Module 1 – Overview of IT Industry

(LAB EXERCISE)

What is a Program?

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

Ans.

Hello World in C

```
#include <stdio.h>
int main() {
   printf("Hello, World!\n");
   return 0;
}
```

- The program starts with a header file #include <stdio.h> which is needed for the printf function.
- Every C program must have a main function, since execution begins there.
- Each statement ends with a semicolon (;).
- Curly braces { } are used to define the block of the function.
- The line return 0; indicates that the program finished successfully.

Hello World in Python

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

- In Python, no header files or imports are required for printing.
- There is no need for a main function; the program runs directly.
- Semicolons and braces are not needed, Python relies on indentation.
- The program is written in just one simple line.

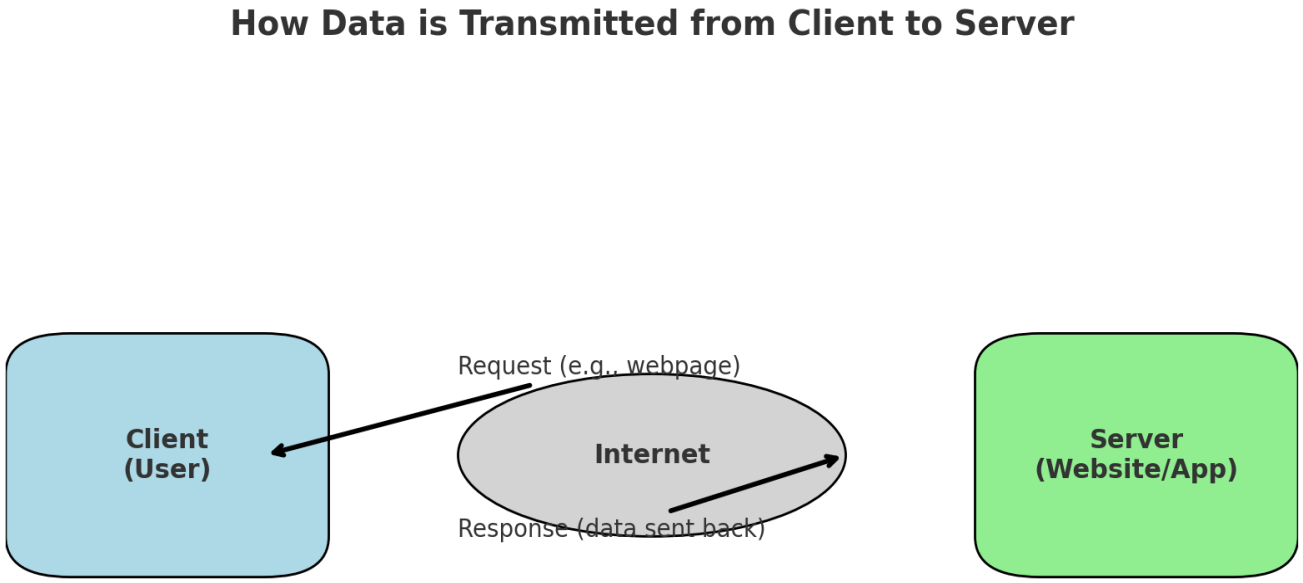
Comparison

- C requires more structure: headers, a main function, semicolons, and braces.
- Python is much simpler and more concise, making it easier for beginners.
- C is lower-level and closer to the system, used for performance-critical applications.
- Python is high-level, designed for readability and quick development.

World Wide Web & How Internet Works

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

Ans.



Network Layers on Client and Server

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

Ans.

1) Client sends a request

- The client creates an HTTP request.
- This request includes the method (like GET or POST), the URL/path, some headers (extra info), and sometimes a body (data).
- Example: A browser sending GET /index.html to a server.

2) Request travels through the Internet

- The request is sent over the internet, passing through routers, switches, and the DNS system (which translates the domain name into the server's IP address).

3) Server receives the request

- The server software (like Apache, Nginx, or Flask in Python) reads the request.
- It checks which resource or function is being asked for.

