

Frontend Assignment Set

Module 1 – Foundation

- **What is HTTP?**

HTTP (HyperText Transfer Protocol) is the way your web browser talks to websites.

Think of it like this:

- **You (the browser) want to ask a website for something.**
- **The website (the server) answers with what you asked for — like a page, image, or video.**

- **What is a Browser?**

A browser is a program (or app) that lets you look at websites on the internet.

Popular browsers include:

- **Google Chrome**
- **Mozilla Firefox**
- **Microsoft Edge**
- **Safari**
- **Opera**

1. You Type a Website Address (URL)

2. The Browser Sends a Request

3. The Website Server Responds

4. The Browser Builds the Page

- **What is Domain Name?**

A domain name is the name of a website — it's what you type in the address bar to visit a site.

 **Example:**

- **google.com**
- **facebook.com**
- **yournameblog.net**

- **What is hosting?**

- **Hosting** means **storing your website on a computer (called a server)** so that other people can visit it on the internet.
- **Think of it like this:**
- **A domain name is your website's address,**
Hosting is your website's house.

Module 2 – Fundamentals of World Wide Web

Theory Exercise

- **Difference between Web Designing and Web Developer?**

- | | |
|--|--|
| • ◆ Web Designer | • ◆ Web Developer |
| • Focuses on how the website looks | • Focuses on how the website works |
| • Designs the layout, colors, fonts, and style | • Builds the site using code |
| • Works with tools like Figma, Photoshop, or Canva | • Works with programming languages like HTML, CSS, JavaScript, and more |
| • Thinks about user experience (UX) and visual appeal | • Makes the design functional and interactive |
| • Doesn't always write code (but might know some) | • Writes the code to create web pages and features |

- **A Web Designer makes the Look of the website.**
- **A web Developer makes the Brain and Function of the website.**
- **Designing makes the design.**
- **Developer build the website using that design.**
 - **What is a W3C?**

- **W3C** stands for **World Wide Web Consortium**.
- It is the **main organization that creates rules and standards** for how the web should work.

- What is Domain?

A **domain name** is the **name of a website** — it's what you type in the address bar to visit a site.

-  **Example:**

- google.com
- facebook.com
- yournameblog.net
- .org – for organizations
- .net – for networks
- .edu – for schools
- .gov – for governments

- What SEO?

SEO stands for **Search Engine Optimization**.

- It's the process of making your website **show up higher in search results** (like on Google) so **more**

- **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**.
 - It's a **step-by-step process** used to plan, build, test, and deliver software (like apps or websites).
 - Think of it like a **recipe** for making software — it helps teams stay organized and build things **the right way**.

Phase	What Happens
1. Planning	What problem are we solving? What does the software need to do?
2. Requirement Analysis	Gather and list all the features the software must have. Create the structure, layout, and system design (like blueprints).
3. Design	Programmers write the code to build the software.
4. Development	Check for bugs and errors. Make sure everything works properly.
5. Testing	

Phase	What Happens
6. Deployment	Launch the software so people can use it.
7. Maintenance	Fix issues, add updates, and keep the software running smoothly.

Fundamentals of IT

- Explain in your own Words What a program is and How it functions?
 - to take to complete a task.
 - **How Does a Program Work?** A **program** is a set of **instructions** that a computer can follow to do a specific job.
 - It's like a **recipe** for a computer. Just like a recipe tells a cook what steps to take to bake a cake, a program tells a computer what steps
1. **You (the programmer) write instructions**
 - These instructions are written in a **programming language** like Python, Java, or C++.
 2. **The computer reads and follows the instructions**
 - The computer **does exactly what the code says**, step by step.
 3. **The result appears**

- It might show a message, do some math, display a web page, or control a robot—whatever the program is meant to do.
- What are the key steps involved in the programming process?

Step

1. 

Understand the Problem

2.  **Plan the Solution**

3.  **Write the Code**

4.  **Test the Program**

5.  **Debug if Needed**

What Happens

Know exactly **what the program needs to do**.

What is the goal or task?

Think of **how** to solve the problem. This may include writing steps (called **algorithms**) or drawing diagrams (like **flowcharts**).

Use a programming language (like Python, Java, C++) to **write the actual instructions** for the computer.

