Assignment

Module 1 – Foundation

• What is a HTTP?

⇨ HTTP stands for HyperText Transfer Protocol. It is the foundation of data communication on the World Wide Web.

⇨ HTTP is a **protocol** (set of rules) that **browsers and servers use to communicate**. It helps your browser request web pages from a server, and the server sends them back.

• What is a Browsers? How they works?

⇨ A browser is a software (like Chrome, Firefox, or Safari) that helps you open and see websites on the internet.

⇨ **You type a website name** (like `www.youtube.com`).

⇨ The browser **goes to find the website** on the internet.

⇨ It **asks the server** for the page.

⇨ The server **sends back** the page.

⇨ The browser **shows** the website to you.

• What is Domain Name?

⇨ A domain name is the address you type into a web browser to visit a website, like www.example.com.

Example: `www.example.com`

⇨ **www**: This stands for **World Wide Web** (optional, but common).

⇨ **example**: This is the **unique name** of the website.

⇨ **.com**: This is the **top-level domain (TLD)**. Other common ones include `.org`, `.net`, `.edu`.

• What is hosting?

⇨ Hosting means putting your website on a special computer (server) so that anyone can see it on the internet.

Imagine This:

You made a project on your computer.
But only **you** can see it.
If you want **everyone** to see it, you need to **upload it** to the **internet**.
That's what **hosting** does — it makes your website **visible to the world**.

Fundamentals of World Wide Web

• Difference between Web Designer and Web Developer

Feature	Web Designer	Web Developer
Focus	Looks/design of the website	Functionality of the website
Work Area	Colors, layout, fonts, images	Code, logic, performance
Tools Used	Photoshop, Figma, Adobe XD, HTML, CSS	HTML, CSS, JavaScript, PHP, Python, databases
Skills	Creative, artistic, UI/UX skills	Coding, logic, problemsolving
Goal	Make it **beautiful** and **userfriendly**	Make it **work properly** and **securely**
Example	Designs how a button looks	Makes the button actually work

⇨ A **Web Designer** makes the **look** of the website.
⇨ A **Web Developer** makes the **brain** and **function** of the website.

⇨ Designer makes the design.
⇨ Developer builds the website using that design.

• What is a W3C?

⇨ **W3C** stands for **World Wide Web Consortium**.
⇨ It is the **main international organization** that **creates rules and guidelines for websites** to make the web work **better for everyone**.

• What is Domain?

⇨ A **domain** is the **name** of a website that you type in the browser to visit it. ⇨ A **domain** is like the **address** of a house — but on the **internet**.

Example:

- www.google.com
- www.facebook.com
- www.yoursite.in

Common Domain Extensions:

- .com → commercial
- .org → organization
- .net → network
- .in, .uk, .us → country-specific
- .edu → education

• What SEO?

⇨ SEO stands for Search Engine Optimization.

⇨ SEO is the process of improving your website so it appears higher in search results (like Google).

⇨ Example:

You search for **"best chocolate cake recipe"** — the websites at the top have **good SEO**.

• Why is SEO Important?

⇨ Higher ranking = **More visitors**

⇨ More visitors = **More business, more views, more sales**

• What is SDLC life cycle?

⇨ **SDLC** stands for **Software Development Life Cycle**.

⇨ **SDLC is the step-by-step process used to plan, build, test, and deliver software.**

⇨ It helps teams make software in an **organized** and **efficient** way.

Step	Name	What Happens
	Planning	Decide what to make and why
	Requirement Analysis	Understand what the user wants
	Design	Plan how the software will look and work

Development	Developers write the actual code
(Coding)	
Testing	Check the software for bugs or errors
Deployment	Make the software live for users
Maintenance	Fix issues and update the software over time

Fundamentals of IT

• Explain in your own words what a program is and how it functions.

- ⇨ A program is a set of instructions written by a person (called a programmer) that a computer can understand and follow to do a task**.**
- ⇨ A program tells the computer what to do, step by step

How Does a Program Work?

1. **You write the program** using a programming language (like Python, Java, C++).
2. The computer **reads the instructions** line by line.
3. It does what the instructions say — like doing calculations, showing a message, opening a file, etc.
4. If everything is correct, the program **runs smoothly** and gives the right result.

• What are the key steps involved in the programming process?