4) Server processes the request

- The server might fetch data from a database, run some logic, or simply prepare a file (like HTML, JSON, or an image).

5) Server sends a response

- The server constructs an HTTP response, which includes:
    - Status code (e.g., 200 OK, 404 Not Found)
    - Headers (extra info, like content type)
    - Body (the actual data: text, JSON, HTML, image, etc.)

6) Client receives the response

- The client reads the response.
- A browser will render the HTML as a webpage, or a program may just display the text/data.

Types of Internet Connections

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

Ans.

Getting an internet connection that works for you is important, and the best choice depends on where you live and what you need it for. Here is a simple breakdown of the main types of connections.

Fiber Internet

The newest and fastest option, using glass cables to send data as light.

- **Pros:**
  - Extremely Fast: The best speeds for both downloading and uploading.
  - Super Reliable: The signal is not affected by weather or distance.
  - Low Lag: Great for online gaming and video calls.
- **Cons:**
  - Limited Availability: Not available in many rural or remote areas.
  - Higher Cost: Often more expensive to install and for monthly service.

Broadband Internet (Cable and DSL)

This is the most common type of internet, using existing phone or TV lines.

- **Pros:**
  - Widely Available: Easy to get in most cities and towns.
  - Affordable: Generally cheaper than a fiber connection.
  - Good Speed: Fast enough for most everyday tasks like streaming and browsing.
- **Cons:**
  - Slower than Fiber: Does not offer the same top speeds.
  - Shared Connection: Speeds can slow down when many people in your neighborhood are online at the same time (especially with cable).
  - Slower Uploads: Upload speeds are often much slower than download speeds.

Satellite Internet

This type of connection uses a dish to connect to a satellite in space.

- **Pros:**
  - Available Almost Anywhere: The only option for many people in very remote areas with no other services.
  - Quick to Set Up: Does not require laying down new cables on the ground.

- **Cons:**
  - High Lag (Latency): The signal has to travel a long distance, causing a delay that is bad for gaming and live video calls.
  - Affected by Weather: Heavy rain or snow can weaken or cut off the signal.
  - Slower Speeds & Data Limits: Generally has slower speeds and often comes with monthly data caps.

Protocols

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

Ans.
Here's how to simulate HTTP and FTP requests using curl from the command line.

HTTP Request (GET)
Use curl to fetch a web page or data from a server. The basic command for a GET request is simply curl followed by the URL.

- **Command:** curl https://www.example.com
- **What it does:** This command sends a GET request to https://www.example.com and displays the HTML content of the page in your terminal.

FTP Request (Download)
You can use curl to download files from an FTP server. You'll need the FTP server address and the file path.

- **Command:** curl -O ftp://ftp.mozilla.org/README
- **What it does**: This command connects to the ftp.mozilla.org server, navigates to the / directory, and downloads the file named README. The -O flag tells curl to save the file with its original name in your current directory.

Application Security

LAB EXERCISE: Identify and explain three common application security vulnerabilities. Suggest possible solutions

Ans.
Here are three common application security vulnerabilities, explained in an easy way, along with simple solutions.

1. SQL Injection (SQLi)

Explanation: This happens when an attacker puts malicious code into a form field or URL. The application treats this code as a database command, which can allow the attacker to steal, change, or delete sensitive data.
Solution: Use prepared statements (or parameterized queries). This separates the user input from the database command, so the application never confuses the data with the code.

2. Cross-Site Scripting (XSS)

Explanation: An attacker injects a malicious script into a trusted web page. When another user visits the page, the script runs in their browser, potentially stealing their cookies, session tokens, or other private information.
Solution: Sanitize and validate all user input before it's displayed on a web page. This means removing or escaping any special characters that could be part of a malicious script.

3. Broken Authentication

Explanation: This vulnerability occurs when an application's login or session management is weak. Attackers can guess passwords, bypass login steps, or steal session tokens to gain access to a user's account.
Solution: Use strong, industry-standard authentication practices. This includes enforcing strong passwords, using multi-factor authentication (MFA), and securing session IDs.

