Module 3 – Fundamentals of IT

* **What is a Program?**

LAB EXERCISE:

-Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.

🡪 PYTHON:

print ("Hello, World!")

🡪HTML:

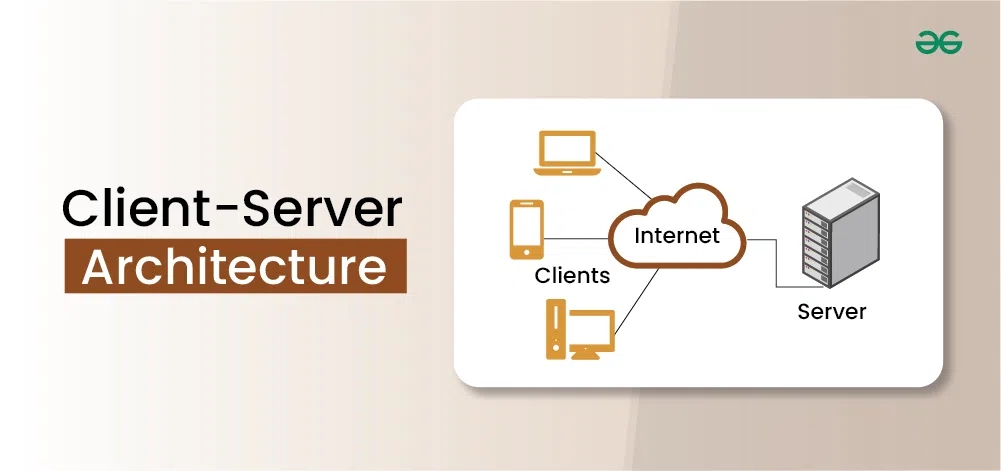
<html>  
<head>  
 <title>Hello World Page</title>  
</head>  
<body>  
 <h1>Hello World! </h1>  
</body>  
</html>

* **World Wide Web & How Internet Works**

LAB EXERCISE:

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

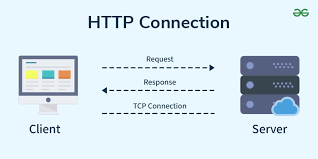
🡪 In the client-server model, the client initiates requests for information or services, while the server responds by processing those requests and providing the requested resources or performing actions. Data is transmitted over the internet through a series of steps, involving protocols like HTTP/HTTPS, DNS, and packet switching.



* **Network Layers on Client and Server**

LAB EXERCISE:

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

🡪 

* **Types of Internet Connections**

LAB EXERCISE:

-Research different types of internet connections (e.g., broadband, Fiber, satellite) and list their pros and cons.

🡪 Broadband Internet Connections:

* [**Fiber Optic**](https://www.google.com/search?sca_esv=d06f5a1c24924b00&cs=1&sxsrf=AE3TifPnD3sNZfGk2XfRMOHFBZp-xel5sQ%3A1753422712493&q=Fiber+Optic&sa=X&ved=2ahUKEwjPoPnOqNeOAxWZSWwGHbyNFcsQxccNegQIEBAC&mstk=AUtExfAceeqWDVQHHg94K1K5PY8emaIedB4cE2keh27E16vJ2HIzv21Tp3Yc70jlKisW4uxv2rcsvN2YPa60THTjGt-hDi2PoLYRjYI9x9BBc_BkcY7UU5F9BvRNJ3RbqHP779BWOPY-10ZDrOCMvaPB4O806IT_2b-4Hbha-pJuCPi6HNMVsD4_QHlkCdlPAptLerVCFqFmrzZJJuq2_VxbgMtwzpgkW8d5SkkkR221ZXa0swjVURQoNlQG2dRhwzk8A4CBoJwkmrYsj9l6aYivrMo0&csui=3)**:**
  + **Pros:** Fastest speeds, symmetrical upload and download speeds (ideal for video conferencing and gaming), reliable, less susceptible to interference.
  + **Cons:** Can be more expensive, availability may be limited in some areas.
* [**Cable**](https://www.google.com/search?sca_esv=d06f5a1c24924b00&cs=1&sxsrf=AE3TifPnD3sNZfGk2XfRMOHFBZp-xel5sQ%3A1753422712493&q=Cable&sa=X&ved=2ahUKEwjPoPnOqNeOAxWZSWwGHbyNFcsQxccNegQIOxAC&mstk=AUtExfAceeqWDVQHHg94K1K5PY8emaIedB4cE2keh27E16vJ2HIzv21Tp3Yc70jlKisW4uxv2rcsvN2YPa60THTjGt-hDi2PoLYRjYI9x9BBc_BkcY7UU5F9BvRNJ3RbqHP779BWOPY-10ZDrOCMvaPB4O806IT_2b-4Hbha-pJuCPi6HNMVsD4_QHlkCdlPAptLerVCFqFmrzZJJuq2_VxbgMtwzpgkW8d5SkkkR221ZXa0swjVURQoNlQG2dRhwzk8A4CBoJwkmrYsj9l6aYivrMo0&csui=3)**:**
  + **Pros:** Widely available, relatively fast speeds, often bundled with other services.
  + **Cons:** Speeds can be affected by the number of users in an area, can be slower than fiber.
* [**Satellite**](https://www.google.com/search?sca_esv=d06f5a1c24924b00&cs=1&sxsrf=AE3TifPnD3sNZfGk2XfRMOHFBZp-xel5sQ%3A1753422712493&q=Satellite&sa=X&ved=2ahUKEwjPoPnOqNeOAxWZSWwGHbyNFcsQxccNegQIQBAC&mstk=AUtExfAceeqWDVQHHg94K1K5PY8emaIedB4cE2keh27E16vJ2HIzv21Tp3Yc70jlKisW4uxv2rcsvN2YPa60THTjGt-hDi2PoLYRjYI9x9BBc_BkcY7UU5F9BvRNJ3RbqHP779BWOPY-10ZDrOCMvaPB4O806IT_2b-4Hbha-pJuCPi6HNMVsD4_QHlkCdlPAptLerVCFqFmrzZJJuq2_VxbgMtwzpgkW8d5SkkkR221ZXa0swjVURQoNlQG2dRhwzk8A4CBoJwkmrYsj9l6aYivrMo0&csui=3)**:**
  + **Pros:** Available in remote areas where other connections are not, suitable for mobile users.
  + **Cons:** Significantly slower speeds than other broadband options, higher latency (delay in data transmission), affected by weather conditions.
* **Protocols**

LAB EXERCISE:

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

🡪Simulating HTTP Requests:

* **GET Request:** To retrieve the content of a URL, use curl followed by the URL.

Code

curl https://www.example.com

* **POST Request:** To send data to a server, use the -X POST or -d option.

Code

curl -X POST -d "param1=value1&param2=value2" https://www.example.com/api/data

* **Application Security**

LAB EXERCISE:

-Identify and explain three common application security vulnerabilities. Suggest possible solutions.

🡪 Three common application security vulnerabilities are: Injection flaws, Broken Authentication, and Sensitive Data Exposure.

🡪Injection flaws:

Injection flaws, like SQL injection, occur when untrusted data is sent to an interpreter as part of a command or query.

**Solution:**

* **Parameterized Queries:** Use parameterized queries or prepared statements to separate code from data, preventing the injection of malicious code.

🡪Broken Authentication:

Broken Authentication allows attackers to bypass or compromise authentication mechanisms, gaining unauthorized access.

**Solution:**

* **Strong Authentication:** Enforce strong password policies, including complexity requirements and regular password changes.

🡪 Sensitive Data:

Sensitive Data Exposure involves the disclosure of sensitive information, such as passwords or credit card numbers, due to inadequate protection measures.

**Solution:**

* **Secure Storage:** Store sensitive data securely, following industry best practices for data storage and access control.

- Identify and classify 5 applications you use daily as either system software or application software.

