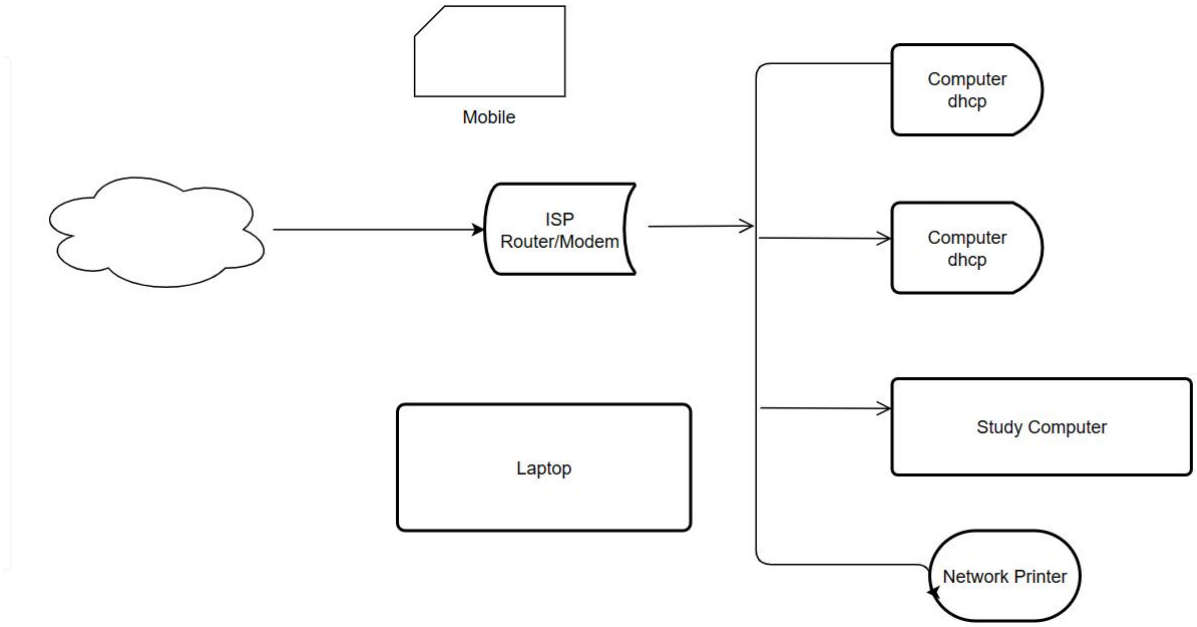
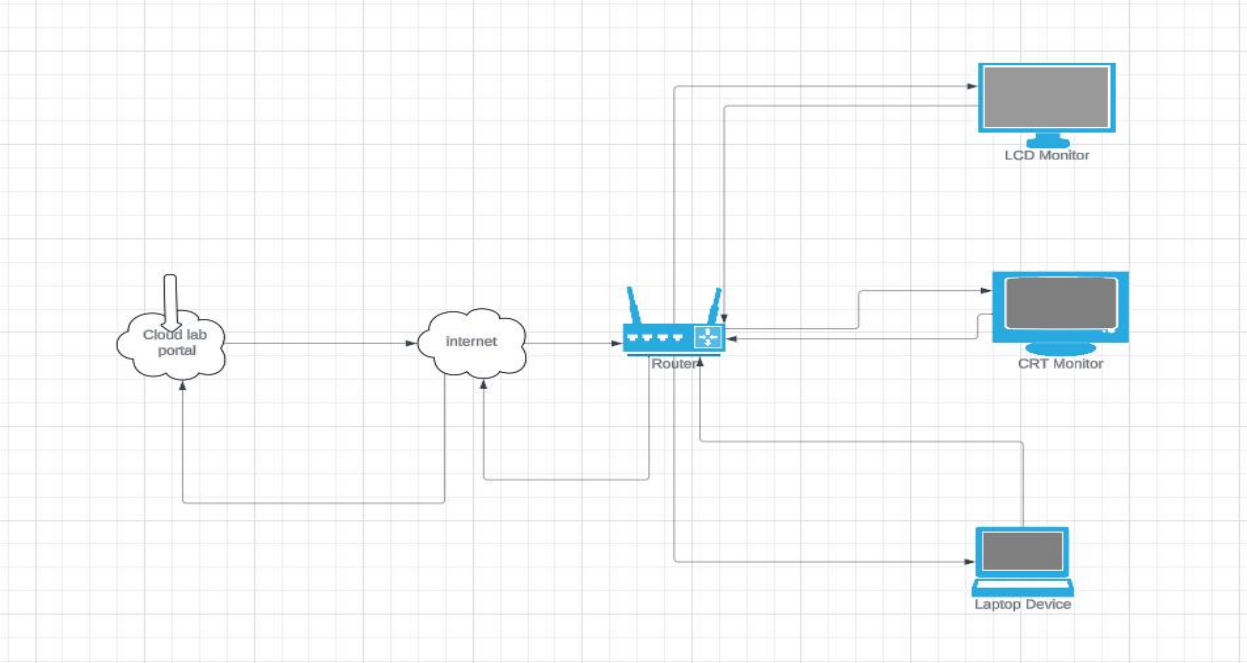
Home network Topology: -