Software Applications and Its Types

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

Ans.
Here are five common applications you use daily, classified as either system or application software.

System Software

- **Microsoft Windows (or macOS/Linux**): This is an Operating System. It's system software because it manages all the computer's hardware and provides the basic platform for everything else to run.

Application Software

- **Google Chrome (or Firefox/Safari):** This is a Web Browser. It's application software because its purpose is to perform a specific task for you, the user: browsing the internet.
- **Microsoft Word (or Google Docs):** This is a Word Processor. It's application software because it's designed for a specific user task—creating and editing documents.
- **VLC Media Player:** This is a Media Player. It's application software because it has one specific job: to play video and audio files for your enjoyment.
- **Fortnite (or any other video game):** This is a Video Game. It's application software because it's created for a specific user task: entertainment.

Software Architecture

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

Ans.

1. Presentation Tier (Client)
This is the top-level tier of the application. It's what the end user sees and interacts with directly.

- Function: It presents the user interface and translates user actions into requests for the next tier.
- Components: Web browsers, HTML, CSS, JavaScript.

2. Logic Tier (Application Server)
This middle tier contains the application's core logic. It processes the user's requests, commands the data tier, and returns the results to the presentation tier.

- Function: It handles the main business logic and processes requests.
- Components: Application servers (like Node.js, Python, or Java), APIs.

3. Data Tier (Database)
This is the bottom tier, which stores and manages all the data for the application.

- Function: It stores and retrieves information as requested by the logic tier.
- Components: Databases (like MySQL, PostgreSQL, MongoDB), file systems.

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.

Ans:

1. Presentation Layer (Your Web Browser)
When you visit an online store, your web browser is the presentation layer. It displays the products, the shopping cart, and the checkout form. When you click "Add to Cart," you are interacting with this layer.

2. Business Logic Layer (The Server)
The website's server is the business logic layer. When you click "Add to Cart," your request is sent here. This layer does the "thinking":

- It checks if the item is in stock.
- It calculates the total price, including any discounts or taxes.
- It updates your shopping cart. This layer makes sure all the rules of the business are followed before it talks to the database.

3. Data Access Layer (The Database)
This is the database where all the information is stored. The business logic layer sends a command to this layer, which then performs the necessary action.

- The database finds the product and checks its inventory.
- It updates the stock count for that product.
- It saves the item to your shopping cart record.

In short, the Presentation Layer shows you the website, the Business Logic Layer handles the rules and calculations, and the Data Access Layer stores and manages all the information.

Software Environments

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

Ans:

Different software environments are used to manage a project's lifecycle from development to final release. They prevent changes and bugs from affecting the live version of the software that users interact with.

1. Development (Dev)
This is where developers write and test new code. It's a personal, flexible environment where mistakes are expected and quickly fixed. It's like a messy workbench for building and experimenting.

2. Testing (Test)
After new features are built, they move to the testing environment. Here, a team of quality assurance (QA) testers checks for bugs and makes sure the new code works as it should. This environment is set up to be a lot like the final production environment.

3. Production (Prod)
This is the live, final environment that real users access. It is the most stable and secure environment, and no unapproved changes are ever made here. The goal is to provide a reliable and smooth experience for the end user.
Setting Up a Basic Environment in a Virtual Machine
A virtual machine (VM) is a great way to set up a clean, isolated environment on your computer without affecting your main operating system. Here's a simple process using a common virtualization tool like Oracle VirtualBox:

1. Download and Install VirtualBox: Get the free software from the Oracle website.
2. Create a New VM: Open VirtualBox and click "New." You'll be asked to name your VM, choose the operating system you want to install (like Ubuntu Linux), and set how much RAM and disk space it can use.
3. Install the OS: Start the new VM. It will guide you through the process of installing the operating system from a downloaded file (called an ISO).
4. Install Development Tools: Once the OS is running, you can install any software you need, such as a code editor, a programming language (like Python), or a web server. This new environment is now your personal, isolated "Dev" environment.

