

# Module 1 - Overview of IT Industry

## (Practical Exercises)

### 1. Write a single 'Hello World' program in two different programming languages of your choice. Compare the structure and syntax.

#### i. Python

Python is known for its readability and simplicity.

- Program :-

```
print("Hello, World!")
```

#### ii. Java

Java is a more structured, object-oriented language.

- Program :-

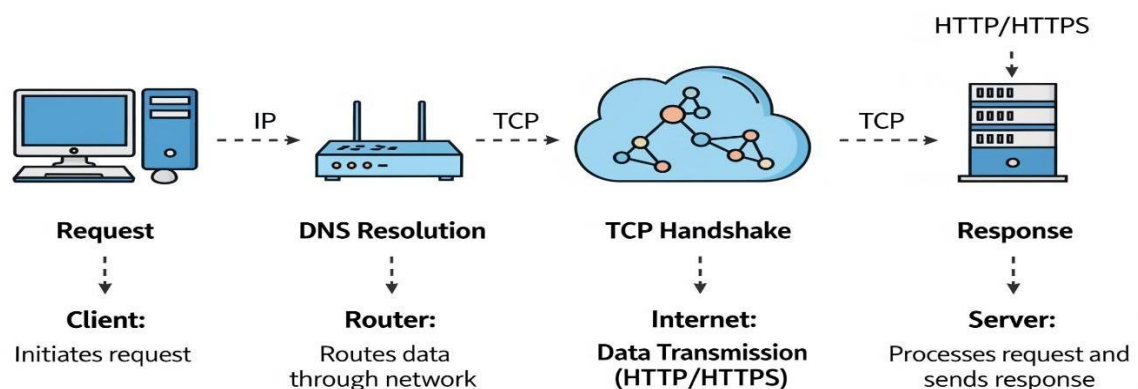
```
class HelloWorld {  
    public static void main(String[] args)  
    { System.out.println("Hello, World!");  
    }  
}
```

#### ❖ Comparison of Structure and Syntax

| Feature          | Python  | Java  |
|------------------|---|---|
| <b>Verbosity</b> | <b>Minimal.</b> A single, direct command is all that's needed. This makes it very easy for beginners to start with. | <b>Verbose.</b> The "Hello, World!" logic is wrapped inside a main method, which itself is inside a HelloWorld class. This is called "boilerplate" code |

| Feature   | Python  | Java  |
|-----------|---|---|
| Structure | <b>Flexible.</b> This example is a simple script. Code doesn't need to be inside a class. | <b>Strictly Object-Oriented.</b> All code in Java must reside within a class. The program's execution starts from the public static void main method. |

## 2. Research and create a diagram of how data is transnitted from a client to a server over the internet.

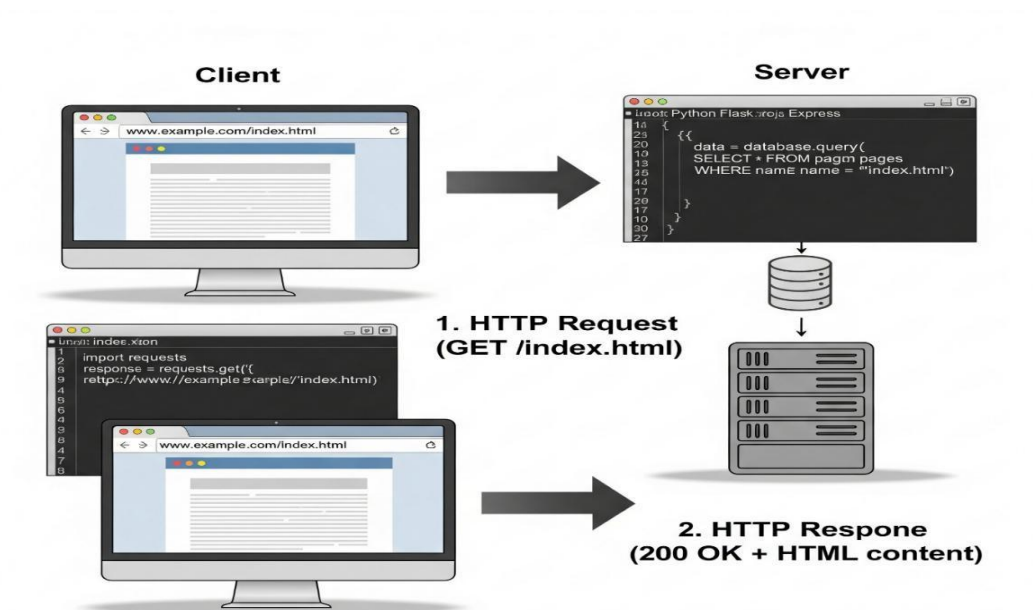


### ❖ How Data Travels Across the Internet :-

- The Request: It starts on your device, the client. When you want to go to a website, your device sends a request for that site's data.

- ii. Finding the Destination (DNS): Your request goes to a Domain Name System (DNS) server. The DNS is like the internet's phonebook; it looks up the website's name (like <https://www.google.com/search?q=google.com>) and finds its numerical IP address.
- iii. The Journey:
  - Once it has the IP address, your request is broken into small pieces called packets.
  - These packets travel from your local router onto the internet.
  - IP (Internet Protocol) addresses the packets to the correct server.
  - TCP (Transmission Control Protocol) makes sure all packets arrive in the right order.
- iv. The Server's Response:
  - The packets arrive at the destination server, a powerful computer holding the website's data.
  - The server processes your request and sends the data back to you in a new set of packets as a response. This whole process is usually handled by HTTP or the secure HTTPS protocol.

### 3. Design a simple HTTP client-server communication in any language.



- **Client Sends Request:** The client (a web browser or code) sends an HTTP Request to the server to ask for a specific webpage (e.g., index.html).
- **Server Processes Request:** The server receives the request, finds the necessary information (sometimes from a database), and prepares to send it back.
- **Server Sends Response:** The server sends back an HTTP Response, which includes a success code (200 OK) and the webpage's HTML content.
- **Client Displays Content:** The client's browser receives the HTML and renders it into the visual webpage you see on the screen.

#### **4. Research different types of internet connections (e.g., broadband, fiber, satellite) and list their pros and cons.**

##### **i. Dial-up (PSTN)**

###### **Pros:**

- Very low cost.
- Uses existing telephone lines.
- Easy to set up (historically).

###### **Cons:**

- Very slow (max ~56 kbps).
- Blocks the phone line during use.
- High latency and outdated.

##### **ii. DSL (Digital Subscriber Line)**

###### **Pros:**

- Always-on connection.
- Uses existing telephone wiring.
- Affordable for basic browsing.

###### **Cons:**

- Speed drops with distance from provider.

- Lower upload speeds than download.
- Less future-proof than fiber.

### iii. Cable Broadband

#### Pros:

- High download speeds.
- Widely available in cities and towns.
- Reliable for home and office use.

#### Cons:

- Shared connection — speed may drop during peak hours.
- Higher latency than fiber.
- Upload speed is usually slower than download

### iv. Fiber to the Home (FTTH)

#### Pros:

- Very high speeds (up to multiple Gbps).
- Equal upload and download speeds possible.
- Low latency and highly reliable.
- Future-proof for growing internet needs.

#### Cons:

- Not available everywhere.
- Installation can be costly.
- Slower rollout in rural areas.

## 5. Simulate HTTP and FTP requests using command line tools (e.g., curl).

```
Microsoft Windows [Version 10.0.22031.629]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN> curl http://example.com
<doctype html>
<html>
<head>
<title>Example Domain</title>
<meta charset="utf-8" />
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<style type="text/css">
body {
background-color: #f0f0f2;
margin: 0;
padding: 0;
font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
}
div {
width: 600px;
margin: 5em auto;
padding: 2em;
background-color: #fdfdff;
border-radius: 0.5em;
box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
}
a:link, a:visited {
color: #38488f;
text-decoration: none;
}
@media (max-width: 700px) {
div {
margin: 0 auto;
width: auto;
}
}
</style>
</head>
<body>
<div>
<h1>Example Domain</h1>
<p>This domain is for use in illustrative examples in documents. You may use this
domain in literature without prior coordination or asking for permission.</p>
<p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>

C:\Users\ADMIN> curl -v http://example.com
* Host example.com:80 was resolved.
* IPv6: 2600:1406:bc00:53::b81e:94ce, 2600:1406:bc00:53::b81e:94ce, 2600:1408:ec00:36::1736:7f24, 2600:1408:ec00:36::1736:7f31, 2600:1406:3a00:21::173e:2e65, 2600:1406:3a00:21::173e:2e66
* IPv4: 23.215.0.138, 96.7.128.175, 96.7.128.198, 23.192.228.80, 23.192.228.84, 23.215.0.136
* Trying [2600:1406:bc00:53::b81e:94ce]80...
* Trying 23.215.0.138:80...
* Connected to example.com (2600:1406:bc00:53::b81e:94ce) port 80
* using HTTP/1.1
> GET / HTTP/1.1
> Host: example.com
> User-Agent: curl/8.13.0
> Accept: */*
>
* Request completely sent off
* HTTP/1.1 200 OK
< Content-Type: text/html
< ETag: "94238dfc8892e3d9c0dac8ef93371a07:1736799088.121134"
< Last-Modified: Mon, 11 Jun 2025 20:11:20 GMT
< Cache-Control: max-age=1000
< Date: Fri, 12 Jun 2025 07:45:50 GMT
< Content-Length: 1256
< Connection: keep-alive
<
<doctype html>
<
<doctype html>
<html>
<head>
<title>Example Domain</title>
<meta charset="utf-8" />
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<style type="text/css">
body {
background-color: #f0f0f2;
margin: 0;
padding: 0;
font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
}
div {
width: 600px;
margin: 5em auto;
padding: 2em;
background-color: #fdfdff;
border-radius: 0.5em;
box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
}
a:link, a:visited {
color: #38488f;
text-decoration: none;
}
@media (max-width: 700px) {
div {
margin: 0 auto;
width: auto;
}
}
</style>
</head>
<body>
<div>
<h1>Example Domain</h1>
<p>This domain is for use in illustrative examples in documents. You may use this
domain in literature without prior coordination or asking for permission.</p>
<p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
* Connection #0 to host example.com left intact

C:\Users\ADMIN> curl -O http://example.com/file.zip
% Total % Received % Xferd Average Speed Time Time Current
Dload Upload Total Spent Left Speed
100 1256 100 1256 0 0 1233 0 0:00:01 0:00:01 --:--:-- 1235

C:\Users\ADMIN>
```

## **6. Identify and explain three common application security vulnerabilities. Suggest possible solutions.**

### **SQL Injection (SQLi)**

This vulnerability allows attackers to inject malicious SQL code into input fields. This can bypass authentication and expose, modify, or delete data from the database.

**Solution:** Use prepared statements or parameterized queries to separate SQL code from user input. Always validate and sanitize all user-provided data.