Assignment 1:

Draw your Home Network Topology and explain how you are accessing the RPS Lab environment

Accessing the RPS lab environmnet:-



Components: -

1)Computers/laptop

2)Router

3)Internet

4)Data fetched/Cloud lab

Procedure: -

At first the computer/laptop are connected to the internet by connecting to the router.

Where the router id connected to the internet via cloud.

In the second stage our browser internet and search the link are the

source which is what we are required i.e. the RPS cloud lab.

The internet fetch for the query and display the required or the related solution.

By click in the suitable link we can see the required destination

All the above process are two ways directional.

Assignment 2:

Identify a real-world application for both parallel computing and networked systems. Explain how these technologies are used and why they are important in that context.

Solution:

- Parallel Computing: -

  Parallel computing refers to a programming paradigm that enables a computer to leverage multiple resources simultaneously to tackle computational tasks. This approach enables computers to handle multiple problems concurrently, thereby enhancing efficiency and performance. Often, large problems can be decomposed into smaller sub-problems, which can then be solved simultaneously, leading to faster overall computation.

- Benefits Parallel Programming: -

  Parallel computing offers the advantage of solving complex problems efficiently. By breaking down large tasks into smaller ones and processing them simultaneously, parallel programs achieve faster execution. Despite potentially requiring more hardware components, parallel systems are more efficient and produce results more quickly than serial processing. Additionally, they optimize resource utilization, leading to improved problem-solving capabilities

- Limitations Parallel Programming: -

  Parallel programming faces challenges in achieving parallel architecture, particularly in clusters requiring advanced cooling technologies. Managed algorithms are necessary, often handled within the parallel mechanism, but multi-core architectures consume high power. Creating low coupling and high cohesion in parallel computing systems is complex. Additionally, coding for parallelism demands expertise, typically managed by highly skilled programmers.

- Real-world application: -

Multithreading is a parallel computing method commonly utilized in parallel computer systems. The multiprocessing module in Python facilitates parallel programming. Intel processors, prevalent in modern high-performance computers, also employ parallel processing.

- Networked System: -

  A networked system connects devices via wired or wireless communication links, aiming to transfer information between them. It executes protocol stacks and switches data among nodes, which can be computers, printers, or other devices capable of sending and receiving data.

- Networking Systems offers various benefits: -

1)File sharing: Easily share data between users or access it remotely from connected devices.

2)Resource sharing: Share peripheral devices like printers and scanners, or software among multiple users, reducing costs.
3)Single internet connection: Cost-efficient and enhances system security when properly secured.
4)Increased storage capacity: Access files and multimedia stored remotely on network-attached devices.

