**Module 1 – Foundation**

**LAB EXERCISE:**

🡪Create Git hub account & make repository.

🡪git init

🡪git add .

🡪git commit –m “msg”

🡪git branch –M main

🡪git remote add origin <https://github.com/riddhigodham/>

🡪git push –u origin main

**Module 2 – Fundamentals of World Wide Web**

**LAB EXERCISE:**

🡪 Explain phases of SDLC life cycle.



**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

**1. C Language**

#include <stdio.h>

​

int main() {

printf ("Hello World");

return 0;

}

### ****2. Python Language****

### Print ("Hello World")

World Wide Web & How Internet Works

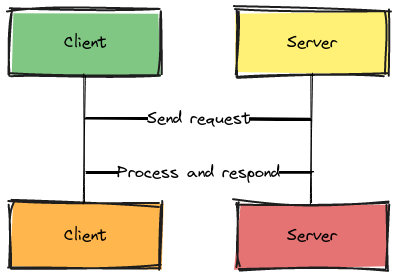
**LAB EXERCISE:**

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

# Diagram of Client-Server Architecture

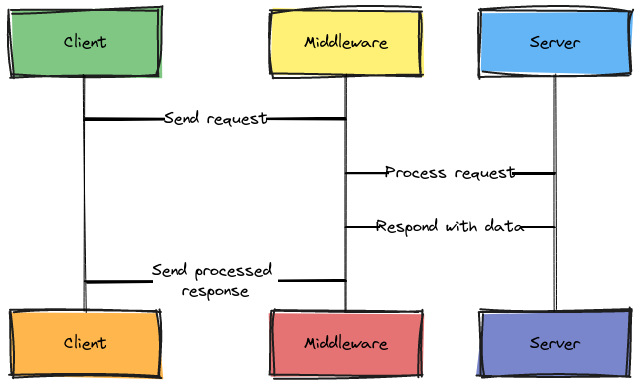
Below are simplified diagrams of the client-server architecture.

1. **Two-Tier Client-Server Model Diagram :**  
   Illustrates direct communication between a client and server, where the client sends a request, and the server processes and responds.



Two tier client server architecture

**2. Three-Tier Client-Server Model Diagram :**  
Adds an intermediary middleware layer, handling business logic or complex database queries.

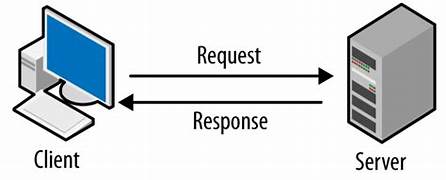


Three tier client server architecture

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.

## **🡪Advantages of Broadband** Connection

### 1. **Better connection**

### 2. **Faster downloads**

## **🡪Disadvantages of Broadband Connection**

### 1. **Higher cost**

### 2. Higher Rates

## 🡪The Pros of Fiber Optic internet

### **Fiber Connections Support Higher Speeds**

### **Available Symmetrical Speeds**

## 🡪The Cons of Fiber Optic Internet

### **Higher Costs**

### **Lack of Widespread Coverage**

## 🡪Advantages of satellite internet

### Wide availability

### Satellite internet reliability

## 🡪Disadvantages of satellite internet

### Data restrictions

### Slow speeds

Protocols

**LAB EXERCISE:**

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

Simulating HTTP Requests:

* **GET Request (fetching content):** To retrieve the content of a web page, use the basic curl command followed by the URL.

curl <https://www.example.com>

Simulating FTP Requests:

* **Listing Directory Contents:** To view the contents of an FTP directory, specify the FTP URL.

Code

curl <ftp://ftp.example.com/>

### Application Security

### LAB EXERCISE:

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

## 1. Structured Query Language (SQL)/Database Queries

This is the **most common area of application vulnerability** specifically due to the use of multiple databases in conjunction with multiple applications.

### 2. Cross-Site Scripting

### Cross-site scripting (XSS) attacks target web applications that fail to properly sanitize user input.

### 3. Software and Data Integrity Failures

Integrity failures occur when an application relies on untrusted software components or fails to validate the integrity of data.