Run the program and see if it works. Fix any **bugs (errors)** you find.

If something goes wrong, **find and fix** the issue in the code. This is called **debugging**.

Step

6. 
**Run/Execute
the Program**



What Happens



After testing and fixing,
you **run the final version**
for actual use.



7. 
**Maintain and
Update**

Over time, you may need
to **add features, improve
performance, or fix new
bugs**. That's called
maintenance.

- What are the main difference between high-level and low-level programming language?

Aspect	High-Level Language 	Low-Level Language 
Definition	Easy-to-read languages close to human language	Close to machine language, harder for humans to read
Examples	Python, Java, C++, JavaScript	Assembly, Machine Code
Readability	Easy to understand	Hard to read, looks

Aspect	High-Level Language 	Low-Level Language 
	read and write	like numbers or symbols
Speed	Slower to run (but easier to write)	Very fast and efficient
Control over hardware	Less control over hardware	Direct control over memory and CPU
Used by	Most app and web developers	System programmers, embedded developers
Translation	Needs a compiler or interpreter	May need an assembler

Aspect	High-Level Language 	Low-Level Language 
Portability	Runs on many systems with little change	Tied to specific hardware

- 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)

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

Example: When you type `www.google.com` in your browser, your 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 requests and sends back the Google homepages.

- **Explain the functions of the TCP/IP model and its layers?**

TCP/IP stands for Transmission Control Protocol / Internet Protocol.

It is a set of rules (protocols) that allows computers to communicate over the internet or a network.

Think of it as the language and rules computers use to send, receive, and understand data.



TCP/IP Model Layers and Their Functions
The TCP/IP model has 4 main layers, each with a specific job:

1. Application Layer (Top Layer)



What it does:

- **Allows users (like you) to interact with the network through apps like browsers or email.**



Functions:

- Provides services like web browsing, email, file transfer.
 - Handles HTTP, FTP, DNS, SMTP, etc.
 - ◆ Example:
 - You type www.google.com in Chrome (Application Layer is involved).
-

2. Transport Layer

- ◆ What it does:
 - Makes sure the data is sent reliably and in the correct order.
 - ◆ Functions:
 - Breaks big data into small pieces called segments.
 - Makes sure all pieces arrive and are put back together correctly.
 - Uses protocols like:
 - TCP (reliable, ordered)
 - UDP (faster, but less reliable)
 - ◆ Example:
 - Watching a YouTube video uses UDP (speed matters).
 - Sending an email uses TCP (accuracy matters).
-

3. Internet Layer

- ◆ What it does:

- **Helps data find the best path to its destination.**

- ◆ **Functions:**

- **Adds IP addresses to data so it knows where to go.**
- **Uses IP protocol to route data between networks.**

- ◆ **Example:**

- **Like writing the address on a letter so the post office knows where to send it.**

4. Network Access Layer (also called Link or Physical Layer)

- ◆ **What it does:**

- **Handles the physical connection between devices.**
- **Sends raw data over wires, Wi-Fi, or other media.**

- ◆ **Functions:**

- **Deals with hardware: cables, Wi-Fi, switches, routers.**
- **Uses MAC addresses, Ethernet, etc.**

- ◆ **Example:**

- **The part of the network that actually moves the data between devices.**

- **Explain Client Server Communication?**
Client-Server Communication is how two computers — the client and the server — talk

to each other over a network (like the internet).

Who Are the Client and Server?

- **Client:** The user's device (like your web browser, mobile app, or computer)
 - **Server:** A powerful computer that stores websites, data, and services
-

How Client-Server Communication Works (Step-by-Step):

1. The Client Sends a Request

- You type `www.google.com` into your browser.
- Your browser (the client) sends a request to Google's server:
"Please send me the homepage."

2. The Server Receives the Request

- The server looks for the requested web page or data.

3. The Server Sends a Response

- It sends the requested files (like HTML, images, CSS) back to the browser.

4. The Client Displays the Result

- Your browser reads the data and shows you the website.

- How does broadband differ from fibre -optic internet?

Broadband is a general term for high-speed internet.