⇨ **7 Key Steps in the Programming Process (Explained Simply):**

Step	Name	What Happens
	Understanding the Problem	First, clearly know what you want the program to do. Example: Add two numbers, sort a list, or make a calculator.
	Planning the Solution	Think and decide **how** you will solve the problem

	(flowchart, pseudocode, steps).
Writing the Code	Write the actual program using a **programming language** (like Python, C++, Java).
Compiling/Interpreting	Convert your code into a form the computer understands (machine language).
Testing and Debugging	Run the program to **check for errors (bugs)** and **fix** them.
Running the Program	Once it works, run the program to get the output.
Maintenance	Update or improve the program over time (fix bugs, add new features).

Programming is not just writing code — it includes **thinking, planning, testing, fixing**, and **improving**.

- **What are the main differences between high-level and low-level programminglanguages?**

Feature	**High-Level Language**	**Low-Level Language**
Definition	Easy-to-understand language for humans	Close to machine (hardware) language
Readability	Easy to read and write	Hard to read and understand
Examples	Python, Java, C++, JavaScript	Assembly language, Machine code (binary)
Speed	Slower (more layers between code and hardware)	Faster (closer to hardware)
Portability	Not portable (depends on hardware)	Works on many computers (platform-independent)
Use Case	Used to build apps, websites, software	Used to control hardware, write system software (like OS, drivers)
Ease of Use	Easier for beginners	Difficult, used by experts

- **Describe the roles of the client and server in web communication.**
    - ⇨ **Client**

The client is the user's device (like your browser: Chrome, Firefox, etc.) that:

- Requests data from the web.
- Displays the result to the user.

Example: When you type www.google.com in your browser, your browser is the client.

⇨ **Server**

A server is a powerful computer that:

- Stores websites, files, and data.
- Listens for requests from clients.
- Sends the correct response (like HTML pages, images, etc.).

Example: Google's server gets your request and sends back the Google homepage.

How They Work Together:

1. Client (browser) sends a request:
   "Please show me the Google homepage." 2.

Server receives the request and responds:
"Here is the homepage."
3. Client displays it to you on screen.

- The client asks, and the server answers. Together, they make websites work.

- **Explain the function of the TCP/IP model and its layers**
- The TCP/IP model (Transmission Control Protocol/Internet Protocol model) is a conceptual framework used to describe how data is transmitted over a network. It forms the foundation of the internet and most modern computer networks.

Function of the TCP/IP Model

The **TCP/IP model** helps:

1. **Standardize communication** between computers across different networks.

2. **Break down complex networking tasks** into manageable layers.
3. **Ensure data is sent and received accurately and efficiently** from source to destination.
4. **Allow interoperability** between different devices, systems, and software.

Layers of the TCP/IP Model

The TCP/IP model has **4 layers**, each with a specific role:

1. Application Layer

- **Function:** Provides services and interfaces for applications to communicate over a network.
- **Examples:** Web browsing (HTTP), email (SMTP), file transfer (FTP), DNS.
- **Users interact here** (e.g., using a web browser).

---

2. Transport Layer

- **Function:** Ensures reliable data delivery between devices.
- **Protocols:**
  - **TCP (Transmission Control Protocol):** Reliable, error-checked delivery (e.g., for emails, web pages).
  - **UDP (User Datagram Protocol):** Faster, connectionless, no error-checking (e.g., for videos, games).
- **Key Features:** Segmentation, error detection, flow control.

---

3. Internet Layer

- **Function:** Handles logical addressing and routing of data packets.
- **Main Protocol: IP (Internet Protocol)**.
- **Key Features:**
  - Assigns IP addresses.
  - Finds the best path to the destination (routing).
  - Breaks data into packets and reassembles it at the destination.

---

4. Network Access Layer (also called Link Layer)

- **Function:** Deals with physical transmission of data over network hardware (like cables or Wi-Fi).
- **Includes:** Ethernet, Wi-Fi, MAC addresses.
- **Responsible for:** Sending bits over a physical medium, accessing network devices.

Explain Client Server Communication

- Client-Server Communication is a method used in networking where one system (the client) requests information or services from another system (the server).

- **How It Works:**
  Client = The user or device that **initiates the request**.
  Server = A system that **waits for requests and responds** with data or actions.

- **Example: Opening a Website**

1. You type a URL (like www.google.com) in your browser.
2. Your browser (client) sends a request to Google's server.
3. The server processes the request and sends back the web page.
4. The browser displays the page.