Source Code

LAB EXERCISE: Write and upload your first source code file to Github.

Ans:

1. Create a Repository

A repository, or "repo," is where your project lives on GitHub.

- Go to your GitHub dashboard and click the green "New" button to create a new repository.
- Give it a name, like my-first-repo, and an optional description.
- Choose to make it Public or Private, then click "Create repository."

2. Prepare Your Code

Now, create a simple source code file on your computer.

- Open a text editor and create a file, for example, hello.py.
- Add a simple line of code, like print("Hello, GitHub!"), and save the file.

3. Upload to GitHub

Finally, you will use the command line to upload your file.

- Open your terminal or command prompt.
- Initialize Git: In your project folder, type git init to start a new Git repository.
- Add your file: Use git add hello.py to stage your file.
- Commit your file: Use git commit -m "Add my first Python file" to save your changes with a message.
- Connect to GitHub: Copy the git remote add origin command from your new GitHub repository page and paste it into your terminal.
- Push your code: Use git push -u origin master to upload your code to GitHub.

Your file will now be visible in your new repository on GitHub.

Github and Introductions

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

Ans:

Step 1: Create a New Repository
This is the central location where your project's code will live on GitHub.

1. Log in to your GitHub account and navigate to your dashboard.
2. In the top-left corner, click the green "New" button or the + icon, and then select "New repository."
3. Choose a name for your repository (e.g., my-project). Keep the name short and descriptive.
4. Optionally, add a brief description of your project.
5. Select whether you want the repository to be Public (visible to everyone) or Private (only visible to you and people you choose).
6. Click the "Create repository" button.

Step 2: Commit and Push Code Changes
Once your repository is created, you can add your code. These commands are performed in your computer's terminal or command prompt.

1. **Initialize Git in your local project folder.**
    - Open your terminal and navigate to your project folder using cd.
    - Type git init to prepare the folder for version control.
2. **Add and Commit your files.**
    - Create or edit a file in your project.
    - Use git add . to stage all of your changes for the next commit.
    - Use git commit -m "Your descriptive message here" to save your changes. The message should explain what you did.
3. **Connect to GitHub and push your changes**.
    - Go back to your new GitHub repository page and copy the git remote add origin command.
    - Paste it into your terminal and press Enter. This links your local folder to the GitHub repository.
    - Finally, use git push -u origin main (or master if your branch is named that) to upload all of your committed changes to GitHub. Your code is now live and can be viewed by others.

Student Account in Github

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

Ans:

Step 1: Create a GitHub Account

1. Go to the GitHub website and click on the "Sign up" button.
2. Follow the instructions to create a new account. You'll need to provide an email address, create a password, and choose a username.
3. Choose the free plan for personal use.

Step 2: Create a Project Repository

1. One person, the project owner, will create a new repository (a project folder) on their GitHub account.
2. On their GitHub dashboard, they should click the green "New" button to create a new repository.
3. They should give it a name and an optional description.

Step 3: Add a Collaborator

1. After the repository is created, the project owner will go to the "Settings" tab of the repository.
2. In the left sidebar, they should click on "Collaborators".
3. They can then search for their classmate's GitHub username and add them as a collaborator. This gives the classmate permission to make changes to the project.

Step 4: Collaborate

Now both students can work on the project.

- The project owner can commit and push their code changes to the repository.
- The collaborator can clone the repository to their computer, make their own changes, and then commit and push them back to the shared repository.
- Both students can now see each other's changes and work on the same project together.

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

- Windows 10 / 11 (Operating System)
- Android (Operating System on mobile)
- Device Drivers (e.g., Printer Driver, Graphics Driver)

2. Application Software

- Microsoft Word (Word Processing)
- Google Chrome (Web Browser)
- WhatsApp (Communication)
- YouTube (Entertainment/Streaming)
- Zoom / Google Meet (Video Conferencing)
- MS PowerPoint (Presentation)

3. Utility Software

- WinRAR / 7-Zip (File Compression)
- Antivirus (e.g., Avast, Quick Heal, Windows Defender)
- Disk Cleanup (Windows Utility)
- CCleaner (System Optimization)

GIT and GITHUB Training

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

Ans.

