**Frontend assignment set**

**Module 1 – Foundation**

* **What is HTTP?**
* **Http** stands for **Hypertext Transfer Protocol**. It's the foundational protocol used by the **World Wide Web** to enable communication between web browsers (clients) and web servers.
* **What is a Browsers? How they works?**
* A **browser** (short for *web browser*) is a **software application** used to access and view websites on the internet. Examples of browsers include **Google Chrome**, **Mozilla Firefox**, **Microsoft Edge**, and **Safari**.
* You type www.google.com in the address bar.
* Browser Sends a Request on internet.
* It ask the server for the page.
* The server send back the page.
* The browser shows the website to you.
* **What is Domain Name?**
* A **domain name** is the **human-friendly address** of a website that you type into a browser to visit a site,like www.google.com.
* Example:-www.example.com
* www: stand for world wide web(optional)
* example:- this is a unique name of the website.
* .com:- Top-level domain (TLD, like .com, .org, .in)
* **What is hosting?**
* Hosting means A place where your website "lives" on the internet.
* **Example:-** 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.

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

**• Difference between Web Designer and Web Developer**

| **Point of Comparison** | **Web Designer** | **Web Developer** |
| --- | --- | --- |
| **Definition** | A web designer is responsible for designing the look and feel of a website. | A web developer is responsible for building and maintaining the actual website using coding languages. |
| **Main Focus** | Focuses on the visual appearance and user experience (UX/UI). | Focuses on how the website works and functions behind the scenes. |
| **Skills Required** | Graphic design, color theory, layout design, user interface design (UI), and tools like Adobe XD or Figma. | Programming skills in HTML, CSS, JavaScript, PHP, Python, etc., and knowledge of databases and frameworks. |
| **Tools Used** | Design tools like Figma, Photoshop, Illustrator, Sketch. | Coding tools like Visual Studio Code, Git, Sublime Text. |
| **Type of Work** | Creates layouts, color schemes, buttons, and website structure. | Writes code to bring the design to life and ensure functionality. |
| **Goal** | To make the website attractive and easy to use. | To make the website work properly, load fast, and be bug-free. |
| **Output** | Delivers design mockups or prototypes. | Delivers a fully working website based on those designs. |

A **web designer** focuses on how the website looks, while a **web developer** focuses on how the website works. Both work together to create a complete and functional website.

* **What is a W3C?**
* **W3C is the main international organization that creates and maintains web standards to ensure the long-term growth of the Web.**
* It was founded in **1994** by **Tim Berners-Lee**, the inventor of the World Wide Web.
* **What is Domain?**
* A **domain** (or **domain name**) is the **unique address** of a website on the internet that people type in their web browser to visit a site.
* **Example:**
* www.google.com  
  www.facebook.com  
  www.yourname.in
* These are all **domain names**.

**Common Domain Extensions:**

* .com – Commercial (most popular)
* .org – Organizations
* .net – Network services
* .Edu – Educational institutions
* .in – India-specific
* .gov – Government websites
* **What is SEO?**
* **SEO** stands for **Search Engine Optimization**.
* SEO is the process of **improving a website** so that it appears **higher in search engine results** (like Google) when people search for something.
* If you have a website and want more people to find it on Google, you use **SEO** techniques to make sure it shows up on the **first page** of search results.
* **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**.
* The **SDLC life cycle** is a step-by-step **process used to plan, develop, test, and maintain software systems**. It helps developers create high-quality software in a structured and efficient way.

| **Phase** | **Description** |
| --- | --- |
| 1.**Requirement Gathering** | Understand what the user wants. Collect all the needs for the software. |
| 2. **Planning** | Decide how the project will be done (time, cost, resources). |
| 3. **Design** | Create the structure or blueprint of the software (UI design, database design, etc.). |
| 4.**Development (Coding)** | Developers write the actual code based on the design. |
| 5. **Testing** | Check the software for bugs and errors. Make sure everything works correctly. |
| 6. **Deployment** | Deliver the software to the user or client for real-world use. |
| 7. **Maintenance** | Fix issues, update features, and support the software after launch. |

**Fundamentals of IT**

* **What is a program?**
* A **program** is a **set of instructions** written in a programming language that a **computer can understand and execute** to perform a specific task.
* A **program** tells a computer **what to do** and **how to do it**, 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):

**1. Define the Problem**

* Understand what needs to be solved.  
  Example: "Create a calculator that adds two numbers."

