Assignment 1 THEORY EXE.	Module 1 – Overview of IT Industry

1. What is Program?

THEORY EXERCISE: Explain in your own words what a program is and how it functions.

- **ANS :**

A program is a set of instructions written in some different languages to tell a computer device that what it have to compute.
HOW IT FUNCTIONS:

As we know a computer knows the Binary Language. But we can't explain big calculations to the computer in Binary language. So some specific developers developed some programming languages (like JAVA, PYTHON, PHP, C, C++, C# etc.) that can be understandable by both HUMAN and COMPUTERS.

There are some main functions of these programming languages:

- Input
- Processing
- Output
- Manage (Storage, Execution)

2. What is Programming?

THEORY EXERCISE: What are the key steps involved in the programming process?

- **ANS :**

There are few steps in writing the program and run and compilation.

The first step is to choose the programming language, in which you would like to solve the problem (EX; JAVA, PYTHON, C, C++, C#).

Then according to the chosen language you need to choose the platform or a complier on which you can execute your program code (EX; V.S. CODE, IDLE(for python), INTELLIJ IDEA (for java), TURBO C & C++).

After that write the code in the chosen programming language. Implement the plan & algorithms step-by-step.

Compile the code to identify syntax errors. Debug these errors. You can also use debugging tools given by the platform or compiler you use. And test your program again.

So, these are the main steps in programming. You can also document the code and improve as your requirement.

3. Types of Programming Languages

THEORY EXERCISE: What are the main differences between high-level and low-level programming languages?

- **ANS:**

The main difference is that the High-Level languages are Human friendly. They are closer to human language, making them easier to read, write, and understand and the Low-Level languages are Machine friendly. They are closer to the computer's hardware and more difficult for humans to read and write.

Example of High-Level languages: Java, Python, PhP, C, C++, C#
Example of Low-Level languages: Binary language

4. World Wide Web & How Internet Works

THEORY EXERCISE: Describe the roles of the client and server in web communication.

- **ANS:**

The main difference is that the High-Level languages are Human friendly. They are closer to human language, making them easier to read, write, and understand and the Low-Level languages are Machine friendly. They are closer to the computer's hardware and more difficult for humans to read and write.

Example of High-Level languages: Java, Python, PhP, C, C++, C#
Example of Low-Level languages: Binary language

5. Network Layers on Client and Server

THEORY EXERCISE: Explain the function of the TCP/IP model and its layers.

- **ANS:**

The TCP/IP model is Transmission Control Protocol/Internet Protocol. It is a conceptual framework used to enable communication between computers over a network.

--> There mainly 4 layers of TCP/IP model

*Application Layer
*Transport Layer
*Internet Layer
*Network Access Layer

6. Client and Servers

THEORY EXERCISE: Explain Client Server Communication

- **ANS:**

Client-server communication is a model where two entities, a client and a server, interact over a network. The client makes requests, and the server processes and responds to these requests. This architecture underpins many internet-based applications, such as web browsing, email, and file sharing.

7. Types of Internet Connections

THEORY EXERCISE: How does broadband differ from fiber-optic internet?

- **ANS:**

Broadband and fiber-optic internet are two types of internet connection technologies.

Broadband is a generic term that refers to high-speed internet connections using various technologies, including DSL (Digital Subscriber Line), cable, satellite, and fiber. Speeds range from 1 Mbps to 500 Mbps, depending on the technology used

Uses electric signals to transmit data over copper or coaxial cables. Higher latency compared to fiber, especially in satellite and DSL connections. Generally more affordable upfront, with lower installation and monthly costs.

Fiber-optic internet is a specific type of broadband connection that uses fiber-optic cables made of thin strands of glass or plastic to transmit data as light signals. Offers significantly faster speeds, often up to 1 Gbps (Gigabit per second) or higher. Transmits data as light signals through fiber-optic cables. Offers low latency, making it ideal for online gaming, video conferencing, and other latency-sensitive activities. Higher installation and subscription costs due to advanced technology and infrastructure requirements.

8. Protocols

THEORY EXERCISE: What are the differences between HTTP and HTTPS protocols?

- **ANS:**

HTTP is Hyper Text Transfer Protocol. It is a protocol for transmitting hypertext over the internet. In this there is no encryption (Data transmitted is in plain text) . Risky for sensitive information such as login credentials, payment details, or personal data. It's URL is **' http:// '** . (EXAMPLE; http://google.com)

HTTPs is Hyper Text Transfer Protocol Secure. An extension of HTTP that uses encryption to secure data transmission. It uses encryption in communication. Data is encrypted and cannot be read by unauthorized parties, so it safe to use in sharing personal data in logging like processes. It's URL is **' https:// '.**
(EXAMPLE; https://google.com)

9. Application Security

THEORY EXERCISE: What is the role of encryption in securing applications?