### **Cross-Site Scripting (XSS)**

XSS involves injecting malicious scripts (e.g., JavaScript) into web pages. When other users view the page, their browsers execute the script, which can lead to session hijacking, data theft, or website defacement.

**Solution:** Output encode all user data before displaying it. Implement a Content Security Policy (CSP) to restrict which scripts are allowed to run on your site.

### **Broken Authentication**

This refers to weak or improperly configured authentication and session management systems. Flaws can allow attackers to compromise passwords, impersonate users, or gain unauthorized access.

**Solution:** Enforce strong password policies and use multi-factor authentication (MFA). Implement secure session management by generating random, unpredictable session IDs and invalidating them after logout. Protect against brute-force attacks by limiting failed login attempts.

## **7. Identify and classify 5 applications you use daily as either system software or application software.**

### **System Software:-**

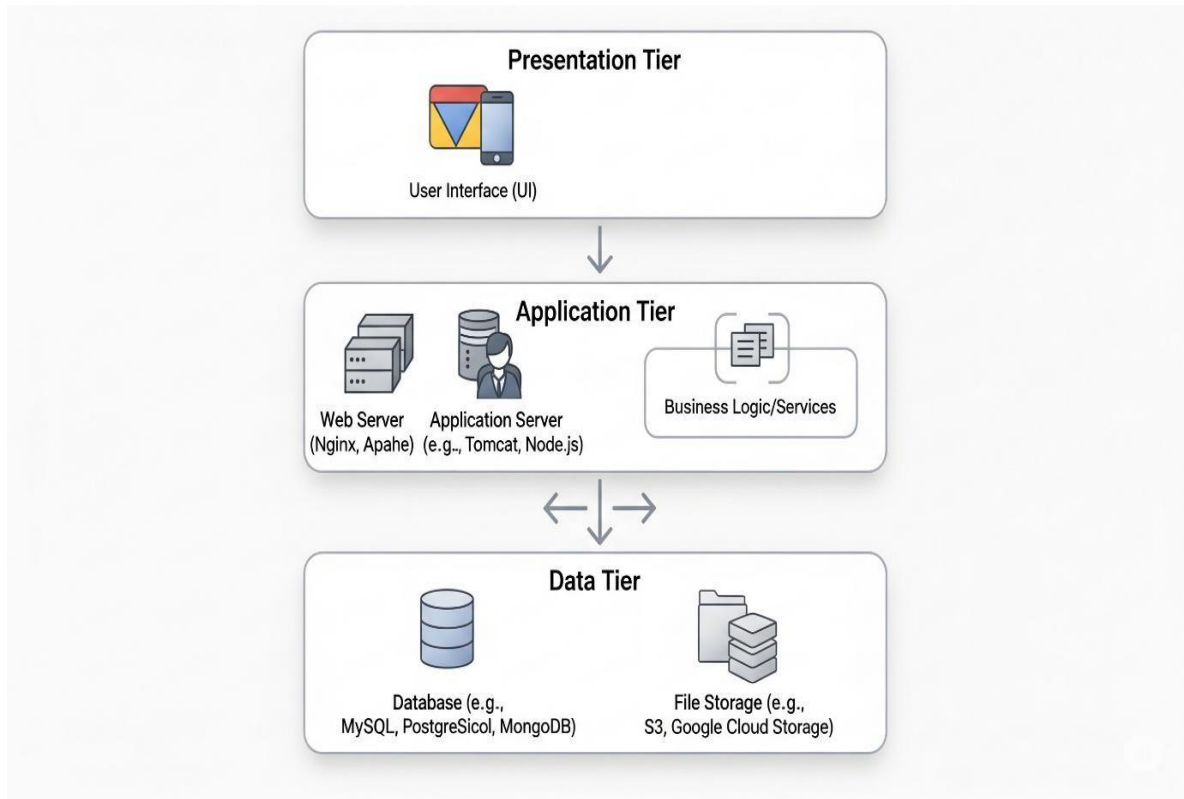
- i. Android OS: This is an operating system that runs on many smartphones and tablets. It manages all the device's resources, including the screen, processor, and memory, and allows you to install and run applications.
- ii. Microsoft Windows: A popular operating system for desktop and laptop computers. It provides the core environment for everything from managing files to running games and productivity software.

### **Application Software:-**

- i. Google Chrome: A web browser used for navigating the internet. Its sole purpose is to allow a user to view websites and interact with web-based content.
- ii. Microsoft Word: A word processor used for creating, editing, and formatting documents. It's a prime example of a productivity application that helps a user complete a specific task.
- iii. Spotify: A media player application for streaming music and podcasts. It's designed to provide a specific form of entertainment to the end-user.



**8. Design a basic three-tier software architecture diagram for a web application.**



## **9. Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system.**

### **❖ Case Study: Functionality of Presentation, Business Logic, and Data Access Layers**

#### **1. Introduction**

Three-tier architecture divides software into:

1. Presentation Layer (PL) – User interface and interaction.
2. Business Logic Layer (BLL) – Core rules and processing.
3. Data Access Layer (DAL) – Database operations.

This case study uses an Online Bookstore System to explain each layer.

#### **2. Layer Functions**

##### **A. Presentation Layer (PL)**

- Role: Displays information and collects user inputs.
- Functions: Show book lists, search/filter books, display cart, validate forms.
- Example: User searches "Science Fiction," PL sends request to BLL and displays results.

##### **B. Business Logic Layer (BLL)**

- Role: Processes requests, enforces rules, coordinates data flow.
- Functions: Validate stock, apply discounts, handle order logic.
- Example: On "Place Order," BLL checks stock, applies discounts, and sends order to DAL.

##### **C. Data Access Layer (DAL)**

- Role: Manages database operations securely.
- Functions: CRUD operations, parameterized queries, data mapping.
- Example: Fetch all books in a category using SQL and return to BLL.

### 3. Data Flow Example: Order Placenment

1. PL: Sends order request to BLL.
2. BLL: Validates and processes order → Sends to DAL.
3. DAL: Saves to database → Confirms to BLL → PL displays success message.

### 4. Benefits

- Scalability
- Easier maintenance
- Improved security

### 5. Conclusion

Separating PL, BLL, and DAL ensures a structured, maintainable, and scalable system. In the Online Bookstore, each layer handles specific tasks, creating a robust and efficient application.

## 10. Explore different types of software environnents (developnent, testing, production). Set up a basic environnent in a virtual nachine.

### ❖ Types of Software Environnent :-

#### i. Development Environment

- Used by programmers to write and debug code.
- Contains IDEs, compilers, version control tools, and dependencies.

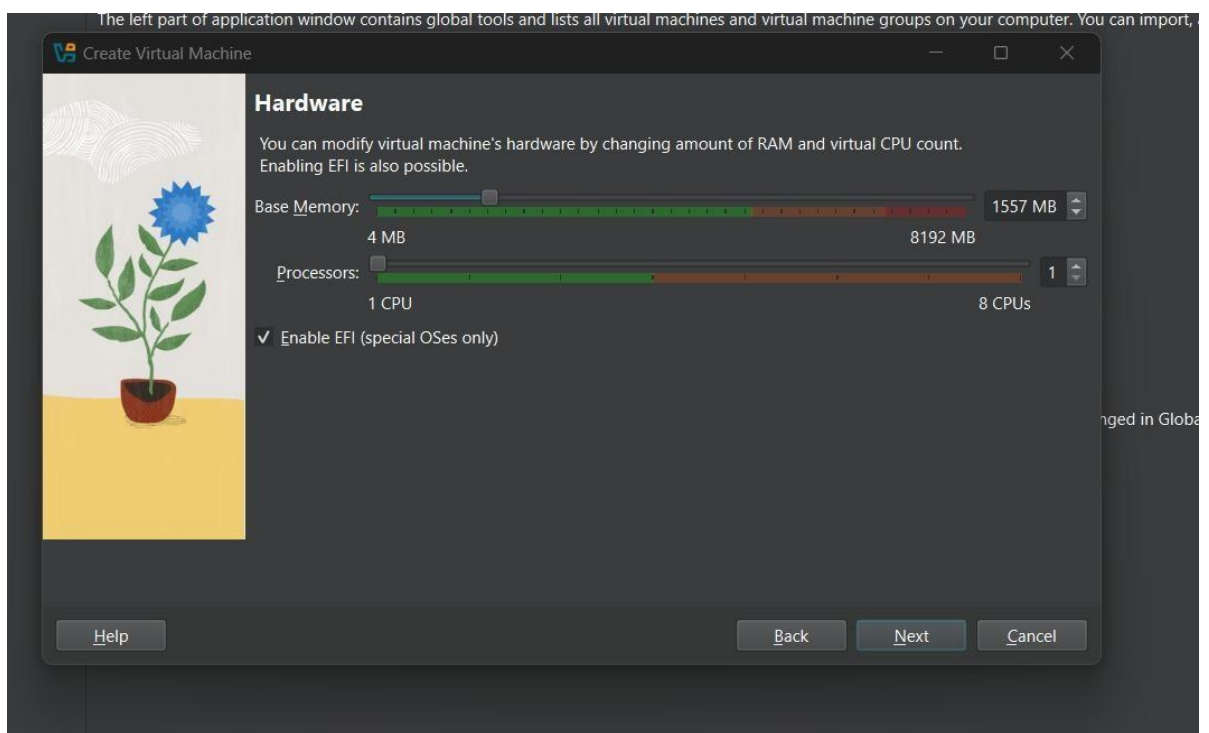
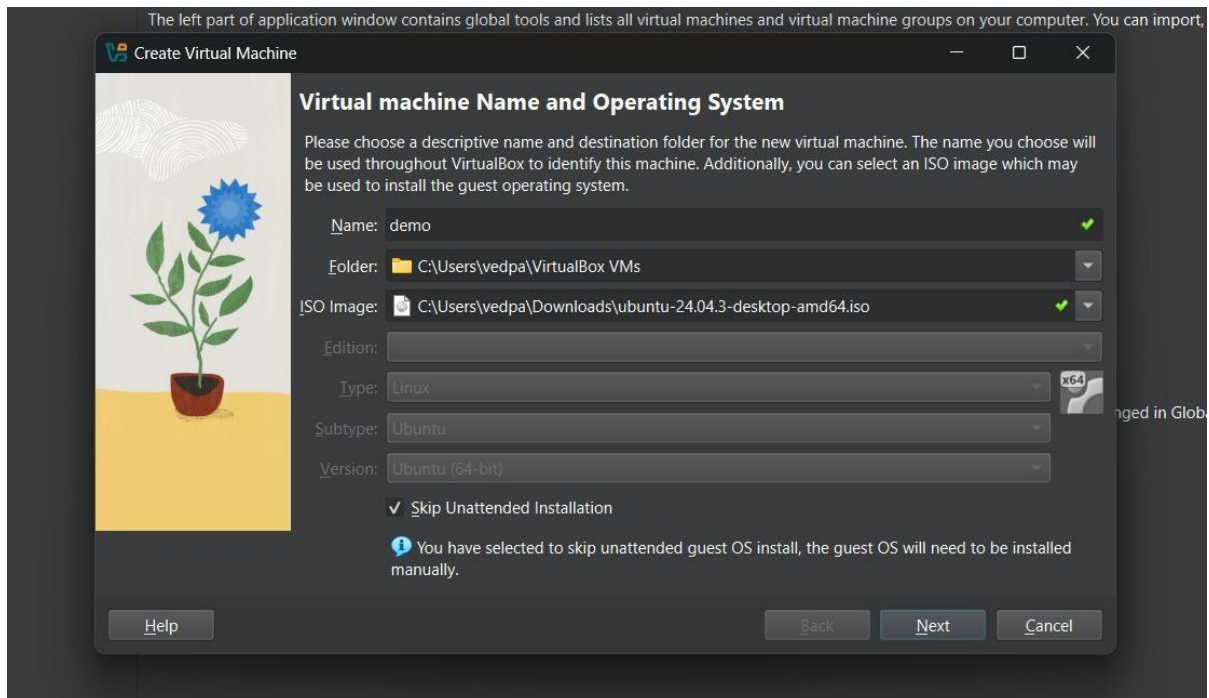
#### ii. Testing Environment