It includes several types of internet connections:

- DSL (uses telephone lines)
- Cable (uses TV cables)
- Fiber-optic (uses glass cables and light signals)

- What are the difference between HTTP and HTTPS protocols?

	HTTP (HyperText Transfer Protocol)	HTTPS (HyperText Transfer Protocol Secure)
Feature		
Security	Not secure; data is transmitted in plain text.	Secure; data is encrypted using SSL/TLS

	HTTP (HyperText Transfer Protocol)	HTTPS (HyperText Transfer Protocol Secure)
Feature		
Port Number	Uses port 80 by default.	Uses port 443 by default.
URL Format	Starts with http://	Starts with https://
Encryption	No encryption; vulnerable to attacks like eavesdropping.	Uses SSL/TLS to encrypt data for confidentiality.
Data Integrity	No protection against	Ensures data integrity

Feature	HTTP (HyperText Transfer Protocol)	HTTPS (HyperText Transfer Protocol Secure)
	tampering.	and protection against tampering.
Authenti cation	No verificati on of the server identity.	Verifies the server using an SSL certificat e.
Use Case	Suitable for non- sensitive content.	Required for sensitive data (e.g., login,

	HTTP (HyperText Transfer Protocol)	HTTPS (HyperText Transfer Protocol Secure) payment).
Feature		

- **What is the role of encryption in securing application , software application and its types?**

Encryption plays a **crucial role** in securing software applications by protecting data at various stages—when it's stored, transmitted, or processed. It ensures:

- **Confidentiality** – Converts readable data (plaintext) into ciphertext, ensuring only authorized parties can decrypt it. [thelegalschool.in+3reddit.com+3esecurityplanet.com+3cloudflare.com+5investopedia.com+5fortinet.com+5](#)
- **Integrity** – Detects unauthorized changes via techniques like authenticated encryption and hashing.
- **Authentication & Non-repudiation** – Uses digital signatures and certificates to verify identity and prevent denial of actions. [thelegalschool.in+1en.wikipedia.org+1](#)
- **Secure communication** – Protects data in transit using protocols like TLS/SSL, ensuring safe online interactions. [reddit.com](#)

- **What are software application?**
- A **software application**—often called an *application* or *app*—is a type of computer program specifically designed to help users perform tasks or achieve particular goals, such as writing documents, browsing the web, editing photos.
 - What is the difference between system software and application software?

1. Purpose & Role

- **System Software** interfaces with hardware and provides a platform for all other software to run—examples include operating systems, device drivers, and utilities
[lifewire.com+12geeksforgeeks.org+12blaze.tech+12](#).
 - **Application Software** is designed for end-users to perform specific tasks—e.g., word processing, browsing, or gaming—operating on top of system software
[en.wikipedia.org+1lifewire.com+1](#).
-

2. Execution Time & Dependency

- **System software** loads at startup and runs continuously in the background; it can operate independently of user actions .
 - **Application software** only runs when launched by the user and stops when closed; it requires system software to function
[reddit.com+8uk.indeed.com+8geeksforgeeks.org+8](#).
-

3. User Interaction

- **System Software** works with minimal user interaction, handling resource management and low-level operations [en.wikipedia.org](#).

- **Application Software** features direct user interfaces (GUIs/CLIs) for tasks—users interact with these regularly
en.wikipedia.org+9techtarget.com+9geeksforgeeks.org+9.
-

4. Programming Level

- **System Software** is often written in **low-level languages** (like C, C++, Rust, or assembly) to manage hardware efficiently
tateeda.com+4geeksforgeeks.org+4uk.indeed.com+4.
 - **Application Software** uses **high-level languages** (e.g., Java, Python, Swift), making it easier to build user-focused functionality .
-

5. Installation & Removal

- **System Software** is typically **pre-installed** and essential for device operation; removing it can render a system unusable
en.wikipedia.org+12geeksforgeeks.org+12lifewire.com+12.
- **Application Software** is **optional**; users install and uninstall based on needs without affecting underlying operation .

➤ Why are layer important in software application?

Layer in software architecture help organize the code into separate parts each with a specific role.

➤ Common layer in software application:

- 1.Presentation Layer: ul, user interaction
- 2.Business Layer: App rule and operations
- 3.Database Layer: stores the actual data

- Explain the important of a development environment in software application?