**LAB EXERCISE:**

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

Here's a classification of five common applications, categorized as either system or application software:

System Software:

1. Operating System (e.g., Windows, macOS, Android)

**2. Device Drivers (e.g., printer driver, graphics card driver)**

Application Software:

1. Web Browser (e.g., Google Chrome, Firefox, and Safari)

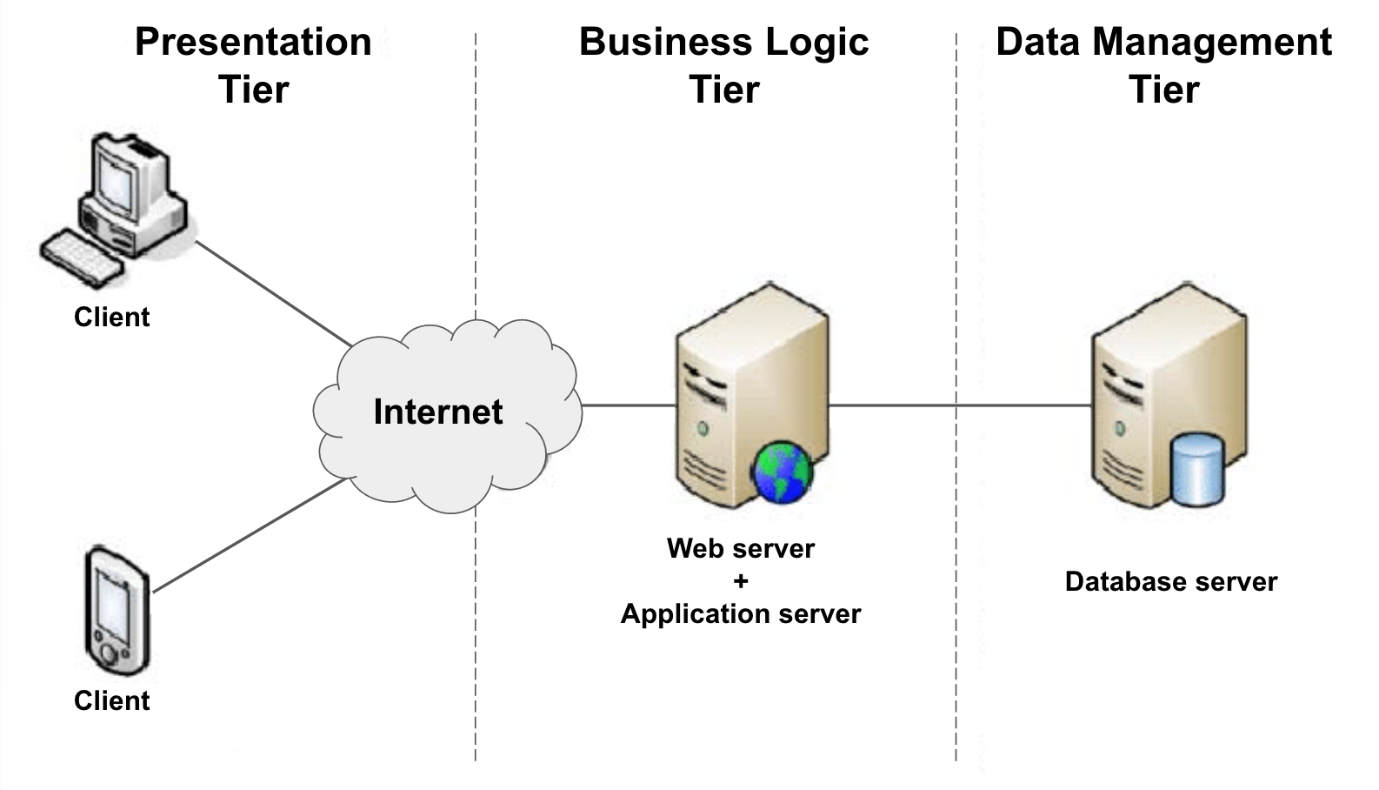
**2. Word Processor (e.g., Microsoft Word, Google Docs)**

**3. Media Player (e.g., VLC Media Player, Windows Media Player)**

Software Architecture

**LAB EXERCISE:**

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



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.

* **Presentation Layer**

The presentation layer performs several types of functions such as data translation, compression, encryption/decryption etc.