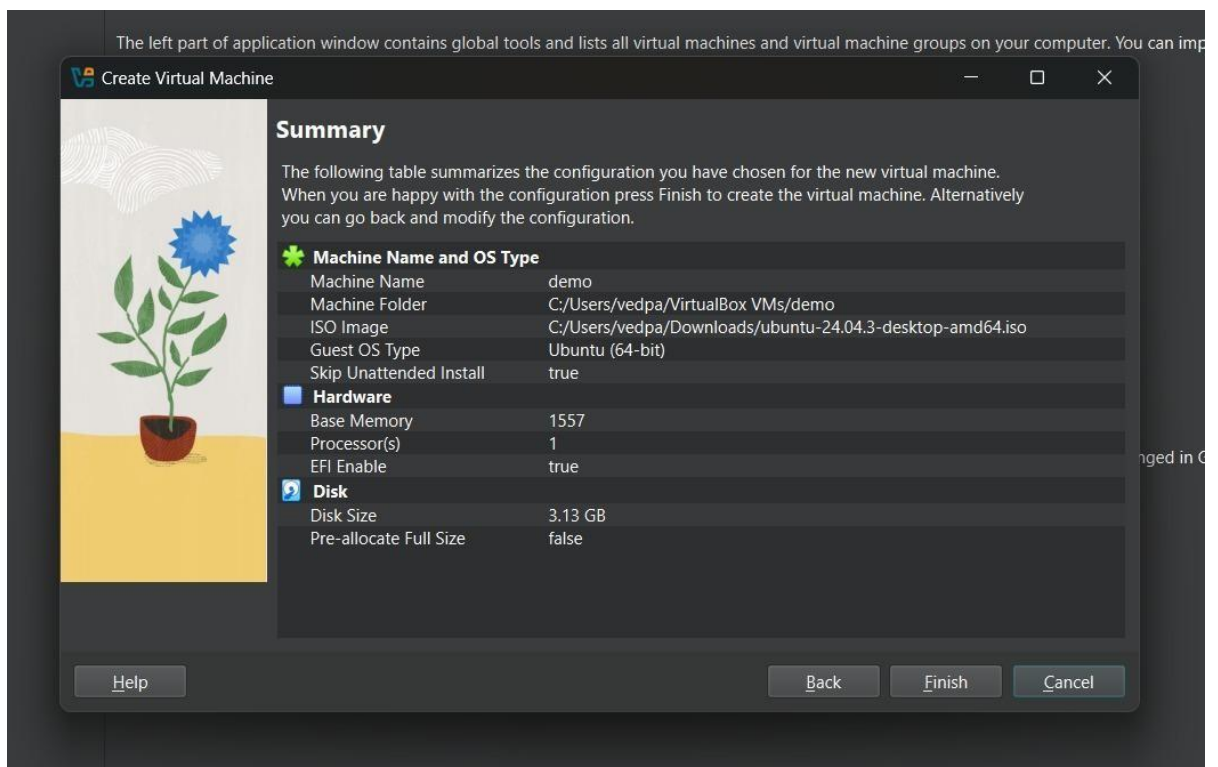
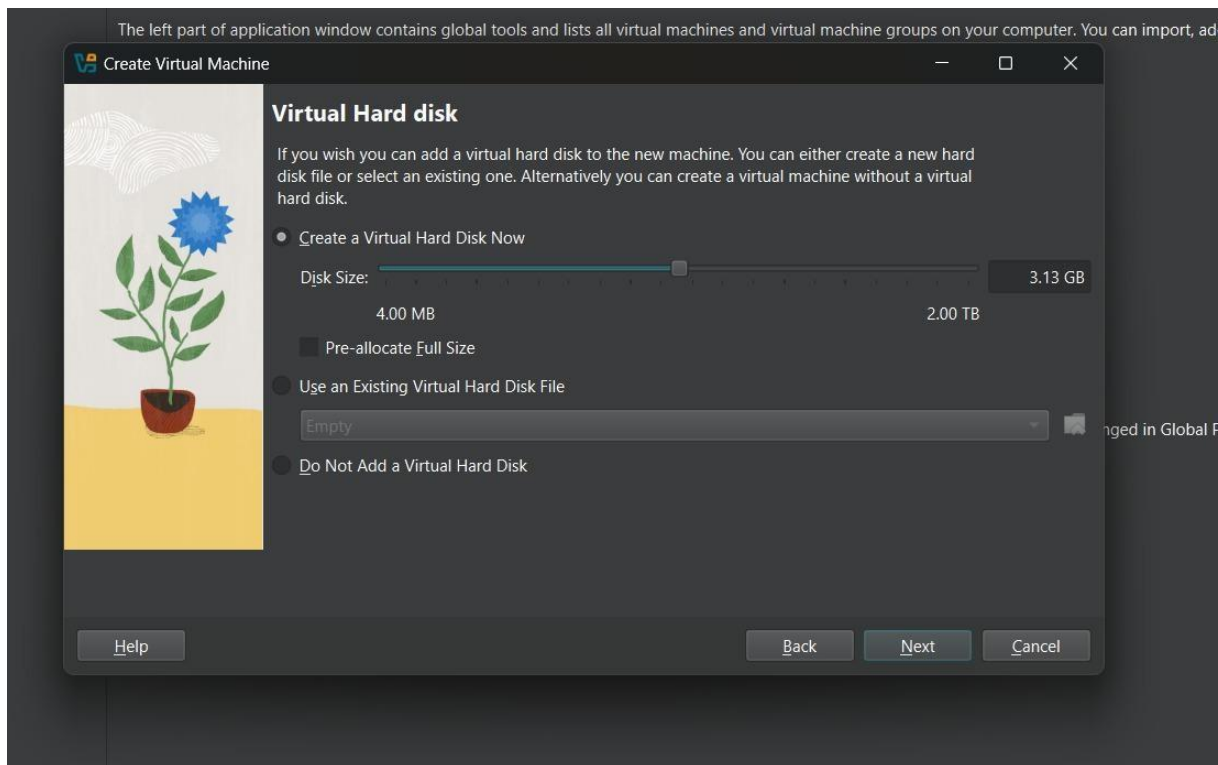
- Used for quality assurance and bug checking.
- Simulates production with test data.

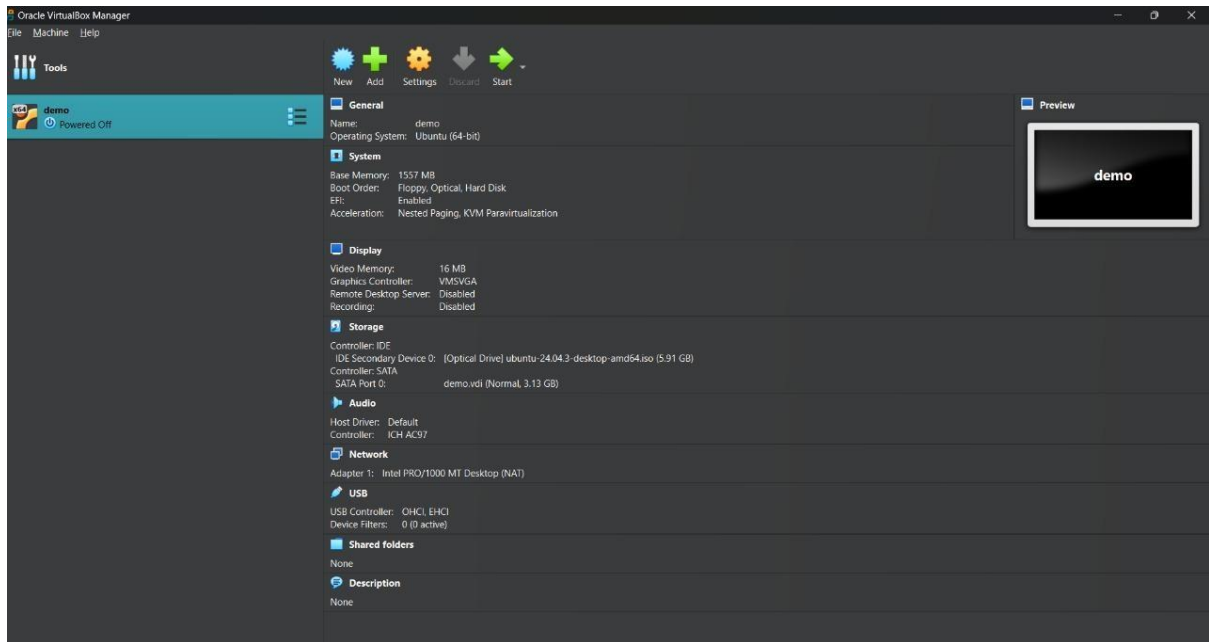
#### iii. Production Environment

- The live system where real users interact.
- Requires high stability, performance, and security.

## ❖ Setting up a Basic Environment in a Virtual Machine :-







## 11. Write and upload your first source code file to Github.

### ➤ Creating a New Repository :-

The screenshot shows the GitHub 'Create a new repository' page. At the top, there's a header with the GitHub logo, a search bar, and navigation icons. The main content area is titled 'Create a new repository' with a 'Preview' button and a link to 'Switch back to classic experience'. Below this, a note states: 'Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository.](#) Required fields are marked with an asterisk (\*)'. The form is divided into two sections: '1 General' and '2 Configuration'. In the 'General' section, the 'Owner' is set to 'RajPandya19' and the 'Repository name' is 'My\_First', with a green checkmark indicating 'My\_First is available.'. A description field is empty, with a character count of '0 / 350 characters'. The 'Configuration' section includes 'Choose visibility' set to 'Public', 'Add README' with a toggle switch turned 'On', 'Add .gitignore' set to 'No .gitignore', and 'Add license' set to 'No license'. A green 'Create repository' button is at the bottom right.