A **development environment** is a dedicated setup where developers **write, build, test, and debug** software before it reaches production. It plays a **critical role** in ensuring software quality and reliability. Here's why it matters.

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

1. Purpose & Role

- **System Software** interfaces with hardware and provides a platform for all other software to run—examples include operating systems, device drivers, and utilities

[lifewire.com+12geeksforgeeks.org+12blaze.tech+12](#).

- **Application Software** is designed for end-users to perform specific tasks—e.g., word processing, browsing, or gaming—operating on top of system software [en.wikipedia.org+1lifewire.com+1](#).

- Why is version control important in software development?

Version control is absolutely essential in modern software development—it offers much more than just "tracking files.

- 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.

➤ What are the difference 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.

➤ Aspect	➤ Open Source	➤ Proprietary
➤ Source Code	➤ Accessible & modifiable	➤ Hidden from users
➤ Cost	➤ Free or minimal support costs	➤ Paid licenses/subscriptions
➤ Customization	➤ Fully customizable	➤ Limited by vendor
➤ Support	➤ Community + third-party	➤ Vendor-provided (often SLA-backed)

➤ Aspect	➤ Open Source	➤ Proprietary
➤ Security	➤ Publicly audited	➤ Vendor-managed patch cycles
➤ Lock-in	➤ No (can fork)	➤ Yes—dependent on vendor
➤ Updates	➤ Frequent, community-driven	➤ Regular, vendor-scheduled updates
➤ Licensing	➤ GPL, MIT, Apache, etc.	➤ Software EULAs with restrictions

- 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 business?
 - 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 stages?
 - The software development process is a step-by-step method used to create, test, and deliver software.
- 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 limit
 - ☐ 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 us


- What are the key elements of system design?
 - 1. Requirements & Problem Understanding**
- **Functional and non-functional requirements (e.g., performance, security, availability) must be clearly defined, measurable, and relevant to user needs**
[moldstud.com+15dev.to+15reddit.com+15geeksforgeeks.org.](#)
- **A solid grasp of the problem domain, user personas, workflows, and constraints is critical before starting design** [dev.to.](#)
- ---
- ☐ **2. Architecture & Structural Design**


- High-level architectural pattern, such as layered (n-tier), microservices, or monolithic design, forms the backbone of your system [dev.to+1reddit.com+1](#).
- Modularity, separation of concerns, encapsulation, and loose coupling enable maintainability and flexibility
[reddit.com+14geeksforgeeks.org+14en.wikipedia.org+14](#).


-
-  **3. Scalability & Performance**
 - Design for vertical and horizontal scaling, using load balancing, caching, and database sharding
[medium.com+3geeksprogramming.com+3dev.to+3](#).
 - Manage latency, throughput, and optimize resource usage through profiling and performance tuning
[geeksprogramming.com](#).


-
-  **4. Reliability, Availability & Fault Tolerance**
 - Incorporate redundancy, failover mechanisms, and data replication strategies to ensure uptime
[reddit.com+10geeksprogramming.com+10reddit.com+10](#).
 - Plan for fault tolerance—design systems to continue operating gracefully even when components fail
[geeksforgeeks.org+12geeksprogramming.com+12reddit.com+12](#).

-
-  **5. Security & Privacy**
 - Implement authentication, authorization, encryption, and input validation to protect data and services
[askhandle.com+1simplicable.com+1](#).
 - Follow the principle of least privilege, and perform threat modeling and security testing early and often .
-

-  **6. Data Management & Storage**
- Select between SQL vs NoSQL, plan for partitioning/sharding, replication, indexing, and data modeling aligned with data access patterns newsletter.systemdesigncodex.com.
- Design API contracts and data formats (e.g., JSON, Protobuf) thoughtfully for consistency and versioning
- .

-
-  **7. Integration & Communication**
 - Define interfaces, APIs, or messaging protocols. Use tools like REST, GraphQL, gRPC, or message queues to establish clear service contracts dev.to+2reddit.com+2reddit.com+2.
 - For microservices and distributed systems, manage service discovery, API gateways, and message buses newsletter.systemdesigncodex.com.