* Business-Logic

The Business-Logic Layer (BLL) is a component of a software architecture that is responsible for implementing the business logic of an application.

* Data-Access Layer

The Data-Access Layer (DAL) is a component of a software architecture that is responsible for managing the data storage and retrieval of an application.

Software Environments

**LAB EXERCISE:**

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

## 1. Production Environment

This is the final stage where we deploy the software and make it available to all end-users. It’s often called the live environment and hosts the fully functional and stable version of the software.

## 2. Development Environment

**The development environment is where developers write and test code, typically, on a local machine**. Most, if not all, software development processes involve working in a team. Therefore, the development environment serves as the initial stage of the software development process and enables developers to create, modify, and debug their code before integrating it with the work of other team members.

### 3. Automated Testing and Builds

Automated testing involves writing and executing automated tests to verify the functionality, performance, and reliability of the integrated codebase. Further, it helps catch regressions, ensure code quality, and provide confidence in the stability of the software.

The basic process to create and run a[virtual machine (VM)](https://www.serverwatch.com/guides/virtual-machines/) is very straightforward. Just about anyone can do it. Having said that, to leverage the performance of advanced VM operations, technical skills are required.

Source Code

**LAB EXERCISE:**

🡪Write and upload your first source code file to Git hub

🡪To upload your project to GitHub, follow these steps:

1. Create a GitHub repository and clone it to your local machine.
2. Configure Git on your local machine.
3. Add your project files to the local repository.
4. Stage and commit your changes.
5. Push your changes to the remote GitHub repository.

## **Steps to Upload a Project On GitHub**

### Step 1: Create a Repository on GitHub and Copy the Repository Link

### Step 2: Install Git and Clone Your Repository

### Step 3: Navigate to Your Project Folder

### Step 4: Check the Status of your Repository

### Step 5: Add Files to the Staging Area

### Step 6: Commit your Changes

### Step 7: Uploading Changes to Remote Profile

### Step 8: Push Changes to GitHub

Git hub and Introductions

**LAB EXERCISE:**

🡪 Create a Git hub repository and document how to commit and push code changes.

Create a Git hub Repository

1. Go to <https://github.com> and login.

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 change to your files, follow these steps:

# Stage modified files

git add.

# Commit the changes

git Commit –m “Describe your change here”

# Push to git hub

git push

Student Account in Git hub

**LAB EXERCISE:**

🡪Create a student account on Git hub and collaborate on a small project with a classmate.

To create a student account on GitHub and collaborate on a small project with a classmate, follow these steps:

1. Create a GitHub Account:

* Navigate to the GitHub website and sign up for a free account if you do not already have one.
* Choose a unique username and provide your email address and a strong password.

2. Apply for GitHub Education Student Benefits:

* Visit the GitHub Education website.
* Click on "Get student benefits" or a similar option.
* Provide the required information, including the name of your school and your school-issued email address if available.
* Upload proof of your academic status, such as a student ID, transcript, or a letter on school letterhead.
* Submit your application and await approval, which typically takes a few days.

3. Collaborate on a Project:

* **Choose a Project:**

Decide on a small project with your classmate. This could be a simple website, a script, or a small application.

* **Create a Repository:**

One of you should create a new repository on GitHub for the project. Make it public or private depending on your needs.

* **Invite Collaborator:**

The repository owner should go to the repository settings, then "Manage access" or "Collaborators and teams," and invite the classmate as a collaborator using their GitHub username or email.

* **Clone the Repository:**

Both you and your classmate should clone the repository to your local machines using Git.

Code

git clone <repository\_url>

* **Make Changes and Commit:** Work on the project, making changes to files. After making meaningful changes, commit them with clear and concise messages.

Code

git add .  
 git commit -m "Descriptive commit message"

* **Push Changes:** Push your committed changes to the remote repository on GitHub.

Code

git push origin main

* **Pull Changes:** Regularly pull changes from the remote repository to ensure your local copy is up-to-date with your classmate's contributions.

Code

git pull origin main

* **Resolve Conflicts:** If both of you modify the same lines of code, Git will flag a merge conflict. Resolve these conflicts by manually editing the affected files and committing the resolution.