**2. Plan the Solution**

* Think about **how** the program will work.  
  Use tools like **flowcharts** or **pseudocode** to map the logic.

**3. Write the Code**

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

**4. Test the Program**

* Run the program with different inputs to check if it gives the **correct results**.

**5. Debug the Program**

* Find and fix **errors or bugs** in the code.  
  These could be:
* **Syntax errors** (wrong code structure)
* **Logic errors** (wrong output)
* **Runtime errors** (crashes)

**6. Document the Code**

* Write helpful **comments** in the code.  
  Create user manuals or guides so others can understand and use your program.

**7. Maintain and Improve**

* After release, **fix new bugs**, **update features**, or **improve performance** as needed.
* **What are the main differences between high-level and low-level programming languages?**

| **Future** | **High-Level Language** | **Low-Level Language** |
| --- | --- | --- |
| **Definition** | Easy-to-read language close to human speech | Language closer to machine code |
| **Examples** | Python, Java, C++, JavaScript | Assembly, Machine Code |
| **Ease of Use** | Easy to learn and write | Difficult to learn and write |
| **Readability** | Human-readable | Hard to understand |
| **Abstraction Level** | High (hides hardware details) | Low (directly interacts with hardware) |
| **Speed of Execution** | Slower (more processing steps) | Faster (directly runs on hardware) |
| **Portability (can run on different machines)** | High | Low (machine-specific) |
| **Used For** | Application/software development | System-level tasks like OS or drivers |

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

**Client:**

* The client is the user's device (like a computer, phone, or tablet) that requests information from the web.
* It sends a request to the server using a web browser (like Chrome or Firefox).
* Example: When you type www.google.com, your browser (the client) asks Google's server for the web page**.**

**Server:**

* The server is a powerful computer that stores websites, files, and data.
* It receives the client’s request, processes it, and sends back the correct information (like a web page or file).
* Example: Google's server sends the search page back to your browser when you request it.

**How They Work Together:**

1. Client → sends a request (e.g., asking for a web page)
2. Server → receives the request, finds the correct data
3. Server → sends the response (e.g., the web page content)
4. Client → displays the result in the browser

* **Explain the function of the TCP/IP model and its layers.**

TCP/IP stands for Transmission Control Protocol/Internet Protocol.  
It is a set of rules that allows computers to communicate over the internet or any network.

* Function of TCP/IP Model:
* It breaks down data into smaller parts (called packets).
* Sends those packets from one computer to another.
* Ensures the data arrives correctly and in order.
* Helps different devices and systems understand each other on a network.

📚 Layers of the TCP/IP Model (4 Layers)

| Layer No. | Layer Name | Function |
| --- | --- | --- |
|  | Application Layer | Provides services like email, web browsing (e.g., HTTP, FTP, SMTP).  It’s what the user interacts with. |
|  | Transport Layer | Breaks data into smaller pieces (segments), ensures reliable delivery.  Uses TCP or UDP protocols. |
|  | Internet Layer | Chooses the best path for data to travel.  Adds IP addresses to identify sender and receiver. |
|  | Network Access Layer | Sends data over the physical network (like cables, Wi-Fi).  Deals with hardware and device drivers. |

* **Explain Client Server Communication:**

Client-server communication is the way two computers (a client and a server) talk to each other over a network, like the Internet.

* **Client**
* A client is usually the user's device (computer, phone, browser).
* It sends a request for data or a service.
* Example: A web browser asking to view a website.
* **Server**
* A server is a powerful computer that stores websites, data, files, or applications.
* It receives the client’s request, processes it, and sends back a response.
* **How It Works (Step-by-Step):**

1. Client sends a request (e.g., "Show me [www.example.com](http://www.example.com)")
2. Server receives the request
3. Server processes the request (looks for the website or file)
4. Server sends a response (the web page or data)
5. Client receives and displays the information (in the browser)

* **How does broadband differ from fiber-optic internet?**

**Broadband Internet:**

* Broad term that means any high-speed internet connection.
* Includes different types:
  + DSL (uses telephone lines)
  + Cable (uses TV cables)
  + Fiber-optic
  + Satellite
* Speeds vary depending on the type.
* More common and widely available.