- **Communication Steps (Simplified):** 1. **Client Sends Request**
  → Asks for a webpage, file, or data.
2. **Server Receives Request** → Understands
   what's needed.
3. **Server Sends Response**
   → Sends back the requested content or message.
4. **Client Receives and Displays** → Shows
   the content to the user.

• How does broadband differ from fiber-optic internet?

- **Broadband** = An umbrella term for **any high-speed internet** (includes DSL, cable, wireless, and fiber).
- **Fiber-optic internet** = A **specific, advanced type of broadband** using light signals for **very high speed and reliability**.

• What are the differences between HTTP and HTTPS protocols?

⇨ HTTP: Sends your data like a postcard – anyone can read it.
⇨ HTTPS: Sends your data like a sealed envelope – private and protected.

Feature	HTTP	HTTPS
Full Form	HyperText Transfer Protocol	HyperText Transfer Protocol Secure
Security	Not secure	Secure (uses SSL/TLS encryption)

Data Encryption	Data sent as plain text	Data is encrypted and safe from hackers
Port Used	Port 80	Port 443
URL Format	http://example.com	https://example.com
Certificate	No SSL certificate required	Requires an SSL/TLS certificate
Used For	Non-sensitive browsing (rare today)	All secure websites (e.g., banking, shopping)
Browser Indicator	May show "Not Secure" warning	Shows a padlock icon in the address bar

- ## What is the role of encryption in securing application, Software Applications and Its Types

⇨ Encryption is the process of converting data into a secret code to prevent unauthorized access.

Role in Applications:

- **Protects data** (like passwords, messages, credit card info)
- Ensures **only the intended person** can read the data
- Keeps **user information safe** during storage and transfer

⇨ **What are Software Applications?**

Programs that help users do specific tasks on computers or phones.

Types of Software Applications:

Type	Description	Examples
Web Apps	Run in browsers using the internet	Gmail, YouTube
Mobile Apps	Run on smartphones	WhatsApp, Instagram
Desktop Apps	Installed on computers	MS Word, Photoshop
Enterprise Apps	Used by businesses for operations	ERP, Salesforce
Utility Apps	Help with maintenance or system tasks	Antivirus, File manager

• What is the difference between system software and application software?

Feature	System Software	Application Software
Purpose	Runs the computer and manages hardware	Helps the user perform specific tasks
User Interaction	Works in the background, not directly used often	Used directly by the user
Examples	Operating System (Windows, Linux), Drivers	MS Word, Chrome, WhatsApp, Photoshop
Installation	Comes pre-installed or needed for system to work	Installed based on user's need
Dependency	Needed for application software to run	Depends on system software

⇨ System Software = Like the engine of a car – it makes everything run.
⇨ Application Software = Like the radio or GPS – helps you do specific things.

• What is the significance of modularity in software architecture?

⇨ Modularity means dividing software into separate, independent modules where each module does a specific task.

• Why are layers important in software architecture?

⇨ Layers in software architecture help organize the code into separate parts (or "levels"), each with a specific role. This structure brings clarity, control, and flexibility to software development.

⇨ **Common Layers in Software Architecture:**

1. **Presentation Layer** – UI, user interaction
2. **Business Logic Layer** – App rules and operations
3. **Data Access Layer** – Interacts with the database
4. **Database Layer** – Stores the actual data

• Explain the importance of a development environment in software production.

⇨ A development environment is the setup where developers write, test, and debug software before it's released.

⇨ A development environment is like a **practice ground** for building software — it lets you **create, test, and improve** your code **safely and efficiently** before showing it to users.

• What is the difference between source code and machine code?

- Source Code = The instructions you write
- Machine Code = The translated version the computer can understand

⇨ Like writing a letter in English (source code) and translating it to binary for a robot to read (machine code).

• Why is version control important in software development?

⇨ Version control is the practice of tracking and managing changes to software code over time. It's essential for both individuals and teams working on software projects.

⇨ Version control is like keeping a **diary** of your software changes. If something goes wrong, you can **look back** at earlier versions and fix it. It helps **teams collaborate** without chaos.

