience directly through the web browser. This approach enhances user engagement, reduces bounce rates, and increases conversions. With PWAs, businesses can reach a broader audience across various devices and platforms while simplifying development and maintenance efforts.

*AI and Machine Learning*: AI and Machine Learning are increasingly integral to modern software architecture, driving innovation and enhancing user experiences. These technologies empower applications to analyse vast amounts of data, extract valuable insights, and automate decision-making processes. From personalized recommendations and predictive analytics to natural language processing and image recognition, AI and Machine Learning enable intelligent, adaptive systems that continuously learn and improve over time, ultimately delivering more tailored and efficient solutions to users.

## Types of cloud services: IaaS, PaaS, serverless, and SaaS:-

Infrastructure as a Service (IaaS):-

IaaS contains the basic building blocks for cloud IT. It typically provides access to networking features, computers (virtual or on dedicated hardware), and data storage space. IaaS gives you the highest level of flexibility and management control over your IT resources. It is most similar to the existing IT resources with which many IT departments and developers are familiar.

Platform as a Service (PaaS):-

PaaS removes the need for you to manage underlying infrastructure (usually hardware and operating systems), and allows you to focus on the deployment and management of your applications. This helps you be more efficient as you don’t need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

Software as a Service (SaaS):-

SaaS provides you with a complete product that is run and managed by the service provider. In most cases, people referring to SaaS are referring to end-user applications (such as web-based email). With a SaaS offering, you don’t have to think about how the service is maintained or how the underlying infrastructure is managed. You only need to think about how you will use that particular software.