- **ANS:**

Encryption plays a critical role in securing applications by converting data into a coded format to protect it from unauthorized access. In a communication, when a user sends data to the other user, it converts the data into miscellaneous codes. It ensures that even if data is intercepted or accessed by malicious actors, it cannot be understood or misused without the appropriate decryption key. Encryption is essential for maintaining the confidentiality, integrity, and authenticity of data in applications.

10. Software Applications and Its Types

THEORY EXERCISE: What is the difference between system software and application software?

- **ANS:**

System software and application software are two types of computer software.

System Software: Software designed to manage and control computer hardware and provide a platform for running application software. It acts as an interface between hardware and user-level software. In simple words Operating Systems are the System software. Operating systems (e.g., Windows, macOS, Linux). Also some Utility programs like antivirus software are also the examples of System Software.

Application Software: These are the software developed to help users perform specific tasks or activities. It runs on top of system software and is tailored to user needs. Tools like Microsoft Word, Excel Google Docs etc. are the examples of Application software. Also web browsers like Firefox, Tor Browser, Google Chrome etc. are the Application Software.

11. Software Architecture

THEORY EXERCISE: What is the significance of modularity in software architecture?

- **ANS:**

Here is why modularity is significant in software architecture:

1. Improved Maintainability (Maintenance after development)
2. Scalability and Flexibility
3. Cost Efficiency
4. Reduced Risk and Higher the Security

12. Layers in Software Architecture

THEORY EXERCISE: Why are layers important in software architecture?

- **ANS:**

There are mainly 5 layers in Software Architecture;

1. Presentation Layer (User Interface Layer)
2. Business Logic Layer (Application Layer)
3. Domain layer
4. Infrastructure layer
5. Database layer

Importance of Layers in Software Architecture;

Layered architecture is a design approach that organizes software systems into distinct layers, each with a specific role and responsibility. Layers improve structure, maintainability, and scalability, making them an essential element of modern software design. Here are the main point that explains why layers are important in Software Architecture;

1. Separation of Concerns: Each layer focuses on a specific aspect of the application, such as data management, business logic, or user interface.
2. Maintainability: Changes in one layer do not affect other layers if the architecture is properly designed.
3. Scalability
4. Reusability etc...

13. Software Environments

THEORY EXERCISE: Explain the importance of a development environment in software production.

- **ANS:**

A development environment is a setup of tools, frameworks, and configurations used by developers to write, test, and debug software during its production. It serves as the foundation for building reliable, scalable, and maintainable software applications. Here's why it is crucial in software production:

1. Facilitates Code Development
2. Standardization
3. Supports Testing
4. Reduces Deployment Risks
5. Collaboration and Team Productivity
6. Encourages Innovation and Experimentation
7. Facilitates Debugging and Monitoring
8. Supports Continuous Integration/Continuous Deployment
9. Security

14. Source Code

THEORY EXERCISE: What is the difference between source code and machine code?

- **ANS:**

--> Definition

Source Code : Human-readable code written by programmers in high-level programming languages (e.g., Python, C).

Machine Code : Binary instructions directly executed by a computer’s CPU.

--> Language

Source Code : Written in high-level or assembly languages.

Machine Code : Written in binary (0s and 1s), understood by the computer's processor.

--> Purpose

Source Code : Designed for humans to write, read, and understand software programs.

Machine Code : Designed for machines to execute tasks at the hardware level.

--> Conversion Process

Source Code : Translated into machine code using compilers, interpreters, or assemblers.

Machine Code : Produced as the output of the compilation or assembly process.

--> EXAMPLE

Source Code : print("Hello, World!") (Python)

Machine Code : 10101010 11011011 00001111 (Binary instructions for a CPU).

15. Github and Introductions

THEORY EXERCISE: Why is version control important in software development?

- **ANS:**

Version control is a system that records changes to a file or set of files over time, enabling developers to track, manage, and collaborate on software projects efficiently. It is an essential part of modern software development for several reasons.

Version control is important for keeping track of changes to code, files, and other digital assets.

With version control, contributors can easily identify and access the most recent draft, reducing the risk of mistakenly working on an outdated version. It enables higher velocity of data teams while reducing the cost of errors.

16. Student Account in Github

THEORY EXERCISE: What are the benefits of using Github for students?

- **ANS:**

GitHub is a web-based platform that allows developers to store, manage, and share their code. It's a collaboration tool that helps developers work together on projects.