**Fiber-Optic Internet:**

* A specific type of broadband.
* Uses thin glass or plastic fibres to send data as light signals.
* Offers very fast speeds, low delay, and high reliability.
* Best for video streaming, gaming, and large downloads/uploads.
* **What are the differences between HTTP and HTTPS protocols?**

| **Feature** | **HTTP** | **HTTPS** |
| --- | --- | --- |
| **Full Form** | Hypertext Transfer Protocol | Hypertext Transfer Protocol Secure |
| **Security** | ❌ Not secure | ✅ Secure (uses encryption) |
| **Encryption** | No encryption | Uses **SSL/TLS** to encrypt data |
| **Data Protection** | Data can be intercepted by hackers | Data is protected and safe |
| **URL Format** | Starts with http:// | Starts with https:// |
| **Padlock Icon** | 🔓 No padlock in browser | 🔒 Shows padlock icon in browser |
| **Used For** | Basic websites or internal networks | Banking, shopping, login pages, etc. |
| **Trust Level** | Low | High (more trusted by users & browsers) |

* HTTP: Sends your data like a postcard – anyone can read it.
* HTTPS: Sends your data like a sealed envelope – private and protected.
* **What is the role of encryption in securing application, Software Applications and Its Types.**

Encryption is the process of converting readable data into unreadable code to protect it from unauthorized access.

* **Protects Sensitive Data**
  + Keeps passwords, personal info, and payment details safe.
  + Even if hackers steal the data, they can’t read it.
* **Ensures Privacy**
  + Prevents others from seeing what users are doing in apps (e.g., messaging apps, banking apps).
* **Secures Communication**
  + Encrypts data sent between client and server (like HTTPS).
  + Used in emails, video calls, and online forms.
* **Prevents Data Tampering**
  + Ensures that data hasn’t been changed during transfer.
  + Protects the integrity of files and messages.
* **Builds Trust**
  + Users feel safer using apps that protect their information.
  + Required for compliance with data protection laws (like GDPR).
* **Software Applications and Its Types**

A **software application** is a program designed to help users perform specific tasks.

* **Types of Software Applications:**

| **Type** | **Description** | **Examples** |
| --- | --- | --- |
| **Productivity Software** | Helps users create documents, spreadsheets | MS Word, Excel, Google Docs |
| **Web Applications** | Runs in a browser, needs internet | Gmail, YouTube, Facebook |
| **Mobile Applications** | Runs on smartphones and tablets | WhatsApp, Instagram, TikTok |
| **Desktop Applications** | Installed on computers | VLC Media Player, Photoshop |
| **Enterprise Software** | Used by businesses to manage operations | SAP, Salesforce, Oracle |
| **Utility Software** | Helps manage system resources | Antivirus, Disk Cleanup |
| **Educational Software** | Used for learning and training | Duolingo, Khan Academy |
| **Game Software** | Entertainment through interactive games | PUBG, Minecraft, Candy Crush |

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

Here’s a simple and clear explanation of the difference between system software and application software:

| **Feature** | **System Software** | **Application Software** |
| --- | --- | --- |
| **Definition** | Software that runs and manages the hardware and basic system operations | **Software that helps the user perform specific tasks** |
| **Purpose** | Supports the computer's core functions | Helps the user do work or enjoy entertainment |
| **User Interaction** | Runs in the background, not directly used by the user | Directly used by the user |
| **Examples** | Operating systems (Windows, Linux), device drivers, utilities | MS Word, Chrome, WhatsApp, Photoshop |
| **Installation** | Comes pre-installed or needed for system to run | Installed by user as needed |
| **Dependency** | Required for the system to work properly | Depends on system software to run |

* **System Software**: Runs the computer (like a manager).
* **Application Software**: Lets you do tasks (like writing, browsing, gaming).
* **What is the significance of modularity in software architecture?**

Modularity means breaking a software system into independent, smaller parts called modules, where each module performs a specific task.

* **Significance of Modularity:**

**1. Easier to Understand**

* Each module handles a specific part of the system.
* Developers can focus on one module at a time.