⇨ **Common Version Control Tools:**

- **Git** (most popular)
- **SVN (Subversion)**
- **Mercurial**

- **What are the benefits of using Github for students?**

⇨ GitHub is a popular platform that helps students manage and share their code

Here are the key benefits of using GitHub for students:

⇨ Collaboration
⇨ Version Control
⇨ Showcase Projects
⇨ Learning Tool
⇨ Real-World Skills
⇨ Free Hosting for Websites
⇨ Documentation
⇨ Networking
⇨ Access to Resources

⇨ GitHub is like a **digital notebook** where students can store, track, and share their code. It's a place to **collaborate** with others, **learn from real-world examples**, and **show off** what they've built.

• What are the differences between open-source and proprietary software?

⇨ **Open-source software** = **Free** and **modifiable** by anyone, with **community-driven** support.

⇨ **Proprietary software** = **Paid** and **controlled** by a company, with **official** support and updates.

Feature	Open-Source Software	Proprietary Software

Source Code Access	**Freely available** to modify and distribute	**Closed**, not available for public modification
License	Released under licenses like **MIT, GPL**	**Paid license** or subscription required
Cost	**Free** (may have paid versions for added features)	**Paid** (usually)
Customization	Can be **customized** by anyone	**Cannot be modified** by users
Updates	Often **community-driven** updates	**Company-driven** updates and support
Security	**Transparency** allows the community to find and fix bugs	**Controlled** security, but vulnerabilities may go unnoticed
Support	Community forums, documentation, paid support for some	Official support from the company
Examples	Linux, Firefox, WordPress	Microsoft Office, Adobe Photoshop, Windows

• How does GIT improve collaboration in a software development team?

⇨ **Git** is a version control system that helps developers work together efficiently on the same codebase.

⇨ Git helps a team of developers **work on the same project without stepping on each other's toes**, while keeping everything **organized, trackable, and safe**.

• What is the role of application software in businesses?

⇨ Application software helps businesses perform specific tasks to run efficiently and effectively.

⇨ **Application software helps businesses work faster, stay organized, and serve customers better.**
It's like a digital helper that makes business tasks easier and smarter.

• What are the main stages of the software development process?

⇨ The software development process is a step-by-step method used to create, test, and deliver software.

You plan it, design it, build it, test it for safety, move in, and then fix things as needed.

• Why is the requirement analysis phase critical in software development?

⇨ The **requirement analysis phase** is when developers and stakeholders **gather, study, and define what the software should do**.

⇨ **Requirement analysis is like reading the recipe before cooking.** If you miss this step, you might make the **wrong dish** or forget **important ingredients**.

• What is the role of software analysis in the development process?

⇨ Software analysis is like making a detailed plan before building a house. It ensures you build the right thing the right way.

Key Roles of Software Analysis:

⇨ **Explanation**
⇨ **Role**
⇨ Helps identify **what the user or client wants**
⇨ **Understand User Needs**
⇨ Specifies all the **features, functions, and limits**
⇨ **Define System Requirements**
⇨ Finds missing or conflicting requirements
⇨ **Detect Problems Early before coding**
⇨ Helps estimate **time, cost, and resources** needed
⇨ **Improve Planning**
⇨ Acts as a **blueprint** for building the software
⇨ **Guide Design and Development**
⇨ Makes sure the final software **meets user**
⇨ **Ensure Quality expectations**

• What are the key elements of system design?

⇨ System design is about planning how the software system will work — both in structure and behavior.

⇨ System design is like planning a **blueprint for a building** —
It covers **what goes where, how things connect**, and **how people use it safely and smoothly**.

⇨ Key Elements of System Design:

Element	Description
Architecture Design	Overall **structure** of the system (e.g., layers, components, data flow)
User Interface Design	How the **user will interact** with the system (layouts, buttons, screens)
Data Design	How **data is stored, managed, and accessed** (e.g., databases, files)
Module Design	Breaking the system into **smaller, manageable parts** or modules
Interface Design	How different parts of the system **communicate** with each other (APIs)
Security Design	Planning for **data protection**, user access, and safe communication
Performance Design	Ensures the system runs **fast, reliably**, and uses resources efficiently
Scalability	Ability of the system to **handle growth** (more users, data, tasks)

• Why is software testing important?

⇨ Software testing checks if the software works correctly, is error-free, and meets user needs before it is released.