-
-  **8. Observability & Monitoring**
 - Implement monitoring with logs, metrics, and tracing (e.g., Prometheus, ELK, Jaeger) to diagnose issues and understand system health newsletter.systemdesigncodex.com.
 - Use alerts and dashboards to proactively respond to anomalies.

-
-  **9. Maintainability & Extensibility**
 - Strive for clean code, modular design, abstraction, and documentation so new features can be added with minimal friction medium.com.
 - Ensure the system is extensible, allowing plug-ins or future modules without big rewrites en.wikipedia.org.
-

- **✓ 10. Simplicity & Flexibility**
- **Prefer the KISS (Keep It Simple, Stupid) principle—avoid unnecessary complexity [reddit.com](https://www.reddit.com).**
- **Iterate in small steps, validate early, and adapt incrementally instead of overdesigning upfront .**
- ---
- **§ 11. Cost & Operational Feasibility**
- **Analyze cost trade-offs for hosting (cloud vs on-prem), compute, storage, and network .**
- **Plan the deployment model—containerization, orchestration (e.g., Kubernetes), CI/CD pipelines—to support reliability and scalability .**

- **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 o **Purpose:** Fixes bugs or issues discovered after the software is in use.

o **Example:** A bug where the software crashes under certain conditions is fixed.

2. Adaptive Maintenance o **Purpose:** Makes the software compatible with new environments or technologies (like new OS versions).

o **Example:** Updating the software to work with a new version of a database.

4. Perfective Maintenance o Purpose: Improves the software by adding new features or optimizing performance.

o Example: Adding a new feature to a mobile app or making it faster.

4. Preventive Maintenance o Purpose: Prevents future issues by improving the software's architecture or fixing potential weaknesses.

o 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 web applications?

Pros:

- **High Performance & Speed:** Leverage local CPU/GPU for resource-heavy tasks like video editing or gaming wired
- **Offline Functionality:** Fully usable without internet, ideal for rem.
- **Rich Feature Set:** Deep system integration allows advanced capabilities and customization .
- **Strong Data Privacy:** User data stays local, reducing exposure to online threats.
- **Stable User Experience:** Consistent UI/performance not reliant on internet speed.

Cons:

- **Platform Dependence:** Separate versions needed for Windows.
- **Installation & Updates:** Manual install and update processes can be cumbersome .
- **Limited Collaboration:** Harder to enable real-time multi-user features .

- **How do flowchart help in programming and system design?**
Flowcharts are powerful visual tools that significantly aid programming and system design. Here's how they support developers and architects:

1. Plan & Understand Complex Logic

- **Visualizing logic:** Flowcharts map out the flow of control with decision points, loops, and branches, helping you grasp algorithm structure before writing code
[mundrisoft.com+8createely.com+8phpizabi.net+8.](#)
- **Algorithm refinement:** They make inefficiencies and logic flaws apparent early—such as unhandled cases or unnecessary steps [mundrisoft.com.](#)

2. Modular Thinking & Efficient Coding

- **Modular decomposition:** Breaking logic into discrete flowchart segments encourages modular design, enhancing clarity and reusability .
- **Code blueprint:** Once the flowchart is complete, translating it into code becomes more straightforward and error-free
[createely.com+15getuplearn.com+15mundrisoft.com+15.](#)

3. Debugging & Problem-Solving

- **Trace execution visually:** Flowcharts simplify tracking the flow of data and finding where logic breaks down .

- **Spot bottlenecks:** Visual layouts make it easier to identify inefficiencies in process paths .

4. Communication & Documentation

- **Universal understanding:** They bridge gaps between technical and non-technical stakeholders, serving as a common reference [correototal.com+8createy.com+8dovetail.com+8.](#)
- **Documentation lifeline:** Flowcharts serve as living documents, improving onboarding and system comprehension [medium.com+6getuplearn.com+6createy.com+6.](#)

5. Collaboration & Architecture Design

- **Team alignment:** Shared diagrams help align everyone's understanding before coding begins [phpizabi.net+3algocademy.com+3reddit.com+3.](#)
- **High-level overviews:** Especially useful in system design, flowcharts outline workflows and component interactions [reddit.com+2dovetail.com+2reddit.com+2.](#)

•