🡪

| **Application** | **Type** | **Classification** | **Explanation** |
| --- | --- | --- | --- |
| **Google Chrome** | Web browser | Application Software | It allows users to browse the internet, making it a user-facing application. |
| **Microsoft Word** | Word processor | Application Software | Used for creating and editing documents, serving a specific user task. |
| **Windows 11** | Operating system | System Software | Manages hardware and provides a platform for applications to run. |
| **Spotify** | Music streaming app | Application Software | Enables users to stream music and podcasts, a specific user-oriented function. |
| **Device Drivers** (e.g., printer driver) | Hardware interface | System Software | These allow the operating system to communicate with hardware components. |

* **Software Architecture**

LAB EXERCISE:

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

🡪

A diagram of a computer

AI-generated content may be incorrect.

* **Layers in Software Architecture**

LAB EXERCISE:

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

🡪 **1. Presentation Layer (Client Tier)**

**Functionality:**

* Provides the **user interface (UI)**.
* Sends user input (like search queries or login info) to the backend.
* Displays results (book listings, order confirmations, etc.).

**2. Business Logic Layer (Application Tier)**

**Functionality:**

* Processes all business rules and operations.
* Validates user input (e.g., login credentials, payment info).
* Manages sessions, shopping carts, and checkout workflows.

**3. Data Access Layer (Data Tier)**

**Functionality:**

* Manages direct access to the **database**.
* Performs **CRUD operations** (Create, Read, Update, Delete).
* Converts data between application objects and database records.
* **Software Environments**

LAB EXERCISE:

-Explore different types of software environments (development, testing, production). Set up a basic environment in a virtual machine.

* **Development Environment:**

This is where developers write, debug, and build software. It is characterized by flexibility, allowing for rapid iteration and experimentation. Tools like Integrated Development Environments (IDEs).

**Testing Environment:**

Also known as Quality Assurance (QA) or Staging, this environment is used to test the software before release. Various types of testing, such as unit, integration, functional, and performance testing, occur here.

* **Production Environment:**

This is the live environment where the software is deployed and accessed by end-users. It is highly stable, secure, and optimized for performance and availability.

Setting Up a Basic Environment in a Virtual Machine:

A virtual machine (VM) provides an isolated and reproducible environment suitable for development and testing.

* **Install Virtualization Software:**

Choose a hypervisor like VirtualBox or VMware Workstation/Fusion.

* **Download an Operating System ISO:**

Obtain an ISO image of your desired operating system (e.g., Ubuntu Server, Windows Server) for the guest VM.

* **Create a New Virtual Machine:**
  + Open your virtualization software and create a new VM.
  + Allocate RAM and CPU resources based on your needs.
  + Create a virtual hard disk for the guest OS installation.
  + Mount the downloaded ISO image as a virtual CD/DVD drive.
* **Install the Guest Operating System:**

Start the VM and follow the on-screen instructions to install the chosen operating system.

* **Install Development/Testing Tools:**

Once the OS is installed, install necessary software like:

* + **Development:** IDEs (VS Code, IntelliJ), programming language runtimes (Node.js, Python), version control (Git).
  + **Testing:** Testing frameworks, automation tools, and any specific software required for your application's dependencies.
* **Configure Networking:**

Set up network adapters within the VM to allow communication with the host machine or external networks as needed (e.g., Bridged Adapter for direct network access, NAT for shared network access).

* **Source Code**

LAB EXERCISE:

- Write and upload your first source code file to GitHub.

🡪Create a file named hello.html with the following content

<html>  
<head>  
 <title>My First HTML Page</title>  
</head>  
<body>  
  
 <h1>Welcome to My Page! </h1>  
</body>  
</html>Save this file on your computer.

**Step 2: Create a GitHub Repository**

1. Go to <https://github.com> and log in.
2. Click the **"+"** icon at the top right > **"New repository"**.
3. Fill in:
   * **Repository name:** first code-upload
   * **Description:** (Optional) “My first source code on GitHub”
   * Choose **Public** or **Private**

Click **Create repository**.

**Step 3: Upload Code to GitHub**

**Option A: Upload via GitHub Website (Easiest)**

1. Go to your new repository.
2. Click **"Add file" > "Upload files"**
3. Drag and drop your hello.html file.
4. Click **"Commit changes"**

**Option B: Upload Using Git Command Line (if you have Git installed)**

**1. Open Terminal or Command Prompt**

# Set up Git (only once)

git config --global user.name "Your Name"

git config --global user. Email "you@example.com"

git init

git add hello.html

git commit -m "First commit: hello world program"

**2. Connect to GitHub Repository**

git remote add origin https://github.com/your-username/first-code-upload.git

git branch -M main

git push -u origin main

Make sure to replace your-username with your GitHub username.