## ➤ Adding Source Code File & new file :-

The screenshot displays the GitHub interface for a repository named 'My\_First' by user 'RajPandya19'. The left sidebar shows the 'Files' section with a search bar and a list of files: 'main', 'Hello.c', and 'README.md'. The 'Hello.c' file is selected. The main content area shows the file's commit history, with the latest commit by 'RajPandya19' titled 'Create Hello.c' at 9745d36, committed 2 minutes ago. Below the commit history, the 'Code' tab is active, showing the source code of 'Hello.c'. The code is a simple C program that includes `<stdio.h>`, defines `main()`, and prints 'Hi Github!'. The code is 5 lines long, 5 loc, and 63 bytes.

My\_First / Hello.c

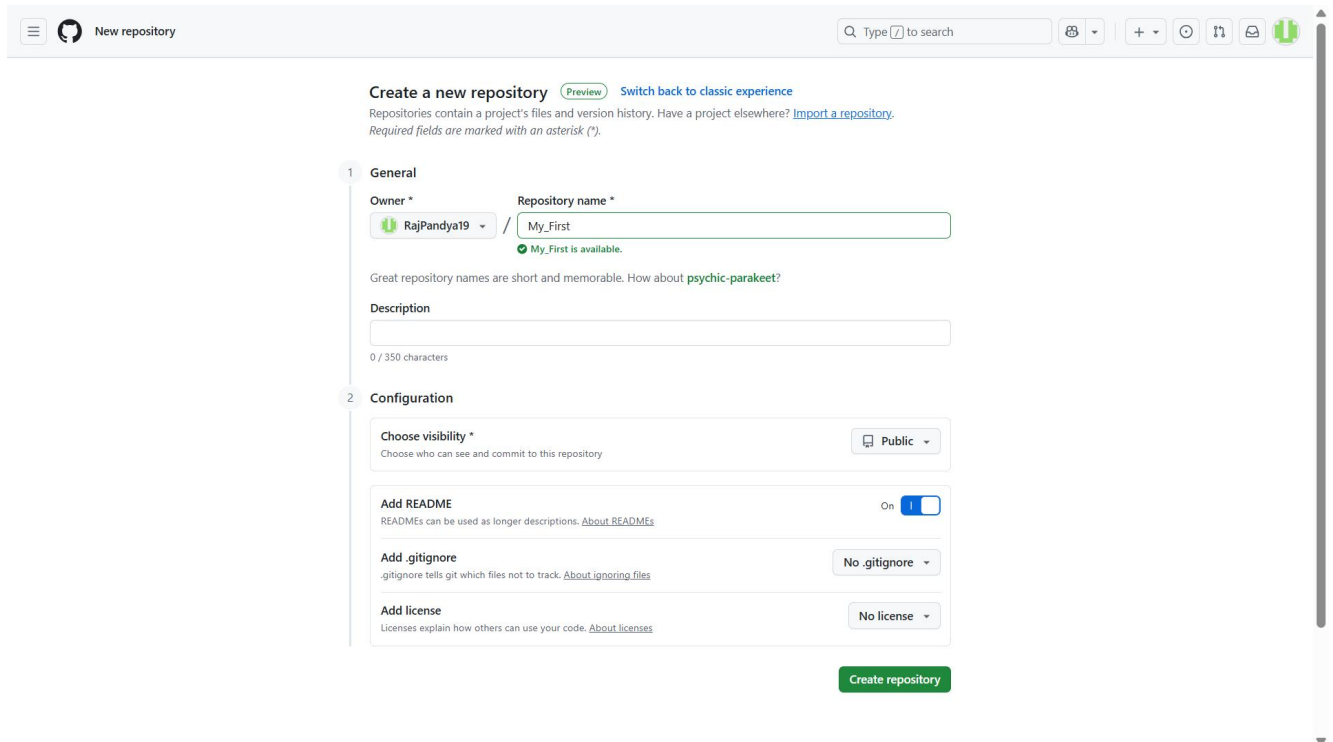
RajPandya19 Create Hello.c 9745d36 · 2 minutes ago History

Code Blame 5 lines (5 loc) · 63 Bytes

```
1 #include<stdio.h>
2 void main()
3 {
4     printf("Hi Github!\n");
5 }
```

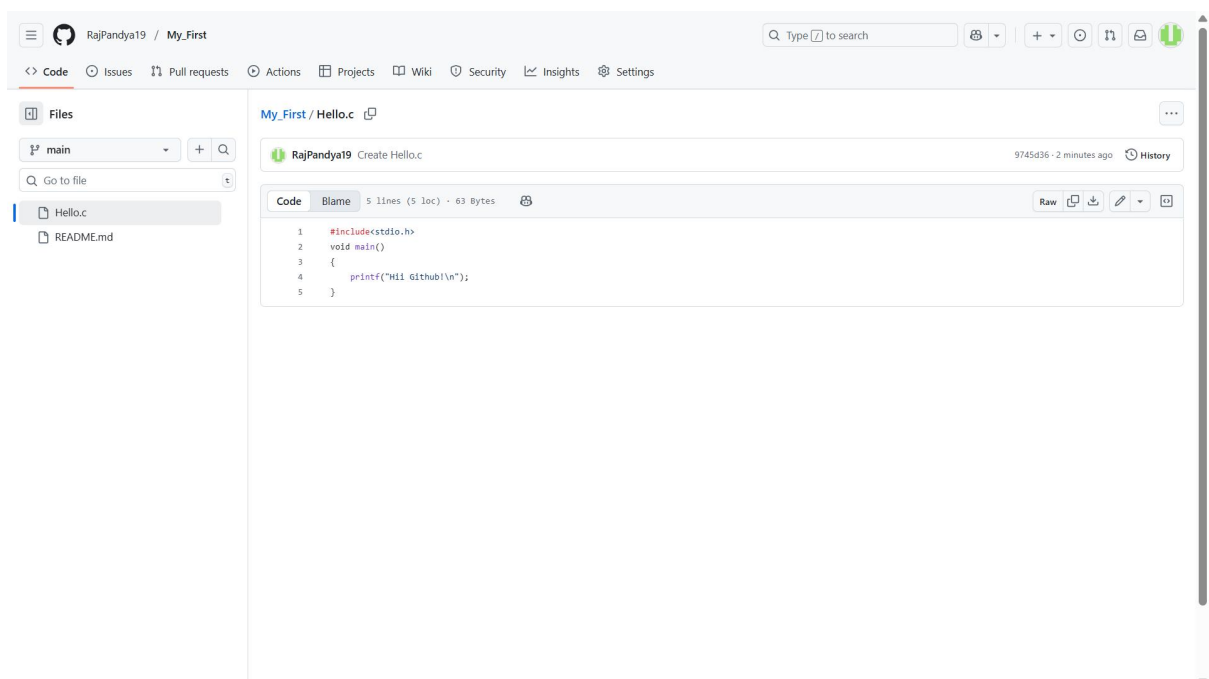
## 12. Create a Github repository and document how to commit and push code changes.

### ➤ Creating a New Repository :-



The screenshot shows the GitHub 'Create a new repository' page. At the top, there's a header with the GitHub logo, a search bar, and navigation icons. The main heading is 'Create a new repository' with a 'Preview' link and a 'Switch back to classic experience' link. Below this, a note states: 'Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository.](#) Required fields are marked with an asterisk (\*).' The form is divided into two sections: '1 General' and '2 Configuration'. In the 'General' section, the 'Owner' is 'RajPandya19' and the 'Repository name' is 'My\_First', with a green checkmark indicating 'My\_First is available.' There's a 'Description' field with a character count '0 / 350 characters'. The 'Configuration' section includes 'Choose visibility' set to 'Public', 'Add README' with a toggle 'On', 'Add .gitignore' set to 'No .gitignore', and 'Add license' set to 'No license'. A green 'Create repository' button is at the bottom right.

### ➤ Adding Source Code File :-

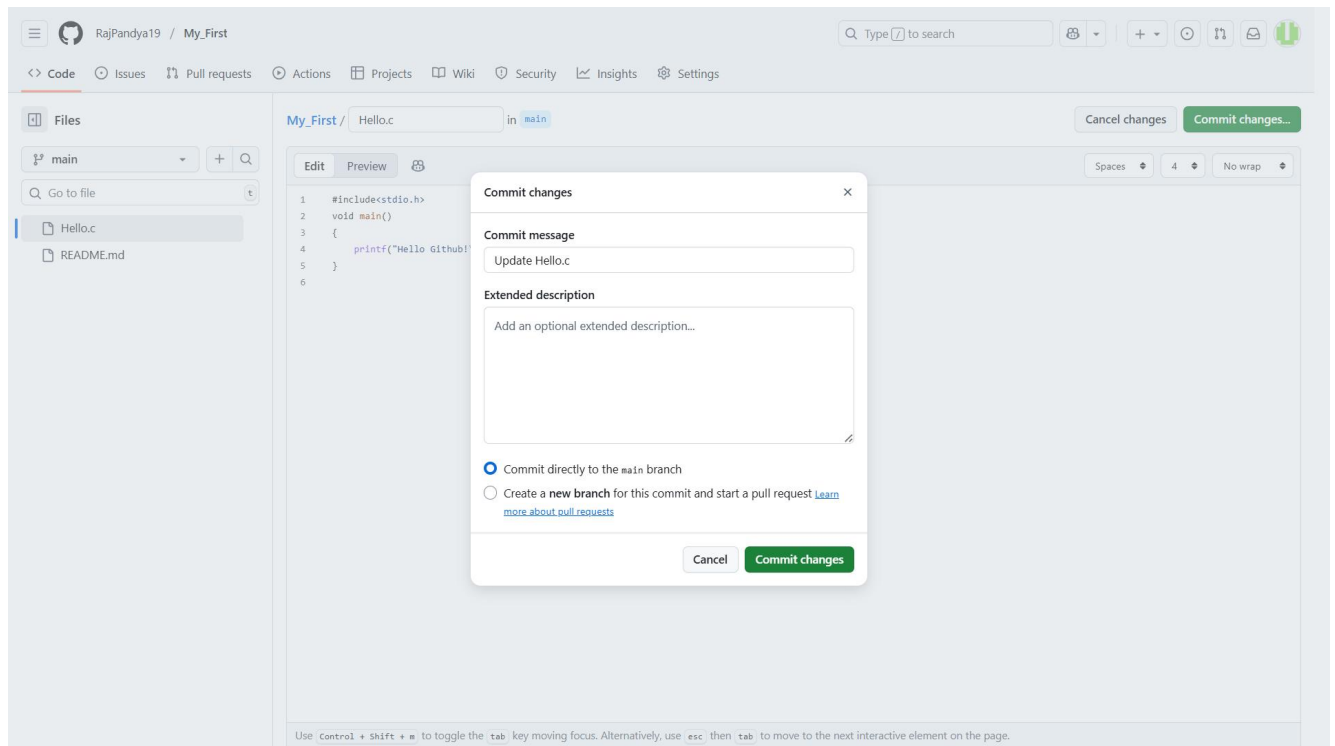


The screenshot shows the GitHub repository 'My\_First' by user 'RajPandya19'. The left sidebar shows the 'Files' section with 'main' branch selected and a search bar. Below the search bar, the file 'Hello.c' is listed. The main area shows the 'Hello.c' file content, which is a C program that prints 'Hello Github!'. The file is 5 lines long, 5 lines of code, and 63 bytes. The code is as follows:

```
1 #include<stdio.h>
2 void main()
3 {
4     printf("Hello Github!\n");
5 }
```

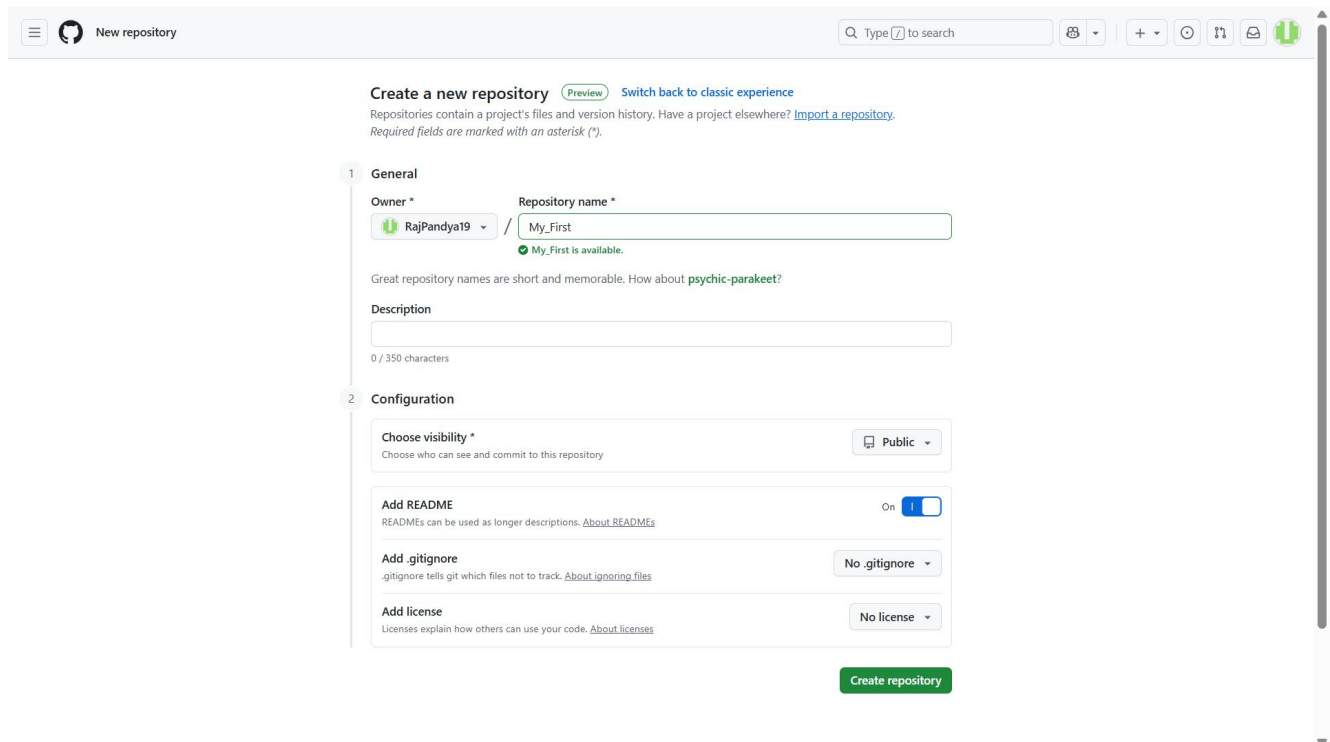


## ➤ Conniting new file :-



## 13. Create a student account on Github and collaborate on a small project with a classmate.

### ❖ Github Repository (Project File) :-



The screenshot shows the 'Create a new repository' page on GitHub. The page has a header with the GitHub logo, 'New repository', a search bar, and navigation icons. The main content area is divided into two sections: 'General' and 'Configuration'.

**General Section:**

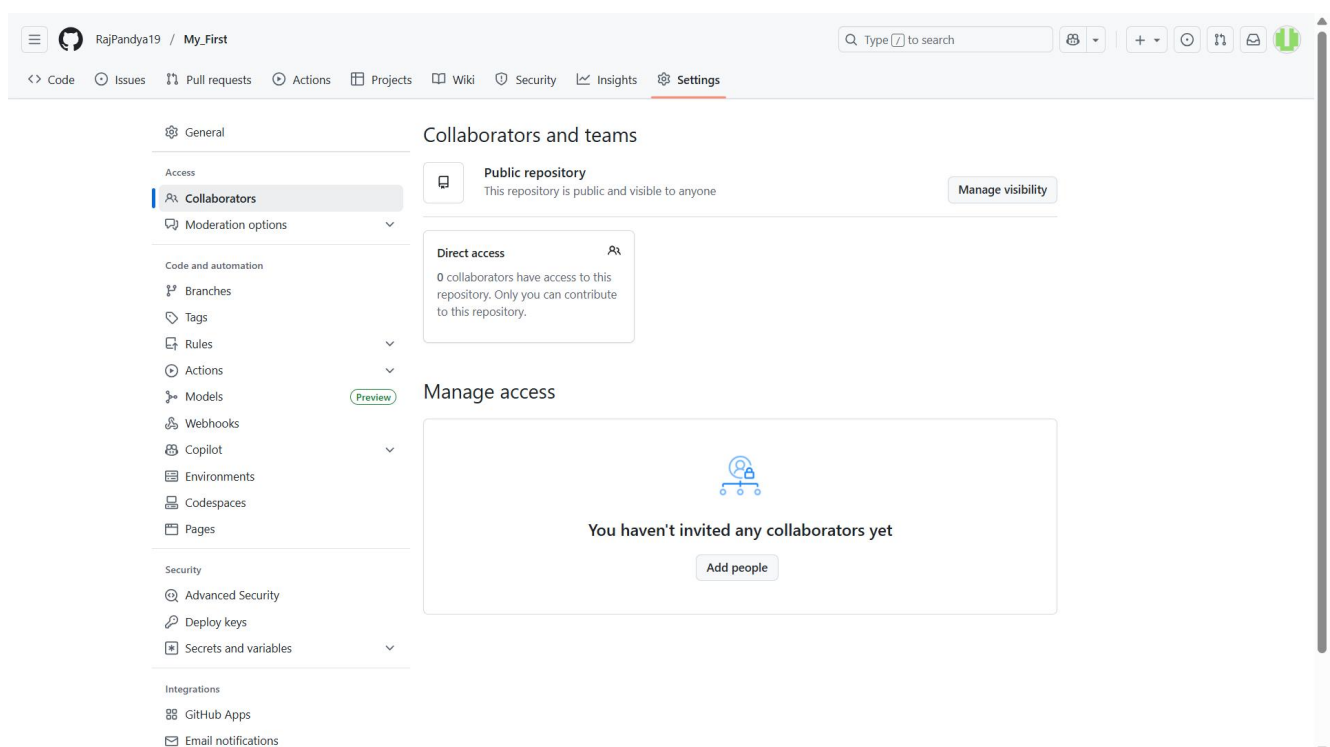
- Owner \***: A dropdown menu showing 'RajPandya19'.
- Repository name \***: A text input field containing 'My\_First'. Below it, a green checkmark indicates 'My\_First is available.'
- Description**: A text input field with a character count '0 / 350 characters'.

**Configuration Section:**

- Choose visibility \***: A dropdown menu set to 'Public'.
- Add README**: A toggle switch set to 'On'.
- Add .gitignore**: A dropdown menu set to 'No .gitignore'.
- Add license**: A dropdown menu set to 'No license'.

A green 'Create repository' button is located at the bottom right of the configuration section.

### ❖ Collaborator's work (Classmate) :-



The screenshot shows the 'Collaborators and teams' page for a repository named 'My\_First' owned by 'RajPandya19'. The page has a header with the repository name, a search bar, and navigation icons. The left sidebar contains a list of settings categories: General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, Models, Webhooks, Copilot, Environments, Codespaces, Pages, Security, Advanced Security, Deploy keys, Secrets and variables, Integrations, GitHub Apps, and Email notifications.

**Collaborators and teams Section:**

- Public repository**: A card indicating 'This repository is public and visible to anyone' with a 'Manage visibility' button.
- Direct access**: A card showing '0 collaborators have access to this repository. Only you can contribute to this repository.'

**Manage access Section:**

- A large card with a blue icon of a person and a plus sign, stating 'You haven't invited any collaborators yet' and an 'Add people' button.

RajPandya19 / My\_First

Q Type [Z] to search

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Models

Webhooks

Copilot

Environments

Codespaces

Pages

Security

Advanced Security

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

Collaborators and teams

Public repository

This repository is public and visible to anyone

Manage visibility

Direct access

2 entities have access to this repository, 0 collaborators, 2 invitations.

Manage access

Add people

Select all

Type

Find a collaborator...

Rudray Pandya

Awaiting RudrayPandya's response

Pending Invite

VedPandya31

Awaiting VedPandya31's response

Pending Invite

< Previous

Next >

**14. Create a list of software you use regularly and classify then into the following categories: system, application, and utility software.**

### **System Software**

- i. Windows 10 Operating System
- ii. Device Drivers
- iii. BIOS/UEFI Firmware
- iv. iOS – Apple’s mobile operating system for iPhones and iPads.

### **Application Software**

- i. Google Chrome (Web Browser)
- ii. Microsoft Word
- iii. YouTube App
- iv. WhatsApp Desktop

### **Utility Software**

- i. WinRAR
- ii. Microsoft Defender Antivirus
- iii. CCleaner
- iv. 7-Zip

## 15. Follow a GIT tutorial to practice cloning, branching, and merging repositories.

### ➤ Cloning a Repository :-

```
C:\Users\ADMIN> git clone https://github.com/RajPandya19/My_First.git
Cloning into 'My_First'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.
```

### ➤ Branching a Repository :-

```
C:\Users\ADMIN\My_First> git branch
* main

C:\Users\ADMIN\My_First> git branch My_First_Code

C:\Users\ADMIN\My_First> git checkout -b My_First_Code
fatal: a branch named 'My_First_Code' already exists

C:\Users\ADMIN\My_First> git checkout -b My_First_Code1
Switched to a new branch 'My_First_Code1'
```

```
C:\Users\ADMIN\My_First> git branch
  My_First_Code
* My_First_Code1
  main
```

### ➤ Merging a Repository :-

```
C:\Users\ADMIN\My_First> git checkout main
M      README.md
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\ADMIN\My_First> git merge feature-branch
merge: feature-branch - not something we can merge

C:\Users\ADMIN\My_First> git merge My_First_Code
Already up to date.
```

## 16. Write a report on the various types of application software and how they improve productivity.

### ❖ Report on the Various Types of Application Software and How They Improve Productivity :-

#### I. Introduction

Application software refers to computer programs designed to help users perform specific tasks or solve particular problems. Unlike system software, which manages and controls computer hardware, application software is user-focused. It enables individuals, businesses, and organizations to complete work efficiently, accurately, and creatively.

This report discusses three key types of application software and explains how each type contributes to improving productivity.

#### II. Types of Application Software

##### II.I Word Processing Software

- **Description:** Used to create, edit, format, and print text documents.
- **Examples:** Microsoft Word, Google Docs.
- **Productivity Benefits:**
  - Speeds up document creation with templates and formatting tools.
  - Enables quick editing and error correction.
  - Facilitates collaboration through track changes and comments.

## II.II Spreadsheet Software

- **Description:** Designed to organize, calculate, and analyze data using tables, formulas, and charts.
- **Examples:** Microsoft Excel, Google Sheets.
- **Productivity Benefits:**
  - Automates complex calculations.
  - Provides data visualization through graphs and charts.
  - Simplifies financial and statistical analysis.

## II.III Presentation Software

- **Description:** Creates visually appealing slideshows for communication and teaching.
- **Examples:** Microsoft PowerPoint, Google Slides.
- **Productivity Benefits:**
  - Enhances communication of ideas with visuals.
  - Offers templates that save design time.
  - Supports integration of multimedia for engaging presentations.

## III. How Application Software Improves Productivity

I. **Automation of Tasks** - Reduces manual effort and saves time.

II. **Accuracy and Consistency** - Minimizes errors in data entry and formatting.

III. **Collaboration** - Enables multiple users to work together in real time.

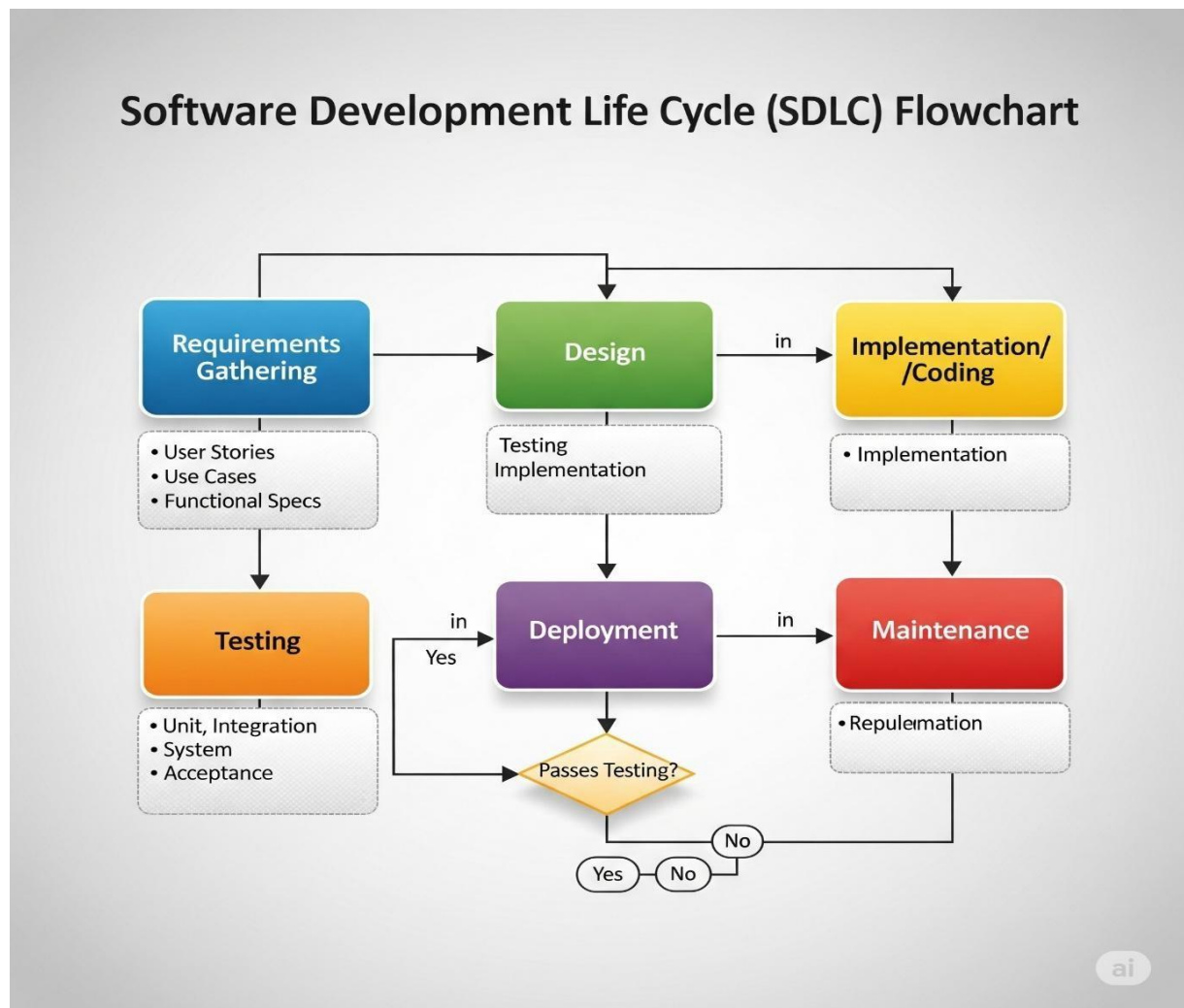
IV. **Flexibility and Mobility** - Cloud-based apps allow work from anywhere.

V. **Better Decision-Making** - Data and visuals improve clarity and understanding.

## IV. Conclusion

The three types of application software discussed—word processing, spreadsheet, and presentation tools—play a significant role in improving productivity across education, business, and personal tasks. They streamline work processes, enhance communication, and support better decision-making. As technology advances, these tools continue to evolve, offering users more features and efficiency in their work.

## 17. Create a flowchart representing the Software Development Life Cycle (SDLC).



## 18. Write a requirement specification for a single library management system.

### ❖ Requirement Specification - Single Library Management System

#### Purpose:

A system to manage books, members, and borrowing/returning activities in a library, replacing manual record-keeping.

#### Scope:

- Maintain book and member records.
- Track book issue/return with due dates.



- Provide search and reporting features.

### **Functional Requirements:**

1. User Management: Add/update/delete members, assign IDs.
2. Book Management: Add/update/delete books, track availability.
3. Issue & Return: Record loans, returns, and late fees.
4. Search: Find books by title/author/ISBN; find members by name/ID.
5. Reports: Borrowed books list, overdue list, member history.

### **Non-Functional Requirements:**

- Fast search ( $\leq 2$  sec).
- Secure admin login.
- Daily backups.
- Simple, user-friendly interface.

### **System Features:**

- Easy management of members and books.
- Quick issuing/returning process.
- Accurate reporting and secure access.

## **19. Perform a functional analysis for an online shopping system.**

### **❖ Functional Analysis – Online Shopping System**

#### **Objective:**

To enable customers to browse, purchase, and receive products online efficiently.

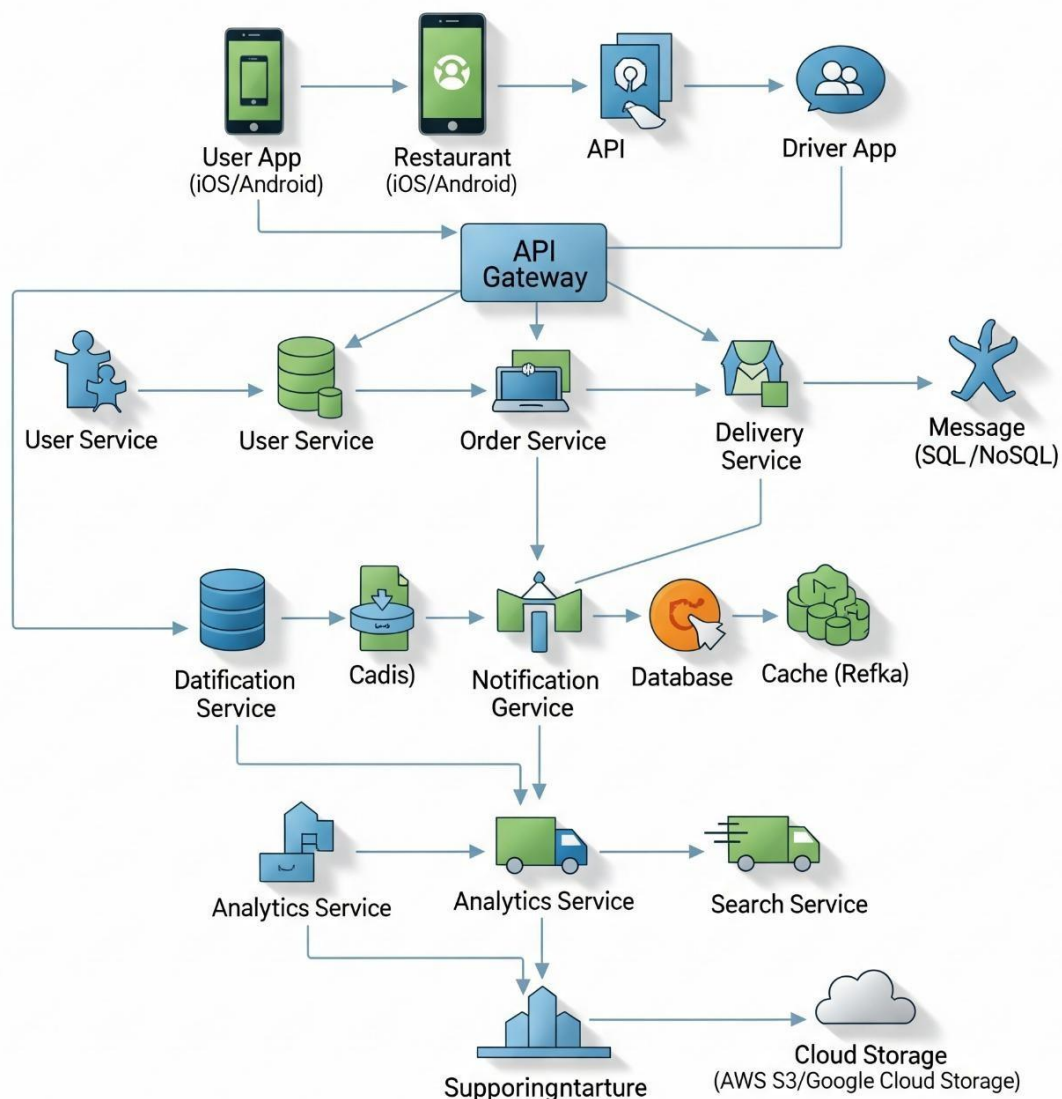
#### **Main Functional Areas:**

##### **i. User Management**

- a. User registration and login.
- b. Profile update and password management.
- c. Address and payment details storage.