1. Cloning a Repository

- Open terminal / Git Bash.
- Run the command:
- git clone <repository_link>

Example:
git clone https://github.com/example/repo.git

- This will create a local copy of the repository on your computer.

2. Creating a Branch

- Navigate into the project folder:
- cd repo
- Check existing branches:
- git branch
- Create a new branch:
- git branch feature-branch
- Switch to that branch:
- git checkout feature-branch

(Shortcut: git checkout -b feature-branch creates + switches in one step)

3. Making Changes and Committing

- Edit files in your branch.
- Add changes:
- git add .
- Commit changes:
- git commit -m "Added new feature"

4. Merging Branches

- Switch back to the main branch:
- git checkout main
- Merge your feature branch into main:
- git merge feature-branch
- Resolve conflicts if any, then commit.

Application Software

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

Ans.

Types of Application Software and Their Role in Productivity:

1. **Word Processing Software**
   - Examples: Microsoft Word, Google Docs
   - Helps in creating, editing, and formatting documents quickly and neatly.
   - Improves productivity by offering spell check, templates, and easy sharing options.
2. **Spreadsheet Software**
   - Examples: Microsoft Excel, Google Sheets
   - Used for calculations, data analysis, and financial planning.
   - Boosts productivity with formulas, charts, and automation features.
3. **Presentation Software**
   - Examples: Microsoft PowerPoint, Canva
   - Assists in creating visual presentations for teaching, business, and reports.
   - Increases productivity by providing templates, design tools, and multimedia support.
4. **Database Management Software**
   - Examples: MySQL, MS Access
   - Helps in storing, retrieving, and managing large amounts of data efficiently.
   - Enhances productivity by reducing manual record-keeping and improving data accuracy.
5. **Communication Software**
   - Examples: Zoom, Microsoft Teams, WhatsApp
   - Enables online meetings, messaging, and collaboration.
   - Improves productivity by saving travel time and supporting teamwork.
6. **Graphics and Multimedia Software**
   - Examples: Adobe Photoshop, VLC Media Player
   - Used for image editing, video playback, and content creation.
   - Boosts creativity and productivity in media-related work.
7. **Web Browsers**
   - Examples: Google Chrome, Mozilla Firefox
   - Allow users to access information, resources, and online tools.
   - Improves productivity by enabling quick research and access to cloud-based apps.

Software Development Process

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

Ans:

Software Requirement

LAB EXERCISE: Write a requirement specification for a simple library management system.

Ans:

1. Purpose

To manage books, members, and borrowing/returning activities in a library.

2. Users

- Admin – controls system and reports.
- Librarian – manages books and issues/returns.
- Member – searches and borrows books.

3. Functional Requirements

- Book Management: Add, update, delete, search books.
- Member Management: Register, update, delete members.
- Borrow/Return: Issue and return books, calculate fines.

- Reports: Show available, borrowed, overdue books, and member activity.

4. Non-Functional Requirements

- Easy to use interface.
- Secure login.
- Works fast for up to 1000 members and 5000 books.

5. Constraints

- Only registered members can borrow books.
- Max 5 books per member.
- Return period: 14 days.

Software Analysis

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

Ans:

Functional Analysis – Online Shopping System

1. Users

- **Customer: Browses, orders, and pays for products.**
- **Admin: Manages products, users, and reports.**
- **Delivery Staff: Handles order delivery status.**

2. Main Functions

A. Customer Functions

- Account Management: Register, login, update profile.
- Product Browsing: Search, filter, view product details.
- Shopping Cart: Add, update, or remove items.
- Order Placement: Confirm order, choose delivery address.
- Payment: Multiple payment options (COD, card, online wallet).
- Order Tracking: View status (processing, shipped, delivered).

B. Admin Functions

- Product Management: Add, update, delete products with details.
- User Management: Manage customer accounts.
- Order Management: Approve, cancel, or update orders.
- Reports: Generate sales, inventory, and customer activity reports.