**Step 4: Verify Upload**

Go to your GitHub repository page. You should now see hello.html uploaded!

* **GitHub and Introductions**

LAB EXERCISE:

Create a GitHub repository and document how to commit and push code changes.

🡪

**Create a GitHub Repository**

1. Go to <https://github.com> and log in.
2. Click the **+ (plus)** sign at the top-right corner → **New repository**.
3. Fill in:
   * **Repository name:** my-first repo (or anything you prefer)
   * **Description:** (Optional) “A sample repository for code push example”
   * Visibility: Public or Private
4. Click **Create repository**

🡪Whenever you make changes to your files, follow these steps:

# Stage modified files

git add .

# Commit the changes

git commit -m "Describe your change here"

# Push to GitHub

git push

* **Student Account in GitHub**

LAB EXERCISE:

-Create a student account on GitHub and collaborate on a small project with a classmate.

🡪

* **Types of Software**

LAB EXERCISE:

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

🡪 **1. System Software**

These manage and control computer hardware, enabling application software to function.

| **Software** | **Description** |
| --- | --- |
| **Microsoft Windows / Linux / macOS** | Operating systems that manage hardware and software resources. |
| **Device Drivers** | Help the OS communicate with hardware devices like printers, keyboards, etc. |

🡪 **2. Application Software**

These are programs designed to perform specific user tasks.

| **Software** | **Description** |
| --- | --- |
| **Microsoft Word / Google Docs** | Word processing software. |
| **Google Chrome / Mozilla Firefox** | Web browsers for internet access. |
| **MS Excel / Google Sheets** | Spreadsheet applications. |
| **Adobe Photoshop / Canva** | Graphic design and image editing. |

🡪 **3. Utility Software**

These help in maintaining, analyzing, and optimizing performance.

| **Software** | **Description** |
| --- | --- |
| **Antivirus Software (e.g., Avast, Windows Defender)** | Protects against malware and viruses. |
| **Ccleaner** | Cleans unnecessary files and optimizes system performance. |
|  |  |
| **Backup Tools** (e.g. Windows Backup) | Create and manage data backups. |
| **Disk Cleanup / Defragmenter** | Improve disk performance and free up space |

Top of Form

* **GIT and GITHUB TrainingBottom of Form**

LAB EXERCISE:

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

🡪1. Initialize a Local Repository:

--Create a new directory for your project:

---Initialize a Git repository within this directory:

2. Create and Commit Files:

-Commit the changes to the repository.

3. Branching:

--Create a new branch for a feature or bug fix (e.g., feature-branch):

--Make changes on this new branch (e.g., add content to README.md):

--Commit the changes on feature-branch.

4. Merging:

--Switch back to the main branch.

--Merge the changes from feature-branch into main:

🡪5. Cloning (Simulated Remote Repository):

--Navigate to a different location outside your current project directory

--Clone your existing local repository as if it were a remote one:

Navigate into the cloned repository.

* **Application Software**

LAB EXERCISE:

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

🡪Productivity Software:

Business Software:

Communication and collaboration software:

Project management software:

Graphic Design Software:

🡪 How Application Software Improves Productivity:

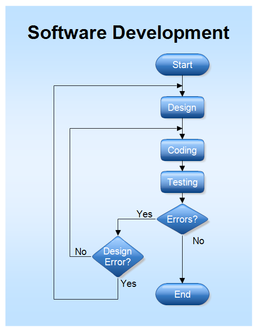
Application software automates repetitive tasks, freeing up time for more complex and strategic work.

🡺 Software helps organize, store, and analyze data, leading to better insights and informed decisions.

* **Software Development Process**

LAB EXERCISE:

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



* **Software Requirement**

LAB EXERCISE

-Write a requirement specification for a simple library management system.

* **Purpose:** To outline the functional and non-functional requirements for a basic library management system designed to streamline core library operations.
* **Scope:** The system will manage books, members, and borrowing/returning processes. It will provide interfaces for librarians and basic access for members.
* **Intended Audience:** Librarians, library staff, and system developers.
* **Software Analysis**

LAB EXERCISE:

- Perform a functional analysis for an online shopping system.

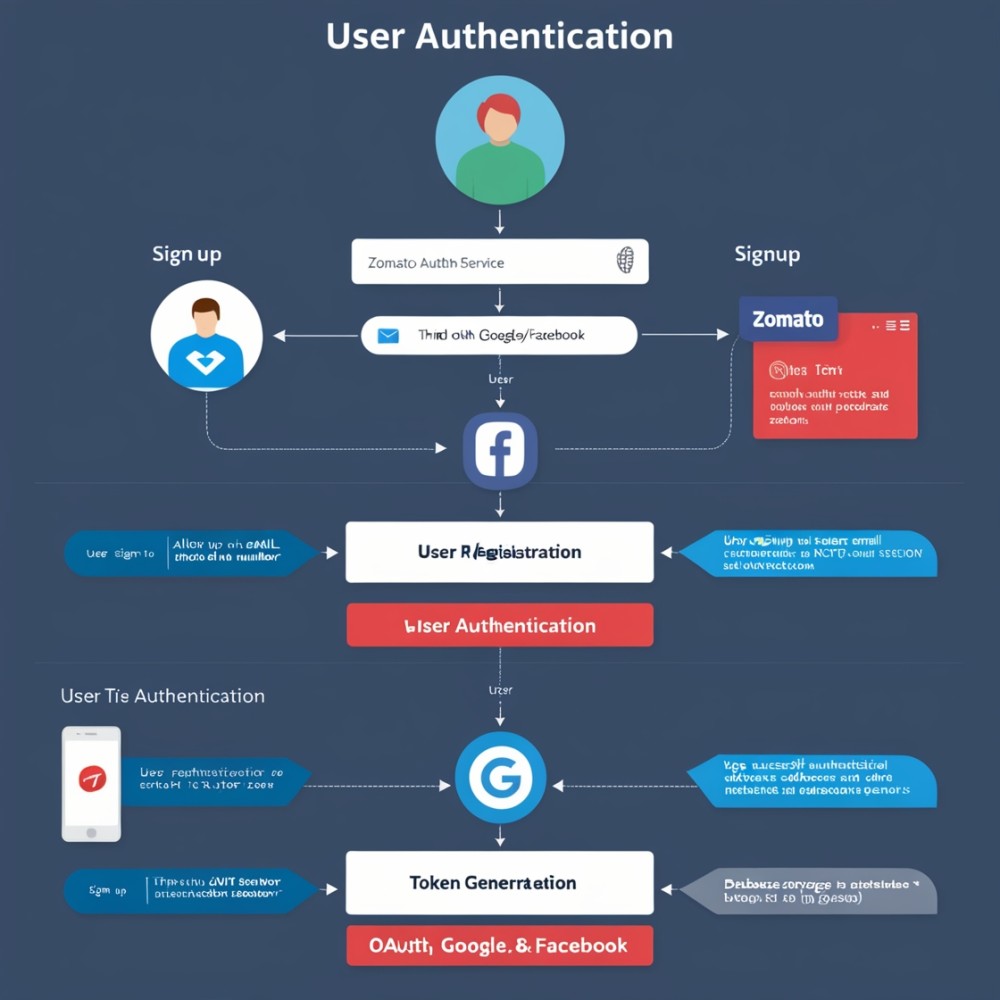
🡪 A functional analysis of an online shopping system identifies and describes the specific actions (functions) the system must perform from the user's and administrator's perspectives.

🡪This includes tasks like user registration, product searching and browsing, shopping cart management, checkout, order tracking, and administrative functions like product and user management.

* **System Design**

LAB EXERCISE:

- Design a basic system architecture for a food delivery app.

🡪 

* **Software Testing**

LAB EXERCISE:

-Develop test cases for a simple calculator program.