⇨ Key Reasons Why Testing Is Important:

Reason	Explanation
Finds Bugs Early	Catches mistakes before the software reaches users
Ensures Quality	Makes sure the software works as expected and meets requirements
Improves User Experience	Helps create smooth, error-free user interactions
Saves Time & Money	Fixing problems early is cheaper than after release
Builds Trust	Reliable, tested software increases user and client confidence
Ensures Security	Finds vulnerabilities that could be exploited
Verifies Functionality	Confirms that every feature does what it's supposed to

⇨ Software testing is like checking your work before submitting it. It helps make sure everything is correct, safe, and ready to use.

• What types of software maintenance are there?

⇨ Software maintenance involves making changes to software after it's deployed to ensure it continues to work well, stays secure, and evolves to meet new needs.

Main Types of Software Maintenance:

1. **Corrective Maintenance** ○ **Purpose**: Fixes **bugs** or issues discovered after the software is in use.
   - **Example**: A bug where the software crashes under certain conditions is fixed.

2. **Adaptive Maintenance** ○ **Purpose**: Makes the software **compatible** with new environments or technologies (like new OS versions).
   - **Example**: Updating the software to work with a new version of a database.

3. **Perfective Maintenance** ○ **Purpose**: Improves the software by adding new features or optimizing performance.
   - **Example**: Adding a new feature to a mobile app or making it faster.

4. **Preventive Maintenance** ○ **Purpose**: Prevents future issues by improving the software's architecture or fixing potential weaknesses.
   - **Example**: Refactoring code to make it easier to maintain or adding security measures to avoid future attacks.

• What are the key differences between web and desktop applications?

⇨ Web and desktop applications both serve different purposes and have unique characteristics

⇨ Web Applications = Accessed online from anywhere, no installation, and relies on the internet.

⇨ Desktop Applications = Installed on a specific computer, works offline, and usually faster.

• What are the advantages of using web applications over desktop applications?

⇨ Web applications offer several benefits, especially in terms of accessibility, maintenance, and cost.

⇨ **Web apps** are easier to access, update, and use on any device because they are **online-based**.

⇨ **No installation** is needed, and maintenance is **simpler** for both users and developers.

• What role does UI/UX design play in application development?

⇨ UI (User Interface) and UX (User Experience) design are critical for creating applications that are functional, user-friendly, and visually appealing

⇨ **UI Design**: Focuses on **how the app looks** — the layout, colors, buttons, and icons. It makes the app **visually appealing** and easy to navigate.

⇨ **UX Design**: Focuses on **how the app feels** — ensuring it's **user-friendly**, efficient, and provides a smooth experience.

⇨ **UI** = The **appearance** of the app.
⇨ **UX** = The **experience** of using the app.

• What are the differences between native and hybrid mobile apps?

⇨ **Native Apps**: Built specifically for one platform, **faster and smoother**, but costlier and need separate development for each platform.

⇨ **Hybrid Apps**: Built for multiple platforms, **cheaper and faster** to develop, but may not be as fast or smooth.

• What is the significance of DFDs in system analysis?

⇨ Data Flow Diagrams (DFDs) are visual tools used in system analysis to represent the flow of data and how it is processed within a system. They play a crucial role in understanding and designing systems.
⇨ DFDs are like **maps** for a system, showing how **data moves** from one place to another and how it gets processed, helping everyone involved **understand** and **build** the system better.

• What are the pros and cons of desktop applications compared to webapplications?

⇨ Here's a quick comparison to highlight the **pros and cons** of both **desktop** and **web applications**

⇨ **Desktop Apps** = **Fast, powerful**, and **work offline**, but need to be **installed** and **platform-specific**.

⇨ **Web Apps** = **Accessible anywhere**, **easy to update**, but need **internet** and may be a bit **slower**.

• How do flowcharts help in programming and system design?

⇨ Flowcharts are visual diagrams that represent the sequence of steps or processes in a system or program. They are an essential tool in both programming and system design.

⇨ Flowcharts are like maps for your program or system. They help you plan, visualize, and understand how everything works before you dive into coding or designing.

Flowchart Benefits:

⇨ Simplifies Logic: Breaks down complex processes into easy steps.

⇨ Clarifies Flow: Shows how different parts of the system work together.

⇨ Improves Communication: Easy to share and discuss with teams.

⇨ Guides Programming: Helps plan the code before starting.

⇨ Identifies Errors: Makes it easier to spot mistakes early.

⇨ Enhances Documentation: Provides clear, future reference.

⇨ Aids Problem-Solving: Helps break down tasks into smaller steps.