- Real-world application: -

E-commerce enables online transactions for buying and selling goods. Communication networks support video calls, emails, and file transfers. Gaming networks facilitate interactive quiz games. Educational networks provide students access to information on the World Wide Web (WWW).

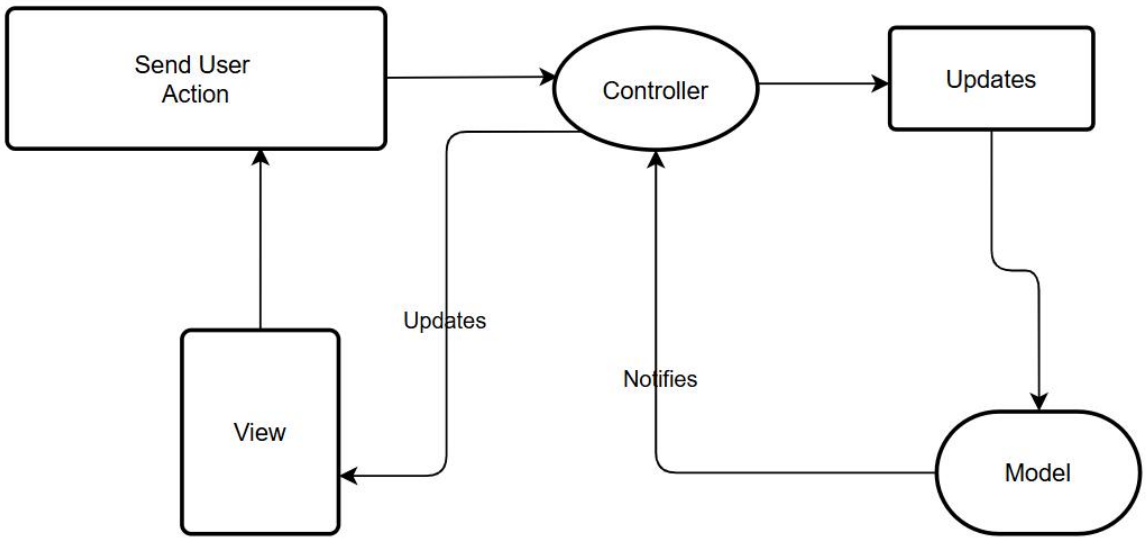
Assignment 3:

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

Summary:

The Model View Controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.

The MVC pattern separates the concerns of an application into three distinct components, each responsible for a specific aspect of the application's functionality.
This separation of concerns makes the application easier to maintain and extend, as changes to one component do not require changes to the other components.



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.

Assignment 4:

Principles in Practice - Draft a one-page scenario where you apply Micro services 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.

Micro services Architecture:-

Product Service: Manages product catalog, including creation, updates, and deletion.
Order Service: Handles orders, including checkout, payment processing, and order history.
User Service: Manages user accounts, authentication, and authorization.
Inventory Service: Tracks inventory levels, updates stock, and manages replenishment.
Notification Service: Sends notifications to users about order status, promotions, and updates.

Event-Driven Architecture:-

Event Bus: Acts as a central communication channel for services to publish and subscribe to events.

Events: Examples include "Product Created," "Order Placed," "Inventory-updated," etc.

Handlers: Services listen for relevant events and react accordingly. For instance, upon "Order-placed," the Inventory Service decrements stock, and the Notification Service sends a confirmation email.

Applying SOLID Principles:

Single Responsibility Principle (SRP): Each micro service has a single responsibility, ensuring that changes are localized and don't impact other services.

Open/Closed Principle (OCP): Services are open for extension but closed for modification. New features are added via new services rather than modifying existing ones.

Obelisk Substitution Principle (LSP): Services can be substituted with others implementing the same interface. For instance, different payment gateways can be integrated into the Order Service seamlessly.

Interface Segregation Principle (ISP): Service interfaces are tailored to specific needs, preventing clients from depending on methods they don't use.

Dependency Inversion Principle (DIP): High-level modules (e.g., Order Service) depend on abstractions (e.g., interfaces), not concrete implementations, facilitating flexibility and testing.