* 🡪 Check the calculator if it starts by on button. If it is software-based calculator then check if it starts via specific means like from searching for calculator in search bar and then executing application. Or by accessing menu item in the Windows.
* Check if the calculator window maximizes to certain window size.
* Check the if the calculator closes when the close button is pressed or if the exit menu is clicked from file > exit option.
* Check if the help document is accessed from Help > Documentation.
* Check if the calculator allows copy and paste functionality.
* Check if the calculator has any specific preferences.
* Check if all the numbers are working (0 to 9)
* Check if the arithmetic keys ( +, -, \*, %, /) are working.
* Check if the clear key is working.
* Check if the brackets keys are working.
* Check if the sum or equal key is working.
* Check if the square and square root key is working.
* **Maintenance**

LAB EXERCISE:

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

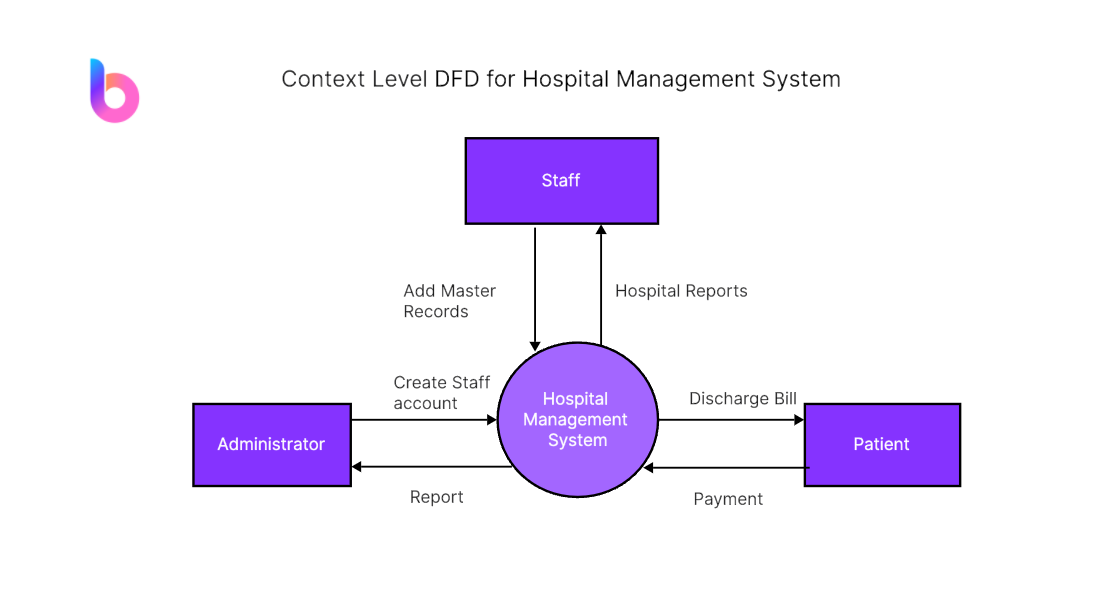
🡪 A real-world example of a software application requiring critical maintenance is a popular e-commerce website experiencing a surge in traffic during a holiday sale, leading to slow loading times and frequent crashes.

🡪This necessitates immediate corrective and adaptive maintenance to handle the increased load and ensure a smooth shopping experience for customers.

* **DFD (Data Flow Diagram)**

LAB EXERCISE:

Create a DFD for a hospital management system.

🡪 

* **Desktop Application**

LAB EXERCISE:

Build a simple desktop calculator application using a GUI library.

🡪 **Import the Tkinter module:** This provides the necessary tools for creating GUI elements.

🡪Initialize the main window: Create an instance of Tk () to represent the application window and configure its properties like title and size.

root = Tk ()  
 root. title ("Simple Calculator")  
 root. geometry("270x150")

**🡪Create a display entry:** Use an Entry widget to display the input and results. A String Var can be used to easily manage the text displayed in the entry.

🡪 Define button actions: Create functions to handle button clicks for numbers, operations, equals, and clear.

o press(key): Appends the pressed key to a global expression string and updates the display.

o equal (): Evaluates the expression string using eval () (with error handling) and displays the result. Resets the expression.

o clear (): Clears the expression string and the display.

🡪 Create and place buttons: Instantiate Button widgets for numbers (0-9), operations (+, -, \\*, /), equals, and clear. Assign their respective command functions to be called on click and arrange them using grid layout.