- ii. **Product Management**
  - a. Add, update, and delete product listings (admin).
  - b. Categorize products (e.g., electronics, clothing).
  - c. Display product details with images and descriptions.
- iii. **Product Search s Browsing**
  - a. Search by name, category, price, or brand.
  - b. Apply filters (price range, ratings, discounts).
- iv. **Shopping Cart s Wishlist**
  - a. Add/remove products to cart.
  - b. Save products in wishlist for later purchase.
  - c. Update product quantity in cart.
- v. **Order Processing**
  - a. Checkout with selected payment method (card, UPI, COD).
  - b. Apply discount codes or offers.
  - c. Generate order ID and confirmation.
- vi. **Paynent Gateway Integration**
  - a. Secure payment processing.
  - b. Transaction success/failure notifications.
- vii. **Order Tracking s Delivery**
  - a. View order status (processing, shipped, delivered).
  - b. Track shipment with tracking ID.
- viii. **Customer Support**
  - a. Contact form, live chat, or helpdesk.
  - b. Order cancellation or return requests.
- ix. **Adnin Functions**
  - a. Manage products, categories, and stock.
  - b. View sales reports and user activity.

## 20. Design a basic system architecture for a food delivery app.



This architecture is designed to be scalable, reliable, and efficient, ensuring a smooth experience for all users. It is broken down into three main components:

### Client-Side (User-Facing Applications)

- **Customer App:** This is the primary interface for users to browse restaurants, place orders, and make payments. It is available on both iOS and Android platforms to ensure a wide reach.
- **Restaurant App:** This application is for restaurant owners and staff to manage their menus, receive and process orders, and track their earnings.

- **Rider App:** This is used by delivery personnel to view and accept delivery requests, navigate to the restaurant and customer locations, and manage their delivery schedule.

## **Server-Side (Backend Infrastructure)**

- **API Gateway:** This acts as a single entry point for all client requests. It routes requests to the appropriate microservice, ensuring that the system is secure and easy to manage.
- **Microservices:** The backend is built using a microservices architecture, where each service is responsible for a specific function. This makes the system more resilient and easier to scale.
  - **User Service:** Manages user authentication, profiles, and account settings.
  - **Restaurant Service:** Handles restaurant information, menus, and ratings.
  - **Order Service:** Manages the entire order lifecycle, from placement to delivery.
  - **Payment Service:** Integrates with payment gateways to process transactions securely.
  - **Delivery Service:** Assigns delivery requests to riders and tracks their location in real-time.
  - **Notification Service:** Sends real-time notifications to users, restaurants, and riders via push notifications, SMS, or email.

## **Data Storage**

- **Databases:** The system uses a combination of SQL and NoSQL databases to store different types of data. For example, user and order data might be stored in a SQL database, while restaurant menus and reviews could be stored in a NoSQL database.
- **Cache:** A caching layer (e.g., Redis) is used to store frequently accessed data, reducing latency and improving performance.
- **Cloud Storage:** Media files, such as images of food items and restaurant logos, are stored in a cloud storage service like AWS S3 or Google Cloud Storage.

## 21. Develop test cases for a single calculator program.

### ❖ Test Cases - Simple Calculator Program

Assumption: Calculator supports addition (+), subtraction (−), multiplication (×), and division (÷) for two numbers.

| Test Case ID | Description                       | Input                | Expected Output          | Remarks            |
|--------------|-----------------------------------|----------------------|--------------------------|--------------------|
| TC01         | Addition of two positive numbers  | $5 + 3$              | 8                        | Pass if correct    |
| TC02         | Addition with zero                | $7 + 0$              | 7                        | Pass if correct    |
| TC03         | Addition with negative number     | $-4 + 6$             | 2                        | Pass if correct    |
| TC04         | Subtraction of two numbers        | $9 - 5$              | 4                        | Pass if correct    |
| TC05         | Subtraction resulting in negative | $3 - 8$              | -5                       | Pass if correct    |
| TC06         | Multiplication of two numbers     | $4 \times 7$         | 28                       | Pass if correct    |
| TC07         | Multiplication with zero          | $9 \times 0$         | 0                        | Pass if correct    |
| TC08         | Division of two numbers           | $12 \div 4$          | 3                        | Pass if correct    |
| TC09         | Division by one                   | $9 \div 1$           | 9                        | Pass if correct    |
| TC10         | Division by zero                  | $5 \div 0$           | Error message / Infinity | Must handle safely |
| TC11         | Floating-point addition           | $2.5 + 1.2$          | 3.7                      | Pass if correct    |
| TC12         | Large number multiplication       | $100000 \times 1000$ | 100000000                | Pass if correct    |

## **22. Document a real-world case where a software application required critical maintenance.**

### **❖ Case Study - Critical Maintenance in WhatsApp (2020 Global Outage)**

#### **Background:**

On January 19, 2020, WhatsApp experienced a major outage that prevented millions of users worldwide from sending or receiving messages for approximately 2 hours.

#### **Issue:**

A configuration error during a routine server update caused a mismatch between the application layer and database synchronization service. This led to message delivery failures and connection drops.

#### **Impact:**

- Affected over 2 billion users globally.
- Disrupted personal and business communications.
- Social media platforms saw a spike in complaints and outage reports.

#### **Maintenance Action Taken:**

- The engineering team rolled back the faulty configuration to the previous stable version.
- Performed emergency database synchronization.
- Increased monitoring alerts to detect similar issues earlier.

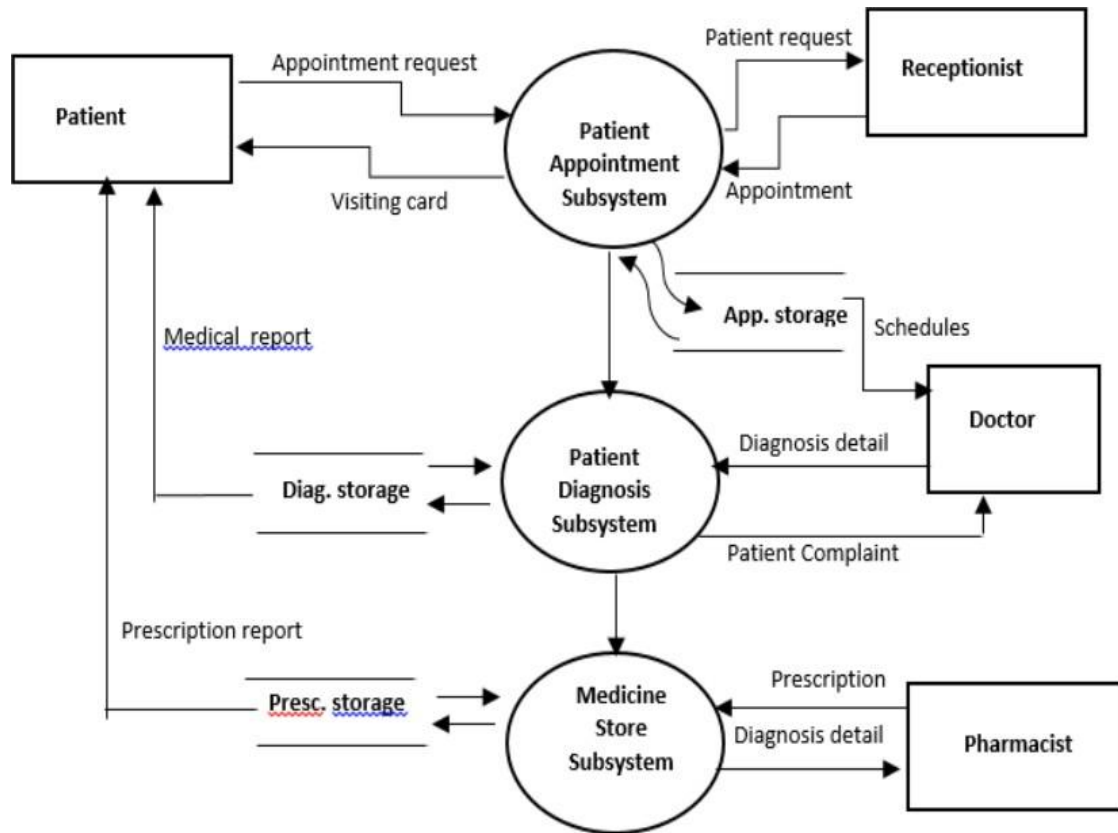
#### **Outcome:**

- Service was restored in under 2 hours.
- A post-mortem analysis identified a need for better pre-deployment testing in a staging environment.
- WhatsApp introduced automated rollback mechanisms to minimize downtime in future incidents.

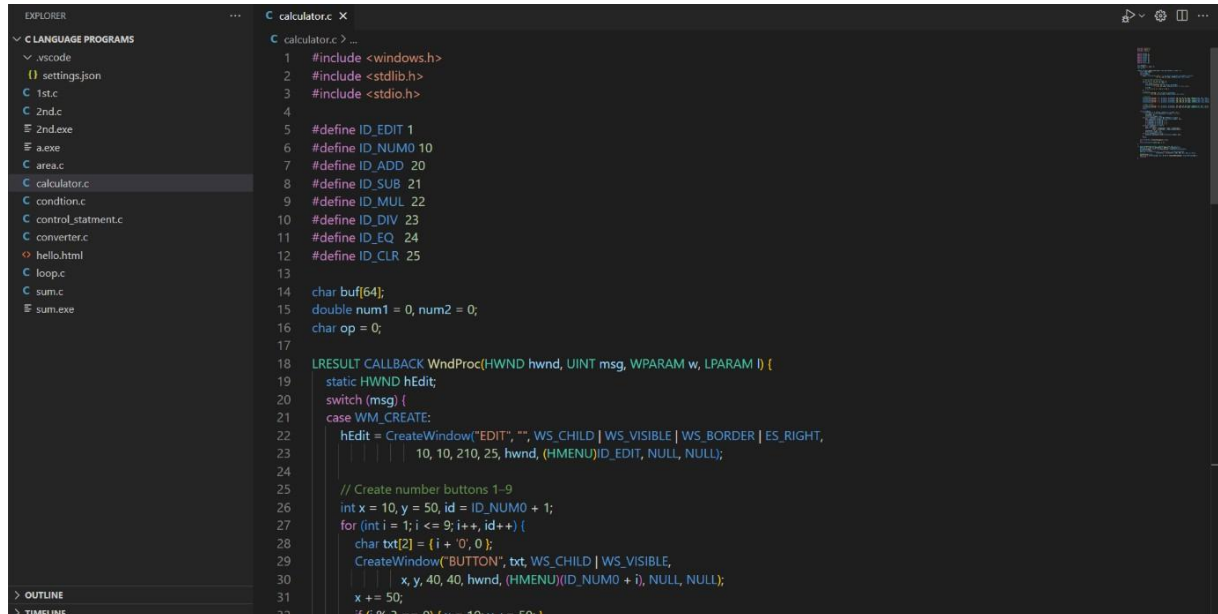
#### **Lesson Learned:**

Critical maintenance requires quick fault detection, immediate rollback plans, and improved testing procedures to prevent repeat failures.

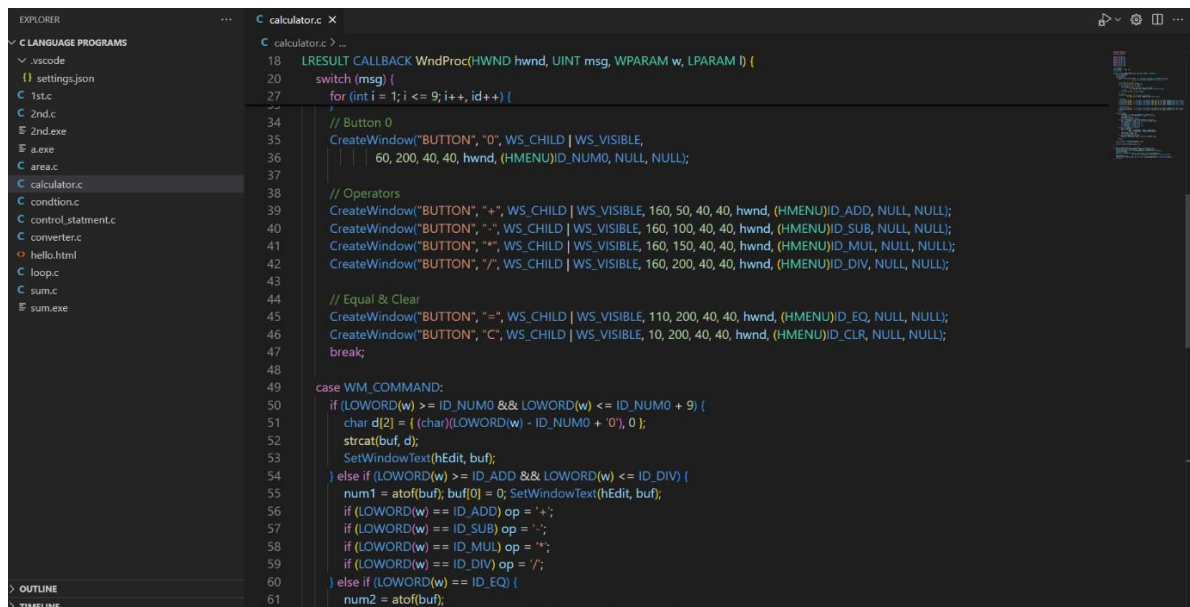
23. Create a DFD for a hospital management system.



## 24. Build a single desktop calculator application using a GUI library.



```
1 #include <windows.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4
5 #define ID_EDIT 1
6 #define ID_NUM0 10
7 #define ID_ADD 20
8 #define ID_SUB 21
9 #define ID_MUL 22
10 #define ID_DIV 23
11 #define ID_EQ 24
12 #define ID_CLR 25
13
14 char buf[64];
15 double num1 = 0, num2 = 0;
16 char op = 0;
17
18 LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM w, LPARAM l) {
19     static HWND hEdit;
20     switch (msg) {
21     case WM_CREATE:
22         hEdit = CreateWindow("EDIT", "", WS_CHILD | WS_VISIBLE | WS_BORDER | ES_RIGHT,
23             10, 10, 210, 25, hwnd, (HMENU)ID_EDIT, NULL, NULL);
24
25         // Create number buttons 1-9
26         int x = 10, y = 50, id = ID_NUM0 + 1;
27         for (int i = 1; i <= 9; i++, id++) {
28             char txt[2] = { i + '0', 0 };
29             CreateWindow("BUTTON", txt, WS_CHILD | WS_VISIBLE,
30                 x, y, 40, 40, hwnd, (HMENU)(ID_NUM0 + i), NULL, NULL);
31             x += 50;
32             if (i % 3 == 0) { x = 10; y += 50; }
```



```
18 LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM w, LPARAM l) {
20     switch (msg) {
27         for (int i = 1; i <= 9; i++, id++) {
34             // Button 0
35             CreateWindow("BUTTON", "0", WS_CHILD | WS_VISIBLE,
36                 60, 200, 40, 40, hwnd, (HMENU)ID_NUM0, NULL, NULL);
37
38             // Operators
39             CreateWindow("BUTTON", "+", WS_CHILD | WS_VISIBLE, 160, 50, 40, 40, hwnd, (HMENU)ID_ADD, NULL, NULL);
40             CreateWindow("BUTTON", "-", WS_CHILD | WS_VISIBLE, 160, 100, 40, 40, hwnd, (HMENU)ID_SUB, NULL, NULL);
41             CreateWindow("BUTTON", "*", WS_CHILD | WS_VISIBLE, 160, 150, 40, 40, hwnd, (HMENU)ID_MUL, NULL, NULL);
42             CreateWindow("BUTTON", "/", WS_CHILD | WS_VISIBLE, 160, 200, 40, 40, hwnd, (HMENU)ID_DIV, NULL, NULL);
43
44             // Equal & Clear
45             CreateWindow("BUTTON", "=", WS_CHILD | WS_VISIBLE, 110, 200, 40, 40, hwnd, (HMENU)ID_EQ, NULL, NULL);
46             CreateWindow("BUTTON", "C", WS_CHILD | WS_VISIBLE, 10, 200, 40, 40, hwnd, (HMENU)ID_CLR, NULL, NULL);
47             break;
48
49     case WM_COMMAND:
50         if (LOWORD(w) >= ID_NUM0 && LOWORD(w) <= ID_NUM0 + 9) {
51             char d[2] = { (char)(LOWORD(w) - ID_NUM0 + '0'), 0 };
52             strcat(buf, d);
53             SetWindowText(hEdit, buf);
54         } else if (LOWORD(w) >= ID_ADD && LOWORD(w) <= ID_DIV) {
55             num1 = atof(buf); buf[0] = 0; SetWindowText(hEdit, buf);
56             if (LOWORD(w) == ID_ADD) op = '+';
57             if (LOWORD(w) == ID_SUB) op = '-';
58             if (LOWORD(w) == ID_MUL) op = '*';
59             if (LOWORD(w) == ID_DIV) op = '/';
60         } else if (LOWORD(w) == ID_EQ) {
61             num2 = atof(buf);
```



```
18 LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM w, LPARAM l) {
20     switch (msg) {
21         case WM_INITMENUDEFID:
22             return 0;
23         case WM_COMMAND:
24             if (LOWORD(w) == ID_ADD) {
25                 double num1 = atof(buf);
26                 double num2 = atof(buf);
27                 double r = (op == '+') ? num1 + num2 : (op == '-') ? num1 - num2 :
28                     (op == '*') ? num1 * num2 : (num2 != 0 ? num1 / num2 : 0);
29                 sprintf(buf, "%.2f", r);
30                 SetWindowText(hwndEdit, buf);
31             } else if (LOWORD(w) == ID_CLEAR) {
32                 buf[0] = 0; num1 = num2 = 0; op = 0; SetWindowText(hwndEdit, buf);
33             }
34             break;
35         case WM_DESTROY: PostQuitMessage(0); break;
36     }
37     return DefWindowProc(hwnd, msg, w, l);
38 }

int WINAPI WinMain(HINSTANCE h, HINSTANCE p, LPSTR cmd, int n) {
    WNDCLASS wc = {0}; wc.lpfnWndProc = WndProc; wc.hInstance = h;
    wc.lpszClassName = "Calc"; wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
    RegisterClass(&wc);
    HWND win = CreateWindow("Calc", "C Calculator", WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, CW_USEDEFAULT, 250, 300, NULL, NULL, h, NULL);
    ShowWindow(win, n);
    MSG m; while (GetMessage(&m, NULL, 0, 0)) { TranslateMessage(&m); DispatchMessage(&m); }
    return 0;
}
```

```
C calculator.c > WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
18 LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM w, LPARAM l) {
20     switch (msg) {
21         case WM_INITMENUDEFID:
22             return 0;
23         case WM_COMMAND:
24             if (LOWORD(w) == ID_ADD) {
25                 double num1 = atof(buf);
26                 double num2 = atof(buf);
27                 double r = (op == '+') ? num1 + num2 : (op == '-') ? num1 - num2 :
28                     (op == '*') ? num1 * num2 : (num2 != 0 ? num1 / num2 : 0);
29                 sprintf(buf, "%.2f", r);
30                 SetWindowText(hwndEdit, buf);
31             } else if (LOWORD(w) == ID_CLEAR) {
32                 buf[0] = 0; num1 = num2 = 0; op = 0; SetWindowText(hwndEdit, buf);
33             }
34             break;
35         case WM_DESTROY: PostQuitMessage(0); break;
36     }
37     return DefWindowProc(hwnd, msg, w, l);
38 }

int WINAPI WinMain(HINSTANCE h, HINSTANCE p, LPSTR cmd, int n) {
    WNDCLASS wc = {0}; wc.lpfnWndProc = WndProc; wc.hInstance = h;
    wc.lpszClassName = "Calc"; wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
    RegisterClass(&wc);
    HWND win = CreateWindow("Calc", "C Calculator", WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, CW_USEDEFAULT, 250, 300, NULL, NULL, h, NULL);
    ShowWindow(win, n);
    MSG m; while (GetMessage(&m, NULL, 0, 0)) { TranslateMessage(&m); DispatchMessage(&m); }
    return 0;
}
```

```
PS D:\C Language Programs> gcc .\calculator.c
PS D:\C Language Programs> .\a.exe
```

```
C calculator.c X
C calculator.c > WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
18 LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM w, LPARAM l) {
20     switch (msg) {
60         } else if (LOWORD(w) == ID_EQ) {
63             SetWindowText(hwnd, buf);
66         } else if (LOWORD(w) == ID_CLR) {
67             buf[0]=0; num1=num2=0; op=0;
68         }
69         break;
70     }
71     case WM_DESTROY: PostQuitMessage(0);
72     }
73     return DefWindowProc(hwnd, msg, w, l);
74 }
75
76 int WINAPI WinMain(HINSTANCE h, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
77 {
78     WNDCLASS wc = {0}; wc.lpszClassName = "Calc"; wc.hbrBackground = (HBRUSH)COLOR_WINDOW+1;
79     RegisterClass(&wc);
80     HWND win = CreateWindow("Calc", "C Calculator", WS_OVERLAPPEDWINDOW,
81                             CW_USEDEFAULT, CW_USEDEFAULT, 250, 300, NULL, NULL, h, NULL);
82     ShowWindow(win, nCmdShow);
83     UpdateWindow(win);
84     MSG msg;
85     while ((msg = GetMessage(&msg, NULL, 0, 0)) != 0) {
86         TranslateMessage(&msg);
87         DispatchMessage(&msg);
88     }
89     return msg.wParam;
90 }
```

C Calculat...

1221.00

|   |   |   |   |
|---|---|---|---|
| 1 | 2 | 3 | + |
| 4 | 5 | 6 | - |
| 7 | 8 | 9 | * |
| C | 0 | = | / |

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS D:\C Language Programs> gcc .\calculator.c
PS D:\C Language Programs> .\a.exe
```

25. Draw a flowchart representing the logic of a basic online registration system.