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.

#### Table of Contents

[System software: The operating system](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#system-software-the-operating-system)

* [Windows](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#windows)
* [Linux](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#linux)
* [mac OS](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#macos)

[Application software and productivity tools](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#application-software-and-productivity-tools)

* [Office suites](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#office-suites)
* [Multimedia software](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#multimedia-software)
* [Web browsers](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#web-browsers)

[Utility programs and security software](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#utility-programs-and-security-software)

* [Antivirus software](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#antivirus-software)
* [Backup tools](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#backup-tools)
* [Optimization programs](https://teachers.institute/computer-in-education/types-of-software-system-application-utility/#optimization-programs)

GIT and GITHUB Training

**LAB EXERCISE:**

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

# 🡪Cloning a repository:-

When you create a repository on GitHub, it exists as a remote repository. You can clone your repository to create a local copy on your computer and sync between the two locations.

# 🡪Managing branches in your repository:-

Whenever you propose a change in Git, you [create a new branch](https://docs.github.com/en/articles/creating-and-deleting-branches-within-your-repository). Branch management is an important part of the Git workflow. After some time, your list of branches may grow, so it's a good idea to delete merged or stale branches.

# 🡪How to Merge two Git Repositories?

Merging two Git repositories can be a complex task, especially if they have distinct histories, branches, and commit structures. This guide explores different approaches and details each step involved in merging two Git repositories.

Application Software

**LAB EXERCISE:**

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

## Types of Application Software

### General Application Software

### Word Processing Software

### Spreadsheet Software

### Presentation Software

### Web Browsers

### Gaming software

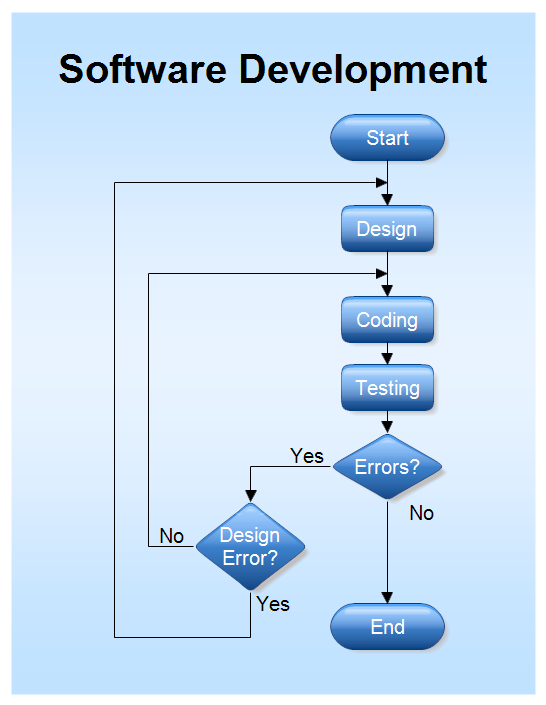
### Communication software

* All types of application software that assist in time management, database management, project management, content management, etc., are known as productivity software. Word processing, spreadsheets, and PowerPoint are examples of the same.

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.

**Software Requirement Specification (SRS) Format** as the name suggests, is a complete specification and description of requirements of the software that need to be fulfilled for the successful development of the software system. These requirements can be functional as well as non-functional depending upon the type of requirement.

Software Analysis

**LAB EXERCISE:**

🡪Perform a functional analysis for an online shopping system.

A functional analysis of an online shopping system identifies the key features and behaviors needed for users and the system to operate effectively.

* **User Management:**

This includes user registration, login, profile management, and potentially social media integration for account creation.

* **Product Catalog:**

Managing product listings, categories, search functionality, and detailed product information pages.

* **Shopping Cart:**

Allowing users to add, remove, and modify items in their cart before checkout.

* **Checkout Process:**

Handling payment processing, shipping information, order confirmation, and potentially guest checkout options.

* **Order Management:**

Providing order tracking, order history, and potentially features for managing returns or exchanges.