C. Delivery Staff Functions

- View Assigned Orders: See pending deliveries.
- Update Status: Mark orders as delivered.

3. System Functions (Supportive)

- Authentication & Security: Secure login, encrypted transactions.
- Notifications: Email/SMS updates on order status.
- Search Engine: Fast product search and filtering.
- Database Management: Store products, users, and order data.

System Design

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

Ans:

1. Users

- Customer: Places food orders.
- Restaurant: Accepts and prepares orders.
- Delivery Partner: Picks up and delivers orders.
- Admin: Manages system operations.

2. Frontend (User Interfaces)

- Mobile App (Customer): Browse restaurants, order food, track delivery, payment.
- Mobile App (Delivery Partner): Accept delivery requests, update status, navigation.
- Restaurant Dashboard: Receive orders, update menu, track deliveries.
- Admin Panel (Web): Manage users, restaurants, delivery partners, and reports.

3. Backend (Server)

- Authentication Service: Login, signup, and role-based access.
- Order Management Service: Handles order placement, updates, and tracking.
- Restaurant Service: Manages menus, availability, and pricing.
- Delivery Management Service: Assigns delivery partners and tracks locations.
- Payment Gateway: Processes secure payments.
- Notification Service: Sends SMS, email, or push notifications.

4. Database

- User Database: Stores customer, restaurant, and delivery partner details.
- Menu Database: Stores restaurant menus and food items.
- Order Database: Stores order details and statuses.
- Payment Records: Stores payment history and invoices.

5. External Services

- Maps & GPS API: For real-time tracking and delivery navigation.
- Payment API (UPI, cards, wallets): For secure transactions.

Software Testing

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

Ans:

Operation	Input	Expected Output
Addition	2 + 3	5
Addition	-5 + 7	2
Addition	0 + 8	8
Subtraction	9 - 4	5
Subtraction	4 - 9	-5
Subtraction	0 - 5	-5
Multiplication	3 × 4	12
Multiplication	-3 × 5	-15
Multiplication	0 × 99	0
Division	8 ÷ 2	4
Division	9 ÷ 4	2.25
Division	5 ÷ 0	Error

Maintenance

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

Ans:

Real-World Case of Critical Software Maintenance
Case: WhatsApp Global Outage (October 2022)
Background:
WhatsApp, a widely used messaging application, experienced a global outage in October 2022. Millions of users worldwide were unable to send or receive messages for about 2 hours.
Reason for Maintenance:
The issue was caused by a technical fault in WhatsApp's server infrastructure. This required corrective maintenance to quickly restore services.
Type of Maintenance:

- Corrective Maintenance – fixing errors in the server systems.
- Emergency Maintenance – since the system failure impacted users globally.

Actions Taken:

- Engineers identified the server configuration issue.
- Applied fixes to restore the messaging service.
- Conducted monitoring to ensure stability after recovery.

Outcome:

- Services were restored after 2 hours.
- Meta (WhatsApp's parent company) confirmed the issue and improved monitoring to prevent future outages.

DFD (Data Flow Diagram)

LAB EXERCISE: LAB EXERCISE: Create a DFD for a hospital management system.

Ans:

DFD – Hospital Management System

External Entities:

- Patient – Registers, books appointments, pays bills.
- Doctor – Provides treatment, updates patient records.
- Admin/Staff – Manages records, schedules, billing.

Processes:

1. Patient Registration & Appointment
2. Treatment & Medical Records
3. Billing & Payment
4. Reports & Administration

Data Stores:

- Patient Database
- Doctor Database
- Medical Records
- Billing Records

Text Representation (Level 0 DFD):

[Patient] → (1. Registration & Appointment) → [Patient Database]
[Patient] → (3. Billing & Payment) → [Billing Records]

[Doctor] ↔ (2. Treatment & Records) ↔ [Medical Records]

[Admin/Staff] ↔ (4. Reports & Administration) ↔ [All Databases]

Flow Chart

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

Ans: