

INSTITUTE :- TOPS TECHNOLOGIES (BARODA)

NAME :- SRUJAL SOLANKI M

COURSE :- JAVA FULL STACK DEVELOPER

ASSIGNMENT 1 :- MODULE 1 OVERVIEW OF IT INDUSTRY

Module 1 – Overview of IT Industry

1) What is a Program?

-> A program is a set of instructions that tells a computer what to do. It tells the computer how to perform tasks like calculations, displaying information, or managing data.

2) LAB EXERCISE: Write a simple "Hello World" program in two different programming languages of

your choice. Compare the structure and syntax.

->

1]#include<stdio.h> // header

2]int main () { // main() function

3]printf(" SAM:Hello world"); //variable declaration

4] return 0; //return type of the program

5]}

number 3 to 5 is body

3.THEORY EXERCISE: Explain in your own words what a program is and how it functions.

What is Programming?

A program is a set of instructions that tells a computer what to do. It tells the computer how to perform tasks like calculations, displaying information, or managing data.

How a Program Works like

Written in Code: A programmer writes the instructions using a programming language (like c or Java).

Converted to Machine Code: The code is turned into a format that the computer can understand.

Execution: The computer follows the instructions one by one to complete the tasks.

4.THEORY EXERCISE: What are the key steps involved in the programming process?

Types of Programming Languages

1) problem identification

2) planning

3)coding

4)testing

5)debugging

6)documentation

7)maintenance

typre of programming language

high leve; language

low level language

scripting language

object oriented language

domain- specific language

5.THEORY EXERCISE: What are the main differences between high-level and low-level

programming lanuage

->High-Level Languages

Abstraction: More abstract and easier to understand.

Readability: Code is closer to human language.

Ease of Use: Easier to learn and faster to write.

Performance: Generally slower due to abstraction.

Portability: Can run on different systems with little change.

Examples: Python, Java, C#, Ruby.

->Low-Level Languages :

Abstraction: Less abstract; closer to machine code.

Readability: Harder for humans to read and understand.

Ease of Use: More complex and harder to learn.

Performance: Faster and more efficient on hardware.

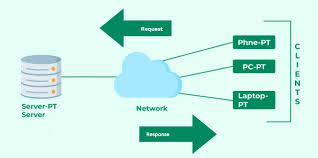
Portability: Less portable; often tied to specific hardware.

Examples: Assembly language, machine code.

World Wide Web & How Internet Works

5.LAB EXERCISE: Research and create a diagram of how data is transmitted from a client to a server

over the internet.



6.THEORY EXERCISE: Describe the roles of the client and server in web communication.

Network Layers on Client and Server

Roles of the Client and Server in Web Communication

1.Client:

- The client is usually your device (computer, phone, etc.) with a browser (like Chrome or Firefox).

- Its role is to ask for information from the server.

- Example: When you type a website address, the client sends a request to the server to get the web page.

2. Server:

- The server is a powerful computer or program that stores information like web pages, images, or videos.

- Its role is to respond to the client’s request by sending the requested data.

- Example: When a client asks for a web page, the server sends back the HTML, CSS, and other files.

Network Layers on Client and Server

Web communication is based on a layered model called the OSI Model:

1. Application Layer:

- Client: Sends requests (like "GET website.com").

- Server: Processes the request and sends the response.

- Examples: HTTP, HTTPS.

2. Transport Layer:

- Ensures data is sent and received reliably.

- Client: Breaks down data into packets and sends them.

- Server: Receives packets and reassembles them.

- Protocols: TCP (reliable), UDP (faster but less reliable).

3. Network Layer:

- Handles the addressing and routing of data.

- Client: Adds the destination IP address (server's address).

- Server: Receives data addressed to its IP.

- Protocol: IP (Internet Protocol).

4. Data Link Layer:

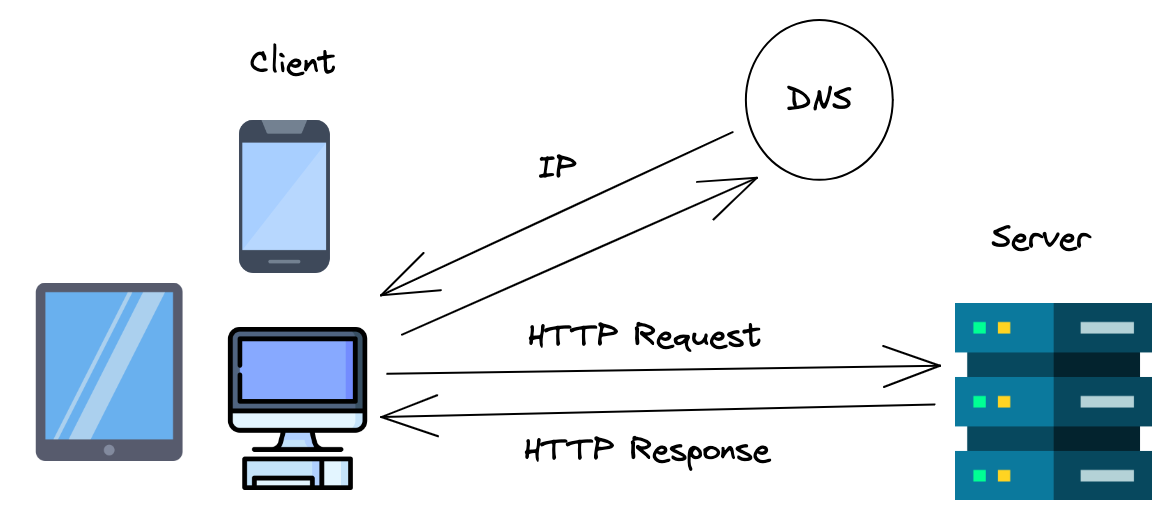
- Ensures the data is delivered to the right physical device.

- Works with MAC addresses.

5. Physical Layer:

- Converts data into electrical signals, light pulses, or radio waves for transmission.

7.LAB EXERCISE: Design a simple HTTP client-server communication in any language.



THEORY EXERCISE: Explain the function of the TCP/IP model and its layers.

Client and Servers

TCP/IP Model and Its Layers

The TCP/IP model is like a set of rules that allows computers to communicate with each other over a network (like the internet). It has \*\*4 layers\*\*, and each layer has a specific job:

1. Application Layer

This is where users interact. It handles software and applications, like web browsers or email programs.

Example Opening a website or sending an email.

2. Transport Layer

This ensures data is delivered correctly between devices. It divides data into smaller pieces and ensures they arrive safely.

Example: Making sure the whole file gets downloaded, not just parts of it.

3. Internet Layer

This decides the best path to send data and addresses the data packets. It works like GPS for your data.

Example: Finding the fastest route to send a message to another computer.

4. Network Interface Layer

This is the physical connection. It handles hardware like cables or Wi-Fi and sends the actual bits of data.

Example: Sending 1s and 0s through wires or wireless signals.

Client and Server

- A client is like a person making a request. It asks for services or resources.

Example Your browser asking for a webpage.

- A server is like a shopkeeper. It provides the requested service or resource.

Example A web server sending the webpage back to your browser.

In simple terms: the client asks, and the server delivers.

8.THEORY EXERCISE: Explain Client Server Communication

Types of Internet Connections

Client-server communication is an essential concept in networking that underpins many modern applications and services. Understanding this model is crucial for designing efficient systems and ensuring effective data exchange.

Types of Internet Connections

1. Dial-Up
2. DSL (Digital Subscriber Line)
3. Cable
4. Fiber Optic
5. Satellite
6. Wireless (Wi-Fi)

Mobile Data (3G/4G/5G)

1. Broadband Over Powerline (BPL)
2. Fixed Wireless

9.LAB EXERCISE: Research different types of internet connections (e.g., broadband, fiber,

satellite)and list their pros and cons.

**1. Dial-Up**

* **Pros**:
  + Available almost everywhere.
  + Low cost.
* **Cons**:
  + Very slow speeds (up to 56 Kbps).
  + Can’t use the phone while online.

**2. DSL (Digital Subscriber Line)**

* **Pros**:
  + Faster than dial-up.
  + Works over phone lines, so no need for new cables.
* **Cons**:
  + Speed decreases with distance from the provider.
  + Slower than cable and fiber.

**3. Cable**

* **Pros**:
  + Fast speeds (10 Mbps to 1 Gbps).
  + Often includes TV and phone services.
* **Cons**:
  + Can be slower during peak times when many people are online.
  + Availability may be limited in rural areas.

**4. Fiber Optic**

* **Pros**:
  + Very high speeds (over 1 Gbps).
  + Reliable and low latency (minimal delays).
* **Cons**:
  + Not available in all areas.
  + Can be more expensive to install.

**5. Satellite**

* **Pros**:
  + Available in remote areas where other connections aren’t.
  + Can reach places without traditional infrastructure.
* **Cons**:
  + High latency (delays in communication).
  + Slower speeds (12 Mbps to 100 Mbps) and affected by weather.

**6. Wireless (Wi-Fi)**

* **Pros**:
  + Convenient for multiple devices without wires.
  + Easily set up and portable within range.
* **Cons**:
  + Depends on the quality of the connection from the main source.
  + Speed can be slower due to interference or distance from the router.

**7. Mobile Data (3G/4G/5G)**

* **Pros**:
  + Portable and accessible on the go.
  + Faster speeds with 5G (over 1 Gbps).
* **Cons**:
  + Data limits can apply, leading to extra charges.
  + Coverage may vary depending on the location.

**8. Broadband Over Powerline (BPL)**

* **Pros**:
  + Uses existing power lines, so easy to set up in some areas.
  + Can provide a decent speed (5 Mbps to 200 Mbps).
* **Cons**:
  + Less reliable than other broadband options.
  + Performance can be affected by electrical interference.

**9. Fixed Wireless**

* **Pros**:
  + Good option for rural areas.
  + No need for physical cables between homes.
* **Cons**:
  + Speeds depend on the distance from the tower.
  + Can be affected by obstacles like buildings or trees.

**10. T1/T3 Lines**

* **Pros**:
  + Reliable and fast dedicated lines for businesses.
  + Consistent speeds (up to 1.5 Mbps for T1 and 45 Mbps for T3).
* **Cons**:
  + Very expensive compared to other options.
  + Typically only available for businesses, not for residential use.

10.THEORY EXERCISE: How does broadband differ from fiber-optic internet?

|  |  |
| --- | --- |
| Broadband | Fiber -optic |
| High-speed internet encompassing various technologies (DSL, cable, satellite, fiber). | A type of broadband using fiber-optic cables for data transmission. |
| Varies by technology (DSL: 1-100 Mbps, Cable: up to 1 Gbps). | Very high speeds (100 Mbps to several Gbps). |
| Can be higher, especially with non-fiber types (like DSL/satellite). | Low latency due to the speed of light transmission. |
| Limited bandwidth can lead to congestion. | High bandwidth capacity supports many devices without slowing down. |
| Reliability varies by technology (satellite affected by weather). | Highly reliable and less affected by interference. |

11.LAB EXERCISE: Simulate HTTP and FTP requests using command line tools (e.g., curl).

simulate HTTP and FTP requests using the command line tool "curl", you can use the basic command structure curl [options] [protocol://server/path] where you specify the protocol (http, ftp), server address, and path to the resource, along with various options to customize the request like headers, data, and authentication details.

Basic HTTP GET request:

* **Fetch a webpage:** curl https://www.example.com

Basic FTP request:

* **Download a file from FTP server:** curl -O ftp://ftp.example.com/file.txt

12.THEORY EXERCISE: What are the differences between HTTP and HTTPS protocols?

Application Security

|  |  |  |
| --- | --- | --- |
| **HTTP** | **HTTPS** |  |
| The full form of HTTP is Hypertext Transfer Protocol. | The full form of HTTPS is Hypertext Transfer Protocol Secure |  |
| URL begins with http:// | URL begins with https:// |  |
| It uses port number 80. | It sends the data over port number 443. |  |
| HTTP is an application layer protocol. | HTTPS is a transport layer protocol. |  |
| Less secure and vulnerable to hacking attacks. | It is highly secure. |  |
| It does not contain an SSL certificate. | HTTPS contains an SSL certificate. |  |
| HTTP websites do not use data encryption. | HTTPS websites use data encryption. |  |
| It is fast. | It is slower than HTTP. |  |
| HTTP does not help in improving search rankings. | HTTPS provides SEO advantages as Google gives the preferences to websites that use HTTPS. |  |

13.LAB EXERCISE: Identify and explain three common application security vulnerabilities.

Suggestpossible solutions.

**1. SQL Injection**

* **What is it?**  
  Attackers insert harmful code into input fields (e.g., login forms) to trick the application into revealing or changing sensitive data stored in the database.  
  Example: Instead of entering a username, an attacker types malicious code like '; DROP TABLE users; --.
* **Solution:**
  + Use **parameterized queries** or **prepared statements** to handle database queries safely.
  + **Validate and sanitize user input** to ensure only expected data is accepted.

**2. Cross-Site Scripting (XSS)**

* **What is it?**  
  Attackers inject malicious scripts (e.g., JavaScript) into a website's input fields. These scripts run in the victim's browser, stealing data like cookies or tricking users.  
  Example: An attacker might enter <script>alert('Hacked!');</script> into a comment box.
* **Solution:**
  + **Escape special characters** in user input so they cannot be treated as code.
  + Use security libraries or frameworks that help prevent XSS, like **Content Security Policy (CSP)**.

**3. Broken Authentication**

* **What is it?**  
  Weak or poorly implemented login systems allow attackers to take over user accounts.  
  Example: Not limiting login attempts makes it easier for attackers to guess passwords (brute force attack).
* **Solution:**
  + Enforce **strong password requirements** and recommend two-factor authentication (2FA).
  + Limit **failed login attempts** and use techniques like **CAPTCHA** to block automated attacks.

14.THEORY EXERCISE: What is the role of encryption in securing applications?

Software Applications and Its Types

**Role of Encryption in Securing Applications**

1. **Data Protection**: Ensures sensitive data (like passwords, credit card numbers) is unreadable to unauthorized users.
2. **Secure Communication**: Protects data transmitted between users and servers (e.g., HTTPS).
3. **Data Integrity**: Prevents tampering by ensuring encrypted data remains unchanged.
4. **Authentication**: Verifies the identity of users and systems via encrypted tokens or certificates.
5. **Compliance**: Meets security standards (e.g., GDPR, HIPAA) for handling sensitive information.

**Types of Software Applications**

1. **Web Applications**: Run on browsers, e.g., online shopping sites.
2. **Mobile Applications**: Designed for smartphones/tablets, e.g., social media apps.
3. **Desktop Applications**: Installed on computers, e.g., word processors, games.
4. **Cloud Applications**: Use internet-based services, e.g., Google Drive.
5. **Enterprise Applications**: Used by organizations for business processes

15.LAB EXERCISE: Identify and classify 5 applications you use daily as either system software

orapplication software.

1)LINKEDIN

2)WHATSAPP

3)YOUTUBE

4)BROWSER

5)UPI PAYENT BANK

16.THEORY EXERCISE: What is the difference between system software and application software?

Software Architecture

**Difference Between System Software and Application Software**

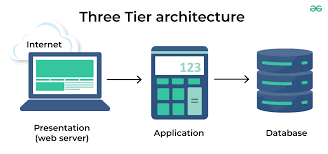
| **System Software** | **Application Software** |
| --- | --- |
| Manages hardware and provides a platform for other software. | Performs specific tasks for the user (e.g., browsing, editing). |
| Operating systems, drivers, utilities. | Web browsers, word processors, games. |
| Runs independently, supports application software. | Requires system software to function. |
| Works in the background to manage system operations. | Interacts directly with the user. |
| Usually comes pre-installed with the device. | Installed as per user needs. |

**Software Architecture** refers to the high-level structure of a software system, focusing on:

1. **Components**: Individual modules or subsystems (e.g., database, user interface).
2. **Interactions**: How components communicate with each other.
3. **Design Patterns**: Common solutions like MVC (Model-View-Controller), Microservices.
4. **Non-Functional Aspects**: Performance, scalability, security, and maintainability.

Good software architecture ensures the system is reliable, efficient, and easy to update.

17.LAB EXERCISE: Design a basic three-tier software architecture diagram for a web application.



18.THEORY EXERCISE: What is the significance of modularity in software architecture?

Layers in Software Architecture

**Significance of Modularity in Software Architecture**

1. **Improved Maintainability**: Breaking software into modules makes it easier to update or fix specific parts without affecting the whole system.
2. **Reusability**: Modules can be reused across different projects, reducing development effort.
3. **Scalability**: Modular design enables adding new features or scaling parts of the system independently.
4. **Enhanced Collaboration**: Teams can work on separate modules simultaneously, speeding up development.
5. **Better Testing**: Each module can be tested independently, improving overall quality and reliability.

**Layers in Software Architecture**

1. **Presentation Layer**
   * **Purpose**: Handles user interactions (UI/UX).
   * **Examples**: Web browsers, mobile apps.
2. **Application Layer**
   * **Purpose**: Contains business logic and processes user requests.
   * **Examples**: API services, application servers.
3. **Data Layer**
   * **Purpose**: Manages data storage, retrieval, and communication with databases.
   * **Examples**: Database servers, data access components.
4. **Integration Layer** (optional in some architectures)
   * **Purpose**: Connects the application to external systems or services

19.LAB EXERCISE: Create a case study on the functionality of the presentation, business logic, and

dataaccess layers of a given software system.

Case Study: Functionality of the Presentation, Business Logic, and Data Access Layers in a Software System

Introduction

Modern software systems are structured using a layered architecture to enhance modularity, scalability, and maintainability. The three common layers are:

1. Presentation LayerThe user interface.

2. Business Logic LayerProcesses application rules and workflows.

3. Data Access LayerManages data storage and retrieval.

This case study focuses on these layers in a hypothetical e-commerce system, ShopEase.

1. Presentation Layer

Handles user interaction and communicates with the business logic layer.

Example: A user searches for a product. The presentation layer captures the input and sends it for processing.

Technologies: React, Angular, Flutter.

2. Business Logic Layer

Implements application rules and ensures workflows are followed.

Example: Applies discounts and checks inventory before confirming an order.

Technologies: Spring Boot, .NET Core, Python.

3. Data Access Layer

Handles data operations like fetching and saving information.

Example: Retrieves product details or saves order data.

Technologies: PostgreSQL, MongoDB, Hibernate.

Interaction Between Layers

- Step 1: User selects products via the presentation layer.

- Step 2: Business logic validates input and calculates totals.

- Step 3: Data access retrieves and saves order data.

- Step 4: Presentation layer displays confirmation.

Benefits

- Modularity -Independent development and testing of layers.

- Scalability: Layers can scale separately.

- Security: Sensitive operations confined to specific layers.

20.THEORY EXERCISE: Why are layers important in software architecture?

Software Environments

**Layers in Software Architecture** are important because:

1. **Organizing Tasks**: Layers help separate different parts of the software, like user interface, logic, and data handling, so each part is easier to work on.
2. **Easier Changes**: You can change one part of the system without affecting the others.
3. **Better Maintenance**: It’s easier to fix or update the system when it’s divided into layers.
4. **Reuse**: You can use parts of the system in other places without rewriting them.
5. **Security**: Layers help protect the system by controlling how data moves between different parts.
6. **Growth**: Layers make it easier to improve or expand the software over time.

**Software Environments**:

1. **Development Environment**: Where programmers create and test code.
2. **Testing Environment**: Where the software is checked for problems before it’s used by others.
3. **Staging Environment**: A copy of the live system used for final tests.
4. **Production Environment**: The live system that real users interact with.

21.LAB EXERCISE: Explore different types of software environments (development, testing,

production).Set up a basic environment in a virtual machine.

**Types of Software Environments:**

1. **Development Environment**:
   * Used to write and test code.
   * Includes tools like code editors (e.g., Visual Studio Code) and version control (e.g., Git).
2. **Testing Environment**:
   * Used to check if the code works properly.
   * Tools like testing frameworks (e.g., Selenium, JUnit) are used here.
3. **Production Environment**:
   * The live version of the software used by real users.
   * Needs to be stable and secure.

**Setting Up a Basic Environment in a Virtual Machine (VM):**

1. **Install Virtualization Software**:
   * Use tools like VirtualBox or VMware to create a virtual machine on your computer.
2. **Install an Operating System**:
   * Download and install an OS like Ubuntu or Windows in the VM.
3. **Set Up Development Tools**:
   * Install code editors (e.g., Visual Studio Code) and tools like Git and programming languages (e.g., Python, Java).
4. **Set Up Testing Tools**:
   * Install testing tools (e.g., Selenium for web apps, JUnit for Java) to check the code.
5. **Set Up Production Environment (Optional)**:
   * Install a web server (e.g., Apache) and database (e.g., MySQL) if needed.
6. **Test and Monitor**:
   * Make sure everything works properly in each environment and monitor the performance.

.22.THEORY EXERCISE: Explain the importance of a development environment in software production.

The **development environment** is important in software production because:

1. **Efficient Coding**: It provides tools like code editors and compilers to help developers write code quickly and easily.
2. **Testing**: Developers can test small parts of the software as they work, finding and fixing bugs early.
3. **Collaboration**: Developers can use version control (like Git) to work together without overwriting each other’s work.
4. **Customization**: Developers can set up the environment to fit their needs (e.g., programming languages, libraries, frameworks).
5. **Faster Development**: It speeds up the process by offering tools for automating tasks like building and testing the software.
6. **Error Handling**: It provides debugging tools to help developers find and fix errors in the code.

23.LAB EXERCISE: Write and upload your firstsource code file to Github.

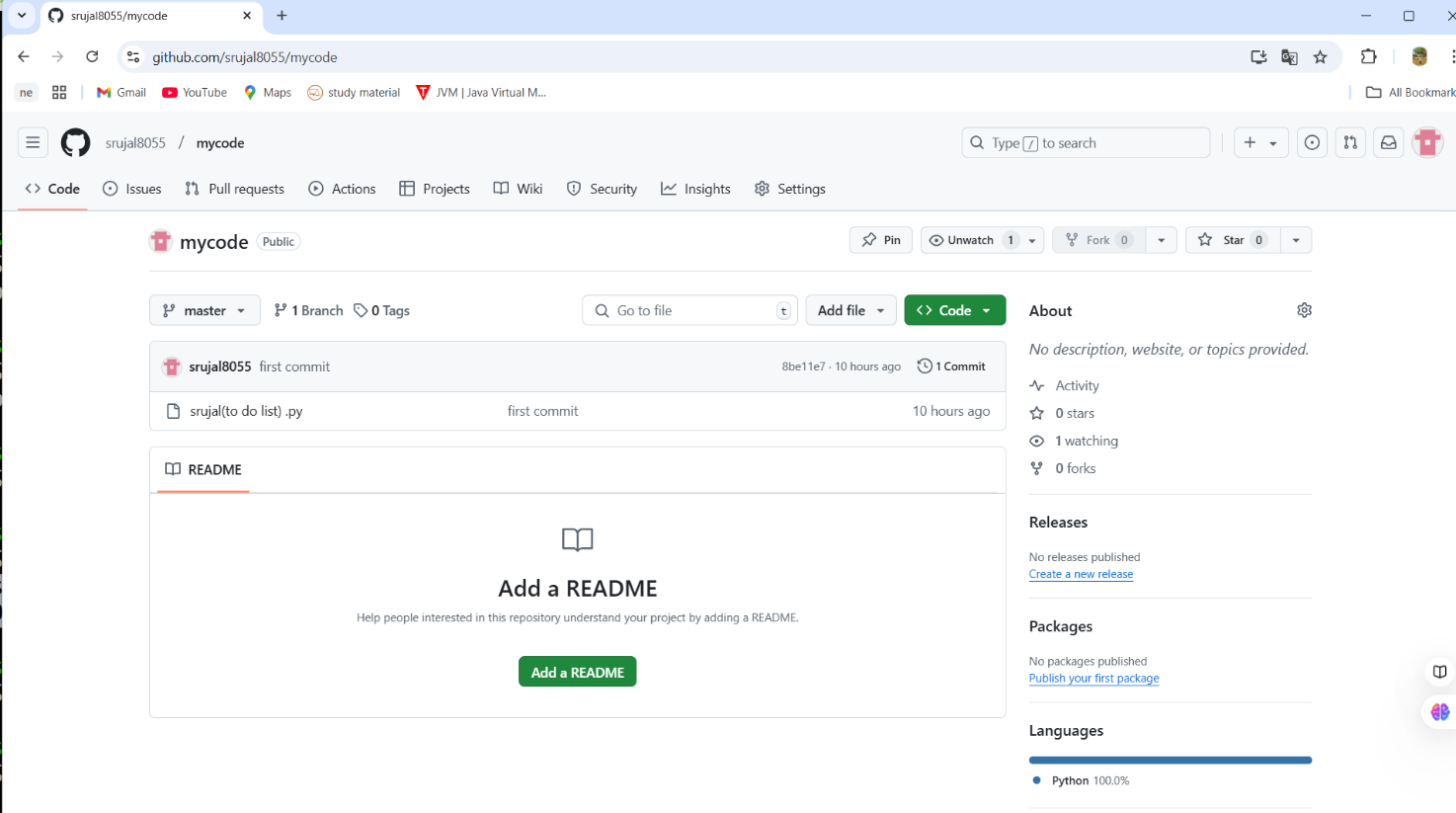
24.THEORY EXERCISE: What is the difference between source code and machine code?

|  |  |
| --- | --- |
| Source code | Machine code |
| Written by programmers in high-level languages like Python, Java, or C++. | Made up of binary (0s and 1s) instructions that the computer's hardware understands. |
| Easy to read, write, and modify. | Difficult for humans to read or modify. |
| : Must be converted into machine code using a compiler or interpreter | Directly executed by the CPU. |
| example: print("Hello, World!") | example: 10101000 00000001 |

Github and Introductions

25.LAB EXERCISE: Create a Github repository and document how to commit and push code changes.

My repository :-[MY GITHUB LINK](https://github.com/srujal8055/mycode)



26.THEORY EXERCISE: Why is version control important in software development?

1. **Tracks Changes**: Keeps a history of all code changes, so you can see who made what changes and when.
2. **Collaboration**: Allows multiple developers to work on the same project without overwriting each other’s work.
3. **Reverts Mistakes**: Makes it easy to undo mistakes by reverting to an earlier version of the code.
4. **Backup**: Acts as a safety net, storing the code in a secure and central repository.
5. **Experimentation**: Supports branching to test new features without affecting the main code.

Student Account in Github

27.LAB EXERCISE: Create a student account on Github and collaborate on a small project with

aclassmate.

Its my project ->link below

[MY GITHUB PROJECT](https://github.com/srujal8055/codesoft)

28.THEORY EXERCISE: What are the benefits of using Github for students?

**Benefits of Using GitHub for Students**

1. **Version Control**:
   * Keep track of changes in your projects and easily revert to earlier versions.
2. **Collaboration**:
   * Work with classmates on group projects, sharing and managing code easily.
3. **Showcase Your Work**:
   * Build a public portfolio of your work to show potential employers.
4. **Learn Industry-Standard Tools**:
   * Gain experience with Git and GitHub, widely used in professional software development.
5. **Backup Projects**:
   * Store and back up your work in the cloud, preventing data loss.
6. **Access Anywhere**:
   * Access your code from any device with an internet connection.
7. **Open Source Contributions**:
   * Contribute to open-source projects, learning from others and giving back to the community.
8. **Free for Students**:
   * GitHub offers free features like private repositories for students.

Types of Software

29.LAB EXERCISE: Create a list of software you use regularly and classify them into the

followingcategories: system, application, and utility software.

**System Software**

1. **Operating System (OS)**:
   * Windows, macOS, Linux
2. **Device Drivers**:
   * Printer drivers, Graphics card drivers
3. **Firmware**:
   * BIOS, Embedded software on devices

**Application Software**

1. **Web Browsers**:
   * Google Chrome, Mozilla Firefox, Safari
2. **Email Clients**:
   * Microsoft Outlook, Apple Mail
3. **Media Players**:
   * VLC Media Player, Windows Media Player
4. **Photo Editing Software**:
   * Adobe Photoshop, GIMP

**Utility Software**

1. **Antivirus Software**:
   * Norton, McAfee, Avast
2. **Backup Software**:
   * Acronis True Image, Google Backup
3. **File Compression Tools**:
   * WinRAR, 7-Zip
4. **Disk Cleaners**:
   * CCleaner, Disk Cleanup
5. **System Monitoring Tools**:
   * Task Manager, CPU-Z

These software types help in managing the system (system software), performing tasks (application software), and maintaining the system (utility software).

30.THEORY EXERCISE: What are the differences between open-source and proprietary software?

GIT and GITHUB Training

| **GIT** | **GitHub** |
| --- | --- |
| A tool to track changes in code locally. | A platform to store and share code online. |
| Version control for code. | Collaboration and hosting for Git repositories. |
| Works offline. | Requires the internet to upload or share. |
| Local changes tracking (git init, git add, etc.). | Online sharing, pull requests, and collaboration. |
| Open-source software. | Owned by Microsoft. |
| Used to save code changes on your computer. | Used to store repositories and share with teams. |

31.LAB EXERCISE: Follow a GIT tutorial to practice cloning, branching, and merging repositories.

**Git Tutorial Summary:**

1. **Clone a Repository**
2. git clone <repository\_url>
3. cd <repository\_name>
4. **Create and Switch Branch**
5. git checkout -b <branch\_name>
6. **Make Changes**
   * Edit files.
   * Stage changes:
   * git add .
   * Commit changes:
   * git commit -m "Message"
7. **Merge Branches**
   * Switch to main branch:
   * git checkout main
   * Merge:
   * git merge <branch\_name>
8. **Push Changes to Remote**
9. git push origin <branch\_name>
10. **Clean Up**
11. git branch -d <branch\_name>
12. git push origin --delete <branch\_name>

32.THEORY EXERCISE: How does GIT improve collaboration in a software development team?

1. **Centralized Repository**: Shared access to a central codebase.
2. **Version Control**: Tracks changes and allows rollbacks.
3. **Branching**: Enables independent work on features or fixes.
4. **Merging**: Combines updates seamlessly into the main codebase.
5. **Conflict Resolution**: Detects and helps resolve code conflicts.
6. **Parallel Development**: Multiple developers work simultaneously.
7. **Code Reviews**: Supports pull requests and reviews for better quality.
8. **Automation**: Integrates with CI/CD pipelines for testing and deployment.
9. **Accountability**: Tracks who made changes and why.
10. **Backup**: Local and remote copies ensure disaster recovery.

Application Software

33.LAB EXERCISE: Write a report on the various types of application software and how they

improveproductivity.

-> Application software refers to programs designed to perform specific tasks for users, enhancing efficiency and productivity in various domains. Below are the main types of application software and their impact on productivity:

**1. Productivity Software**

* **Examples**: Microsoft Office, Google Workspace.
* **Functions**: Word processing, spreadsheets, presentations.
* **Impact**: Improves documentation, data analysis, and collaboration, enabling faster and more accurate work.

**2. Business Software**

* **Examples**: CRM (Salesforce), ERP (SAP).
* **Functions**: Manages business processes like sales, inventory, and accounting.
* **Impact**: Streamlines operations, reduces manual tasks, and improves decision-making.

**3. Communication Software**

* **Examples**: Zoom, Slack, Microsoft Teams.
* **Functions**: Facilitates instant messaging, video conferencing, and file sharing.
* **Impact**: Enhances team collaboration, reduces travel costs, and improves remote work efficiency.

**4. Multimedia Software**

* **Examples**: Adobe Photoshop, VLC Media Player.
* **Functions**: Handles video, image, and audio editing.
* **Impact**: Supports creative tasks and boosts the quality of digital content production.

**5. Educational Software**

* **Examples**: Duolingo, Moodle.
* **Functions**: Offers learning tools and training resources.
* **Impact**: Enables self-paced learning and improves skill acquisition.

**6. Utility Software**

* **Examples**: Antivirus programs, file compression tools.
* **Functions**: Optimizes system performance and ensures security.
* **Impact**: Saves time by automating maintenance tasks and protecting data.

**7. Database Software**

* **Examples**: MySQL, Microsoft Access.
* **Functions**: Manages and organizes large datasets.
* **Impact**: Speeds up data retrieval and improves data management.

34.THEORY EXERCISE: What is the role of application software in businesses?

Software Development Process

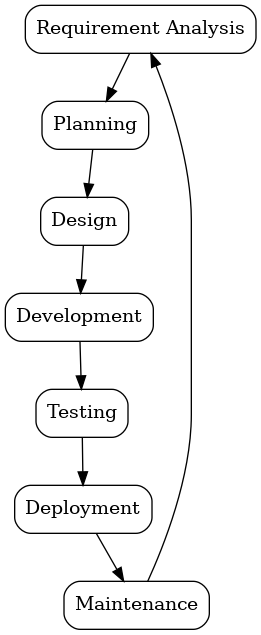
**Role of Application Software in Businesses**

Application software plays a vital role in helping businesses operate efficiently and achieve their goals. Its key roles include:

1. **Automating Tasks**: Reduces manual work by automating repetitive tasks like data entry, accounting, and payroll.
2. **Improving Communication**: Tools like email clients and messaging apps enhance team collaboration and customer interactions.
3. **Streamlining Operations**: Business software like CRM and ERP systems help manage workflows, track sales, and optimize processes.
4. **Data Management**: Database applications store, organize, and analyze business data for better decision-making.
5. **Enhancing Productivity**: Productivity tools like spreadsheets, word processors, and scheduling software save time and improve accuracy.
6. **Cost Efficiency**: Reduces operational costs by improving resource utilization and minimizing errors.

**Software Development Process**

1. **Requirement Analysis**
   * Understand the purpose of the software and what users need.
2. **Planning**
   * Create a roadmap for development, set timelines, and allocate resources.
3. **Design**
   * Develop blueprints or models for the software’s structure and user interface.
4. **Development (Coding)**
   * Programmers write the code based on the design.
5. **Testing**
   * Check the software for errors, bugs, and performance issues.
6. **Deployment**
   * Release the software to users, either locally or online.
7. **Maintenance**
   * Continuously update and fix issues to improve functionality.

35.LAB EXERCISE: Create a flowchart representing the Software Development Life Cycle (SDLC)

36.THEORY EXERCISE: What are the main stages of the software development process?

**Main Stages of the Software Development Process**

1. **Requirement Analysis**
   * Understand what the software needs to do based on user and business needs.
2. **Planning**
   * Set goals, timeline, and resources for the project.
3. **Design**
   * Create a blueprint for how the software will look and function.
4. **Development (Coding)**
   * Programmers write the code to build the software.
5. **Testing**
   * Check for bugs, errors, and ensure everything works as planned.
6. **Deployment**
   * Launch the software for users to access and use.
7. **Maintenance**
   * Continuously fix issues, update features, and improve the software

Software Requirement

37.LAB EXERCISE: Write a requirementspecification for a simple library managementsystem.

**Library Management System - Requirements Specification (Short Version)**

**1. Functional Requirements**

1. **User Management**:
   * Admin can add, update, or delete member profiles.
   * Members can register and update their profiles.
2. **Book Management**:
   * Admin can add, update, or delete books.
   * Members can search books by title, author, or genre.
3. **Book Borrowing & Returning**:
   * Members can borrow and return books.
   * System tracks due dates and calculates fines for overdue books.
4. **Transaction History**:
   * System records and displays all transactions (borrow, return, fines).
5. **Search & Filter**:
   * Members can search for books by various criteria.

**2. Non-Functional Requirements**

1. **Performance**:
   * Fast, responsive searches and updates.
2. **Security**:
   * Secure storage of member and transaction data.
3. **Usability**:
   * Simple, intuitive interface.
4. **Scalability**:
   * Can handle increasing users and data.
5. **Backup**:
   * Regular automatic backups of data.

**3. System Requirements**

* **Hardware**: Desktop computers, tablets with internet.
* **Software**: Web browser compatibility, database (e.g., MySQL).

**4. User Roles**

1. **Admin**: Manage users, books, and transactions.
2. **Members**: Borrow/return books, view history, pay fines.

38.THEORY EXERCISE: Why is the requirement analysis phase critical in software development?

Software Analysis

**Importance of Requirement Analysis in Software Development**

1. **Clarifies Expectations**:
   * Ensures that developers understand exactly what the client or user needs, reducing the chances of misunderstandings.
2. **Defines Scope**:
   * Helps in defining what the software will and won’t do, preventing scope creep (when the project grows beyond its original goals).
3. **Foundation for Design**:
   * The information gathered during requirement analysis serves as the base for the software design phase, ensuring the right features are included.
4. **Reduces Costs**:
   * Identifying potential issues early helps in avoiding costly changes during later phases of development.
5. **Improves Quality**:
   * With clear and precise requirements, developers can create software that better meets user needs and functions correctly.

39.LAB EXERCISE: Perform a functional analysis for an online shopping system.

**Functional Analysis for an Online Shopping System (Short Version)**

1. **User Management**:
   * **Registration/Login**: Users can sign up and log in securely.
   * **Profile Management**: Users can update personal details and payment information.
2. **Product Catalog**:
   * **Product Listing/Search**: Display products with details; users can search and filter.
   * **Product Details**: Show detailed product info, including images and reviews.
3. **Shopping Cart**:
   * **Add/Remove Items**: Users can add or remove items from their cart.
   * **Price Calculation**: Display total cost, including taxes and discounts.
4. **Order Management**:
   * **Checkout**: Users can review and place orders with shipping and payment info.
   * **Order Confirmation**: Users receive confirmation emails and order details.
5. **Payment Processing**:
   * **Payment Gateway**: Secure payment through credit cards, PayPal, etc.
   * **Payment Confirmation**: Confirmation sent after successful payment.
6. **Shipping & Delivery**:
   * **Shipping Options**: Users choose delivery methods.
   * **Track Order**: Users can track their orders.
7. **Customer Support**:
   * **Live Chat/Help Desk**: Provide customer support for inquiries and returns.
   * **FAQs**: Offer self-service support with common questions.
8. **Reviews & Ratings**:
   * **Product Reviews**: Users can leave reviews and rate products.
9. **Admin Panel**:
   * **Product/Order Management**: Admins can manage products, orders, and customer data.
   * **Reports & Analytics**: Admins can view sales and user behavior data.

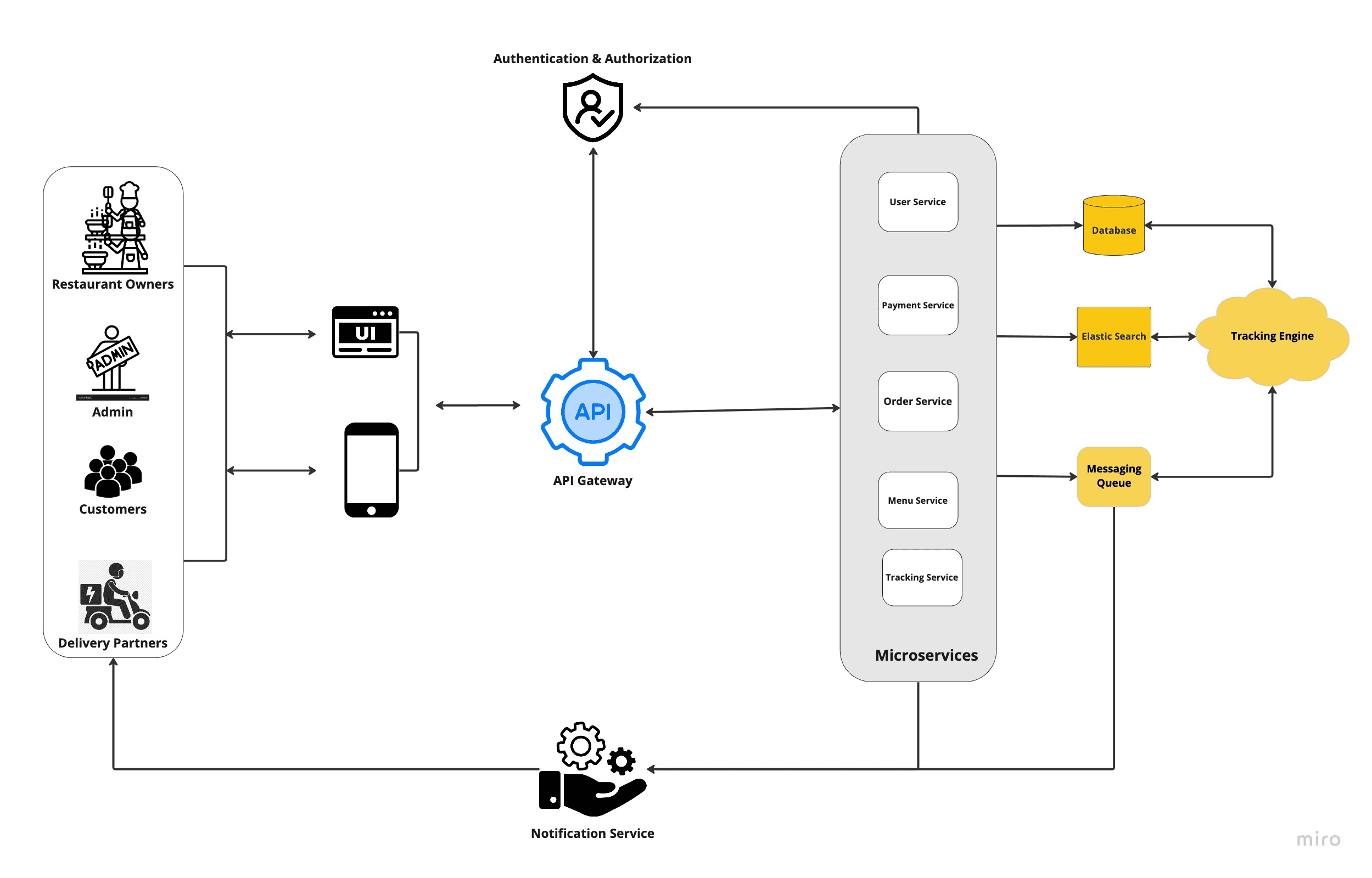
40.THEORY EXERCISE: What is the role of software analysis in the development process?

**Role of Software Analysis in the Development Process**

1. **Understanding Requirements**:
   * Helps gather and clarify user and business needs to ensure the software will meet expectations.
2. **Defining Scope**:
   * Establishes the boundaries of the project, ensuring developers know what features are included and what is not.
3. **Identifying Problems**:
   * Analyzes existing systems or processes to identify inefficiencies and areas for improvement.
4. **Feasibility Check**:
   * Assesses whether the software can be built within the given constraints (time, budget, technology).
5. **Foundation for Design**:
   * Provides the necessary information for the design phase, ensuring the software is built to solve real problems effectively.

System Design

41.LAB EXERCISE: Design a basic system architecture for a food delivery app.



42.THEORY EXERCISE: What are the key elements of system design?

**Key Elements of System Design**

1. **Functional Requirements**: Defines system tasks and features.
2. **Non-Functional Requirements**: Specifies performance, security, and scalability.
3. **System Architecture**: Overall structure and components of the system.
4. **Database Design**: Organizes and stores data efficiently.
5. **User Interface Design**: Focuses on user experience and interaction.
6. **Security Design**: Ensures protection against threats.
7. **Scalability & Performance**: Handles increased load and efficiency.
8. **Integration**: Enables interaction with external systems.
9. **Testing & Validation**: Verifies the system works as intended.
10. **Deployment & Maintenance**: Ensures smooth launch and ongoing updates.

43.LAB EXERCISE: Develop test cases for a simple calculator program.

**Test Cases for a Simple Calculator Program**

**1. Addition Test Cases**

| **Test Case ID** | **Test Case Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC\_01 | Test for adding two positive numbers | 5 + 3 | 8 |  |  |
| TC\_02 | Test for adding a positive and a negative number | 5 + (-3) | 2 |  |  |
| TC\_03 | Test for adding two negative numbers | -5 + (-3) | -8 |  |  |
| TC\_04 | Test for adding zero to a number | 0 + 5 | 5 |  |  |

**2. Subtraction Test Cases**

| **Test Case ID** | **Test Case Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC\_05 | Test for subtracting two positive numbers | 5 - 3 | 2 |  |  |
| TC\_06 | Test for subtracting a larger number from a smaller number | 3 - 5 | -2 |  |  |
| TC\_07 | Test for subtracting a negative number from a positive number | 5 - (-3) | 8 |  |  |
| TC\_08 | Test for subtracting zero from a number | 5 - 0 | 5 |  |  |

**4. Division Test Cases**

| **Test Case ID** | **Test Case Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC\_13 | Test for dividing two positive numbers | 6 ÷ 3 | 2 |  |  |
| TC\_14 | Test for dividing a positive by a negative number | 6 ÷ (-3) | -2 |  |  |
| TC\_15 | Test for dividing two negative numbers | -6 ÷ (-3) | 2 |  |  |
| TC\_16 | Test for dividing by zero | 5 ÷ 0 | Error/Infinity |  |  |
|  | | | | | |

**5. Edge Cases and Additional Tests**

| **Test Case ID** | **Test Case Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC\_17 | Test for large numbers | 1000000 + 1000000 | 2000000 |  |  |
| TC\_18 | Test for small numbers (decimals) | 0.1 + 0.2 | 0.3 |  |  |
| TC\_19 | Test for floating-point precision | 0.333 + 0.666 | 0.999 |  |  |

44.THEORY EXERCISE: Why is software testing important?

**Why Software Testing is Important**

1. **Ensures Quality**:
   * Makes sure the software works as expected.
2. **Finds Bugs**:
   * Detects errors or problems before users encounter them.
3. **Improves User Experience**:
   * Ensures the software is easy to use and reliable.
4. **Saves Money**:
   * Fixing issues early is cheaper than after release.
5. **Increases Security**:
   * Identifies security flaws to protect user data.
6. **Meets Standards**:
   * Ensures the software follows industry rules and regulations.
7. **Improves Performance**:
   * Confirms the software works well under heavy use.
8. **Builds Trust**:
   * Well-tested software makes users feel confident using it.

Maintenance

45.LAB EXERCISE: Document a real-world case where a software application required

Criticalmaintenance

**Real-World Case: CrowdStrike Antivirus Update Disruption (2024)**

**The Problem:**  
In July 2024, an update to CrowdStrike's Falcon antivirus software inadvertently caused widespread IT outages, particularly in healthcare systems worldwide. The update led to critical failures, including:

* System crashes displaying the "blue screen of death" on Windows computers.
* Disruption of essential services like electronic medical records (EMRs) and communication tools.

**Why Maintenance Was Critical:**  
The healthcare sector relies heavily on uninterrupted IT systems for patient care, including scheduling, diagnostics, and emergency response. The disruption caused significant operational delays, risking patient safety and requiring urgent resolution.

**The Solution:**  
CrowdStrike implemented emergency maintenance by:

1. Rolling back the faulty update.
2. Issuing a hotfix to address the root cause.
3. Coordinating with affected institutions to restore normal operations.

**Outcome:**  
While the disruption highlighted vulnerabilities in software deployment, the swift maintenance response minimized prolonged impacts and restored trust in the affected systems.

This case emphasizes the importance of thorough testing and rapid remediation in critical sectors.

46.THEORY EXERCISE: What types of software maintenance are there?

The types of software maintenance are:

1. **Corrective Maintenance**
2. **Adaptive Maintenance**
3. **Perfective Maintenance**
4. **Preventive Maintenance**

Development47THEORY EXERCISE: What are the key differences between web and desktop applications?

|  |  |
| --- | --- |
| **DESKTOP APPS** | **WEB APPS** |
| They require installation on the computer to run. | They are accessible through web browsers and do not require installation. |
| Generally, desktop apps do not require an internet connection to run. | Web apps cannot run without an internet connection. |
| They are accessible only in the machine they are installed in. | They are accessible from anywhere and through any device with an internet connection and a web browser. |
| They take space on the hard drive of the local computer. | They take up space on the remote server. |
| Deployment and updating are to be done individually on each computer. | Deployment and updating are done only on the server. |
| They have strict hardware requirements for proper functionality. | Web apps are hardware-independent and just require a web browser and internet connection to function. |
| As they are confined to a device and single or limited users, they are highly secure. | As web apps are accessible to all through the internet, they are less secure than desktop apps. |
| Generally, they are faster than web applications. | Generally, they are slower than desktop applications. |

48.THEORY EXERCISE: What are the advantages of using web applications over desktop applications?

**Advantages of Web Applications Over Desktop Applications**

1. **Accessible Anywhere**: Works on any device with an internet connection.
2. **No Installation**: No need for installation on individual devices.
3. **Automatic Updates**: Updates are done automatically on the server.
4. **Cross-Platform**: Works on any operating system with a browser.
5. **Centralized Data**: Easier to manage and secure data on a server.
6. **Cost-Effective**: Cheaper to maintain and deploy.
7. **Real-Time Collaboration**: Enables collaboration from different locations.
8. **Scalable**: Easy to scale by upgrading server resources.

49..THEORY EXERCISE: What role does UI/UX design play in application development?

**Role of UI/UX Design in Application Development**

1. **Improves User Experience**: Makes the app easy and enjoyable to use.
2. **Increases Engagement**: Attracts users and encourages interaction.
3. **Boosts Usability**: Simplifies tasks and improves efficiency.
4. **Builds Trust**: Creates a professional and reliable image.
5. **Enhances Accessibility**: Makes the app usable for all users, including those with disabilities.
6. **Reduces Costs**: Identifies issues early, saving on future revisions.
7. **Differentiates the Product**: Makes the app stand out from competitors.

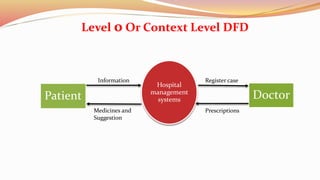
50. Mobile Application

THEORY EXERCISE: What are the differences between native and hybrid mobile apps?

|  |  |
| --- | --- |
| **Native Apps** | **Hybrid Apps** |
| iOS- Swift, Objective C Android- Kotlin, Java | HTML, CSS and JavaScript Frameworks: PhoneGap, Ionic |
| It is platform-specific. Separate app development for individual platforms | It is for multiple operating systems. Develop a codebase and share it over all the platforms (iOS, Android, Windows) |
| As they are developed for individual platforms, the [cost of app development](https://www.spaceotechnologies.com/blog/cost-of-app-development/) is higher. You also need separate infrastructure for maintenance of the application | You are using only one codebase across platforms, which means it is cost-effective. A single framework and resource can help you maintain the application |
| Native apps are rich in performance as they are built using the design and development guidelines for the platform | Hybrid apps offer dependable performance. However, they are enclosed in native containers, so may not offer as high speeds as expected |

51. DFD (Data Flow Diagram)

LAB EXERCISE: Create a DFD for a hospital managementsystem.