🡪 Run the main event loop: Start the Tkinter event loop to make the GUI interactive.

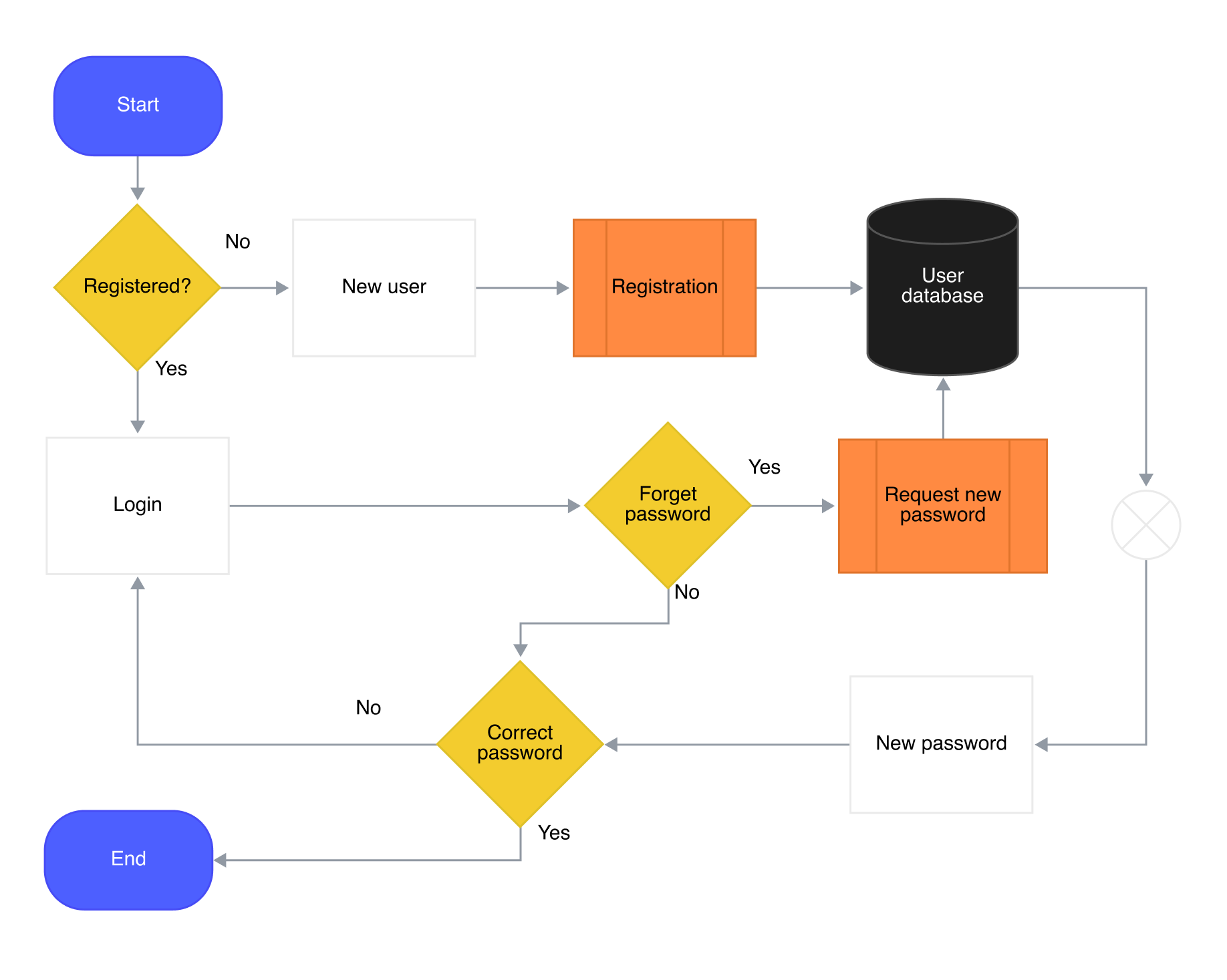
root. mainloop()

**.**

* **Flow Chart**

LAB EXERCISE:

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

🡪 

Bottom of Form