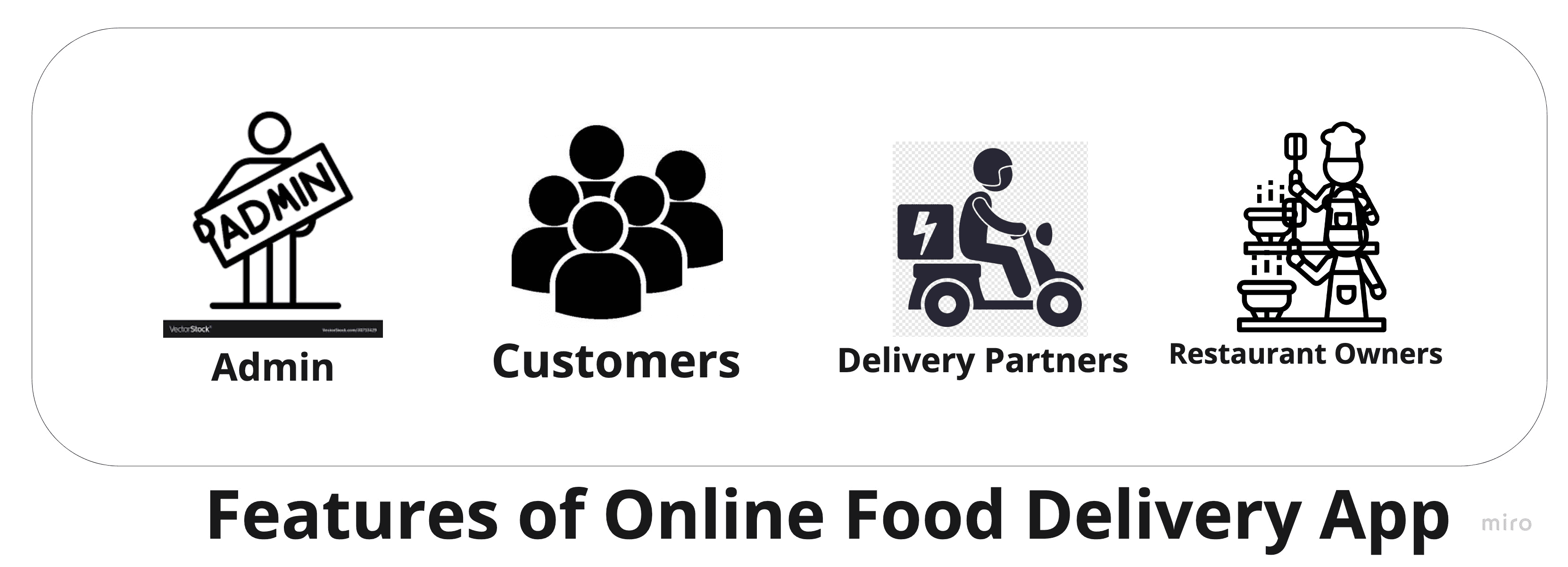
* **Administrative Functions:**

Managing products, users, orders, and potentially generating reports.

System Design

**LAB EXERCISE:**

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



* **Admin Panel**: Responsible for on boarding of Restaurants Owners and Delivery Partners.
* **Restaurant Owners:** Responsible for adding menus and managing orders and payments.
* **Delivery Partners:** Responsible for picking an order from the *Restaurant Owners* and delivering them to the customers.
* **Customers:** They are the actual client. They will be consuming the application. The customer will make an order from the system.

Software Testing

**LAB EXERCISE:**

🡪Develop test cases for a simple calculator program.

Developing test cases for a simple calculator program involves verifying its core arithmetic operations and handling of various inputs.

## **Functional Test Cases For A Calculator**

* Check whether the calculator can perform addition for two interfering numbers.
* Check whether the calculator can perform addition for two negative numbers.
* Check whether the calculator can perform subtraction for two integer numbers.
* Check whether the calculator can perform subtraction for two negative numbers.
* Check whether the calculator can perform subtraction of one negative and one positive number.
* Check whether the calculator can perform the multiplication of two integer numbers.
* Check whether the calculator can perform the multiplication of two negative numbers.
* Check whether the calculator can perform the multiplication of one negative and one positive number.
* Check whether the calculator can perform the division of two integer numbers.
* Check whether the calculator can perform the division of two negative numbers.
* Check whether the calculator can perform the division of one positive number and one negative number.
* Check whether the calculator can divide numbers by zero

Maintenance

**LAB EXERCISE:**

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

A critical software maintenance situation arose with the introduction of a new operating system, requiring immediate "adaptive maintenance" for a widely used banking application.

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 encure a smooth shopping experience for customers.

DFD (Data Flow Diagram)

**LAB EXERCISE:**

🡪Create a DFD for a hospital management system.

## **Data Flow Diagram – Hospital Management System**

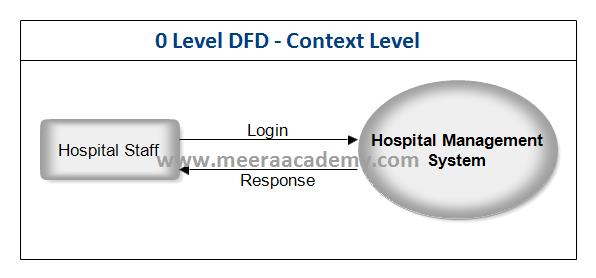
A data flow diagram is a graphical view of how data is Processed in a system in terms of input and output.  
The Data flow diagram (DFD) contains some symbol for drawing the data flow diagram.

## **Data flow diagram symbols**

|  |  |
| --- | --- |
| **Symbol** | **Description** |
| [dfd symbol](https://meeraacademy.com/wp-content/uploads/2016/09/arro.jpg) | **Data Flow** – Data flow are pipelines through the packets of information flow. |
| [dfd process](https://meeraacademy.com/wp-content/uploads/2016/09/process.jpg) | **Process :**A Process or task performed by the system. |
| [dfd entry symbol](https://meeraacademy.com/wp-content/uploads/2016/09/Entity.jpg) | **Entity :**Entity are object of the system. A source or destination data of a system. |
| [dfd source symbol](https://meeraacademy.com/wp-content/uploads/2016/09/source.jpg) | **Data Store :** A place where data to be stored. |

Context level DFD – 0 level

The Context level data flow diagram is describe the whole system. It shows the all user module who operate the system. The Hospital management System data flow diagram shows there is only one User hospital admin staff to run the system.

DFD 0 Level – Hospital Management System

Desktop Application

**LAB EXERCISE:**

🡪Build a simple desktop calculator application using a GUI library.

A simple desktop calculator application can be built using a GUI library such as Python’s Tkinter. the following outlines the steps involved:

* **Import Tkinter:** Begin by importing the tkinter module.

Python

from tkinter import \*

* **Initialize the Main Window:** Create the main application window.

Python

root = Tk()  
 root.title("Simple Calculator")  
 root.geometry("270x150") # Set desired window size

* **Create a Display Field:** Use an Entry widget to display input and results. A StringVar can be used to manage the text content of the entry field.

Python

display\_var = StringVar()  
 entry\_field = Entry(root, textvariable=display\_var)  
 entry\_field.grid(row=0, column=0, columnspan=4, ipadx=70) # Position and span across columns

* **Implement Button Actions:**
  + press(key) function: This function appends the pressed digit or operator to a global expression string and updates the display.
  + equal() function: This function evaluates the expression string using eval() and displays the result. Error handling (e.g., for division by zero) should be included.
  + clear() function: This function clears the expression string and the display.
* **Create Buttons:**

Create Button widgets for digits (0-9), operators (+, -, \\*, /), equals, and clear. Each button should be linked to its corresponding action function using the command attribute. Arrange the buttons using a layout manager like grid().

# Example for a number button

btn\_1 = Button(root, text='1', command=lambda: press('1'))

btn\_1.grid(row=2, column=0)

# Example for an operator button

btn\_plus = Button(root, text='+', command=lambda: press('+'))

btn\_plus.grid(row=1, column=3)

# Example for equals button

btn\_equals = Button(root, text='=', command=equal)

btn\_equals.grid(row=5, column=2)

# Example for clear button

btn\_clear = Button(root, text='C', command=clear)

btn\_clear.grid(row=5, column=0)

* **Run the Main Loop:** Start the Tkinter event loop to make the GUI interactive.

Python

root.mainloop()

Flow Chart

**LAB EXERCISE:**

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