**2. Improves Code Reusability**

* Modules can be reused in other programs or projects.
* Example: A login module can be reused in many apps.

**3. Simplifies Debugging and Testing**

* Easier to find and fix bugs in smaller modules.
* You can test modules individually (unit testing).

**4. Enhances Team Collaboration**

* Teams can work on different modules at the same time.
* Increases productivity and speeds up development.

**5. Supports Maintainability**

* You can update or change one module without affecting the whole system.
* Makes long-term maintenance easier.

**6. Encourages Scalability**

* Easier to add new features by creating or updating modules.
* The system grows in a controlled and organized way.

**7. Promotes Separation of Concerns**

* Each module focuses on **one task only**, reducing complexity.

Modularity is like building with LEGO blocks — small, separate pieces that can be built, tested, and replaced easily without breaking the whole structure.

* **Why are layers important in software architecture?**

Layers in software architecture organize a system into separate levels, where each layer has a specific role. This structure is called a layered architecture.

**Importance of Layers in Software Architecture:**

**1. Separation of Concerns**

* Each layer handles a specific function (like UI, logic, or data).
* This makes the system easier to understand and manage.

**2. Improves Maintainability**

* Changes in one layer (like updating the UI) can often be made **without affecting other layers**.

**3. Supports Reusability**

* Common functionality (like data access) can be reused across multiple parts of the application.

**4. Easier Testing and Debugging**

* Individual layers can be tested separately (unit testing).
* Easier to locate bugs in a specific part of the system.

**5. Scalability**

* Layers help scale specific parts (e.g., adding more servers for the database layer) without changing the whole system.

**6. Team Collaboration**

* Different teams can work on different layers (e.g., frontend team on UI, backend team on logic).

**7. Security and Control**

* Sensitive operations (like data storage or authentication) can be isolated in secure layers.

**📚 Example of Common Layers:**

| **Layer Name** | **Purpose** |
| --- | --- |
| **Presentation Layer** | User Interface (UI) – what users see |
| **Business Logic Layer** | Processes data, handles rules |
| **Data Access Layer** | Connects to databases/files |
| **Database Layer** | Stores and retrieves data |

Layers act like floors in a building — each floor has its job, and together they form a strong structure.

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

A development environment is the setup of tools, software, and systems that developers use to write, test, debug, and build software. It acts like a workspace for programmers.

**Why Is It Important?**

**1. Provides the Right Tools**

* Includes code editors, compilers, debuggers, and version control.
* Helps developers write, run, and test code all in one place.

**2. Improves Productivity**

* Features like **auto-complete**, **syntax highlighting**, and **error detection** make coding faster and easier.

**3. Supports Testing and Debugging**

* Allows developers to test their programs and fix bugs before releasing the final product.

**4. Enables Team Collaboration**

* With version control tools (like Git), multiple developers can work on the same project without conflicts.

**5. Simulates Real Conditions**

* Helps developers see how their software will run on different devices or operating systems before it goes live.

**6. Keeps Code Safe and Organized**

* Tracks changes, backs up code, and keeps everything organized in one place.

**7. Helps with Deployment**

* Some environments include tools to **package and deploy** the software to users or servers.

A development environment is like a toolbox and workshop for software developers — it provides everything needed to build, test, and polish software efficiently and correctly.

* **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 essential in software development for the following reasons:

**1.Tracks Changes**

* It keeps a history of every change made to the code.
* Developers can view what was changed, who changed it, and why it was changed.

2. **Supports Collaboration**

* Multiple developers can work on the same project at the same time.
* Version control merges their work efficiently and resolves conflicts.

3. **Enables Experimentation**

* Developers can create branches to try out new features or fix bugs without affecting the main code.
* If something breaks, they can simply go back to a working version.

**4.** **Simplifies Debugging**

* When bugs appear, developers can review the commit history to find when and where the bug was introduced.

**5**. **Provides Backup**

* The entire codebase is stored and versioned, reducing the risk of accidental data loss.

**6**. **Ensures Accountability**

* Each change is logged with the developer’s name and comments, making it easy to review and audit.

**7. Supports Deployment**

* Helps maintain different versions of the software (e.g., beta, production).
* Developers can release updates and patches cleanly and reliably.