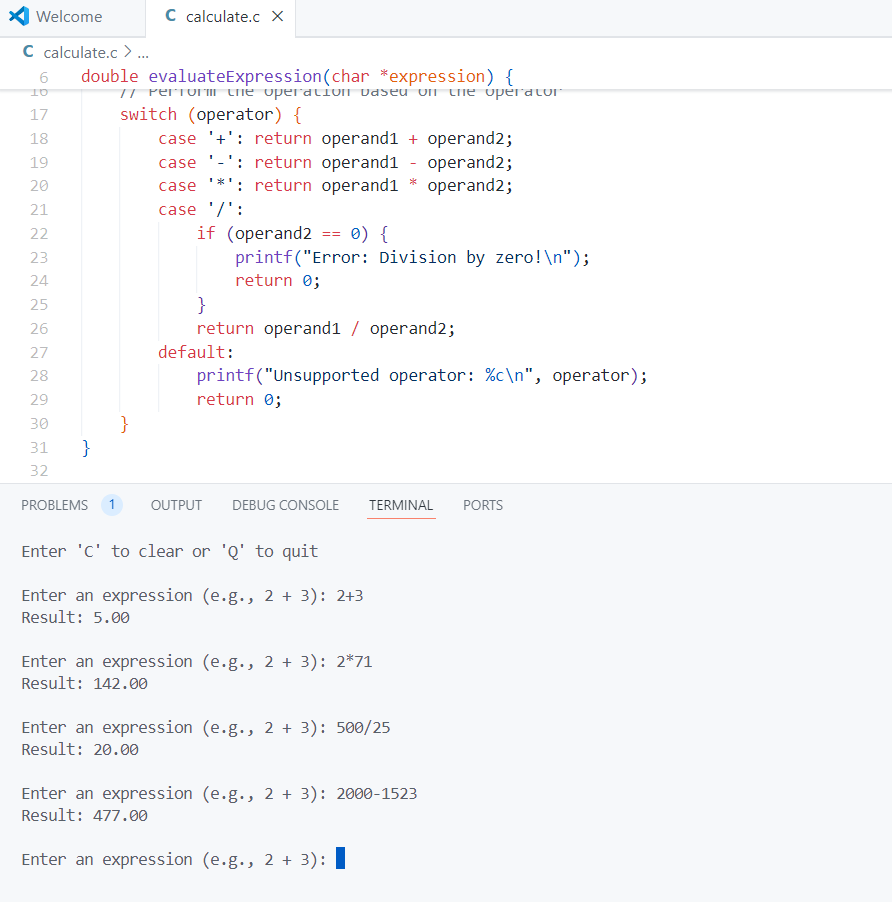
THEORY EXERCISE: What is the significance of DFDs in system analysis?

**Significance of DFDs in System Analysis**

1. **Clear Visualization**: Shows data flow and system components.
2. **Simplifies Communication**: Makes system understanding easier for stakeholders.
3. **Identifies Inefficiencies**: Highlights bottlenecks and redundant processes.
4. **Supports Requirements Gathering**: Helps define functional requirements.
5. **Assists System Design**: Provides a blueprint for system structure.
6. **Ensures Data Integrity**: Ensures correct data flow and reduces errors.
7. **Easily Updatable**: Can be modified as the system evolves.

52. Desktop Application

LAB EXERCISE: Build a simple desktop calculator application using a GUI librarssy.



THEORY EXERCISE: What are the pros and cons of desktop applications compared to

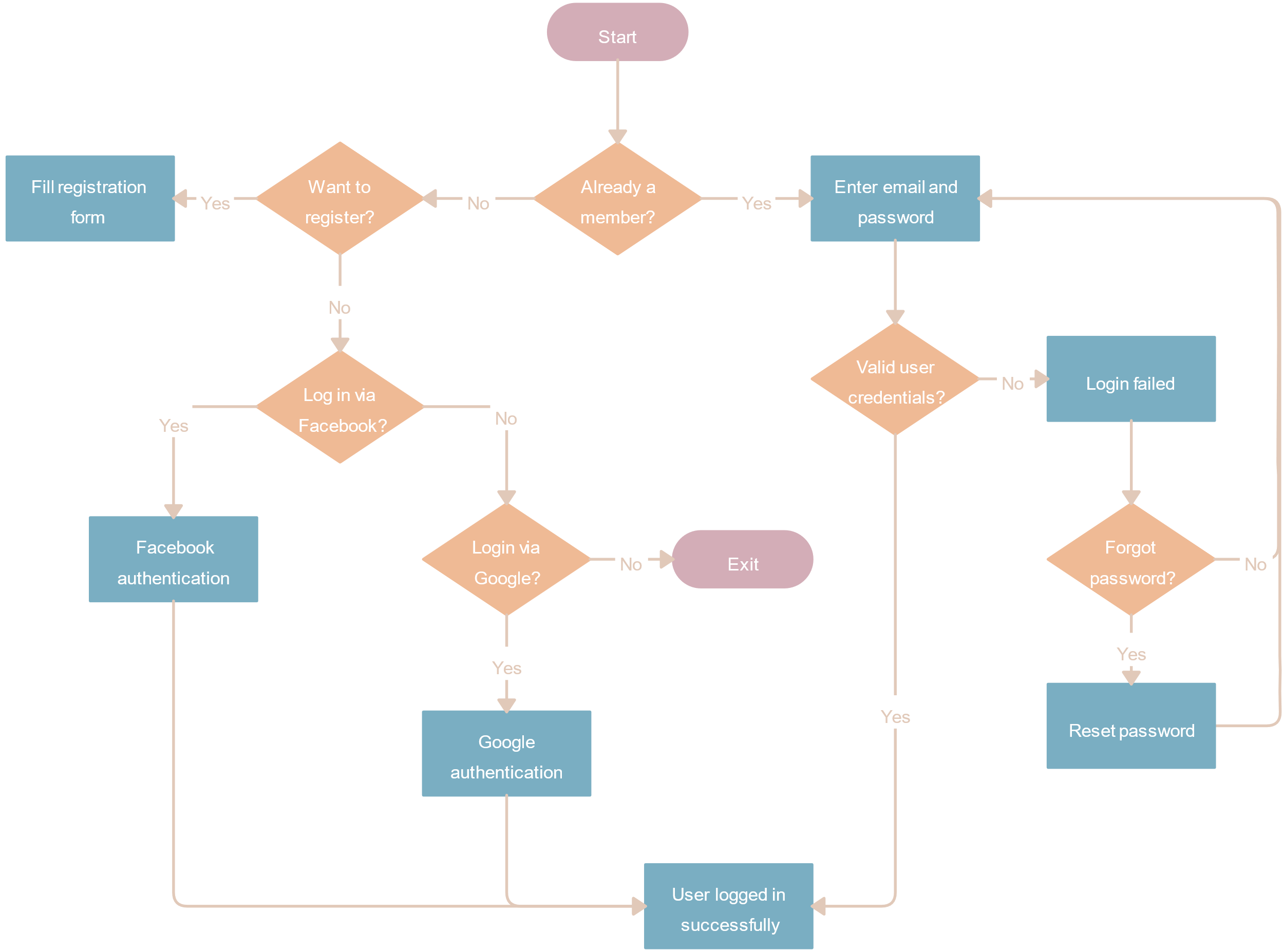
webapplications?

|  |  |
| --- | --- |
| **Pros of Desktop Applications** | **Cons of Desktop Applications** |
| Works without an internet connection | May require specific OS compatibility |
| Faster due to local processing | Users must install and update manually. |
| Better access to system resources (e.g., GPU, peripherals). | Restricted to the device it’s installed on. |
| Less vulnerable to web-based attacks. | Takes up local disk space. |
| More powerful and capable for complex tasks (e.g., video editing, gaming). | Harder to share and deploy updates. |

|  |  |
| --- | --- |
| **Pros of Web Applications** | **Cons of Web Applications** |
| Can be accessed from any device with a browser and internet. | Requires a stable connection to function. |
| Works across operating systems. | Limited by internet speed and server response. |
| Users always get the latest version. | More exposed to online threats like hacking. |
| Minimal or no local storage required. | Restricted access to system resources |
| Easier to scale for large user bases. | Performance may vary across browsers. |

54. Flow Chart

LAB EXERCISE: Draw a flowchart representing the logic of a basic online registration system.



THEORY EXERCISE: How do flowcharts help in programming and system design? Flowcharts help in programming and system design by:

1. **Simplifying Complex Ideas**: They break down complex processes into simple steps.
2. **Improving Clarity**: They make the logic easier to understand.
3. **Finding Errors Early**: Help spot mistakes before coding starts.
4. **Better Communication**: Non-technical people can easily understand them.
5. **Teamwork**: They help teams work together more efficiently.
6. **Guiding Development**: Provide a clear plan for writing code.
7. **Better Design**: Help improve and optimize the system.

-**Thank you**