GitHub can help students learn to code, build projects, and collaborate with others.

Benefits of GitHub for students:

1. Free tools and services
2. Collaboration
3. Version control (Students can save different versions of their projects for future reference.)
4. Showcase projects

17. Types of Software

THEORY EXERCISE: What are the differences between open-source and proprietary software?

- **ANS:**

--> Definition

Open-Source Software : Software whose source code is freely available for modification, redistribution, and use.

Proprietary Software: Software with a closed source code that is owned and controlled by its creator or vendor.

--> Cost

Open-Source Software : free or available at a minimal cost

Proprietary Software: Usually requires purchasing a license or subscription to use.

--> Security

Open-Source Software : Security depends on the community's vigilance in finding and fixing vulnerabilities.

Proprietary Software: Security updates and patches are managed by the vendor, but users must wait for official fixes.

--> Examples

Open-Source Software : Linux, Apache, Mozilla Firefox, LibreOffice, MySQL

Proprietary Software: Microsoft Windows, Adobe Photoshop, Microsoft Office, Oracle Database

18. GIT and GITHUB Training

THEORY EXERCISE: How does GIT improve collaboration in a software development team?

- **ANS:**

Git improves collaboration in software development teams by allowing developers to work on different parts of a project simultaneously without overwriting each other's changes. Git also allows developers to review and discuss changes before merging them into the main branch.

19. Application Software

THEORY EXERCISE: What is the role of application software in businesses?

- **ANS:**

Application Software helps businesses work more efficiently by automating and optimizing processes, and by improving communication and data management. The points given belove are the factors in which application software helps for business:

1. Communication
2. Data management
3. Process management
4. Customer relationship management

A big point as an example is ERP model :

Enterprise resource planning (ERP): Helps businesses manage core business processes like finance, human resources, and supply chain

20. Software Development Process

THEORY EXERCISE: What are the main stages of the software development process?

- **ANS:**

The main stages of the software development process are:

1. Planning: Create a basic plan for the application, based on business requirements
2. Requirements analysis: Gather and analyze the requirements for the application
3. Design: Create the product architecture
4. Coding: Write the code for the application
5. Testing: Test the application for functionality
6. Deployment: Make the application available for use
7. Maintenance: Continue to support and improve the application

21. Software Requirement

THEORY EXERCISE: Why is the requirement analysis phase critical in software development?

- **ANS:**

The requirement analysis phase is critical in software development because it ensures a clear understanding of what the client or user needs, providing a solid foundation for the entire project. By defining goals and expectations early, it prevents misunderstandings,

saves time and money by avoiding costly changes later, and ensures the final product meets user requirements. This phase also serves as a roadmap for the design, coding, and testing stages, improving the overall quality and effectiveness of the software.

22. Software Analysis

THEORY EXERCISE: What is the role of software analysis in the development process?

- **ANS:**

Software analysis plays a key role in the development process by helping the team understand the problem they need to solve and plan how to build the solution. It involves studying user needs, system requirements, and any constraints to create a clear roadmap for the project. This step ensures that the team focuses on the right features, avoids mistakes, and builds software that meets user expectations, saving time, effort, and costs in the long run.

23. System Design

THEORY EXERCISE: What are the key elements of system design?

- **ANS:**

The key elements of system design are:

1. System Architecture: Overall structure and component interaction.
2. Data Design: Organizing and structuring data.
3. User Interface Design: Ensuring user-friendliness.
4. Component Design: Dividing the system into modules.
5. Security Design: Protecting data and operations.
6. Scalability and Performance: Handling growth and maintaining efficiency.
7. Integration Design: Connecting with external systems.
8. Reliability and Fault Tolerance: Ensuring system stability.
9. Technology Stack: Tools and platforms used.

10. Compliance and Standards: Meeting legal and industry requirements.

24. Software Testing

THEORY EXERCISE: : Why is software testing important?

- **ANS:**

Software testing is important because it ensures the software is reliable, functional, and meets user expectations. It helps identify and fix bugs, errors, or security vulnerabilities before the product is released, reducing the risk of failures and costly fixes later. Testing also improves performance, usability, and compatibility across different devices and environments, ensuring a high-quality product. Ultimately, it builds user trust, saves time and money, and ensures the software works as intended.

25. Maintenance

THEORY EXERCISE: : What types of software maintenance are there?

- **ANS:**

There are mainly 4 types of Software maintenance;

1. **Corrective Maintenance:** Fixing bugs, errors, and defects in the software identified after deployment.
2. **Adaptive Maintenance:** Updating software to work with changes in the environment, such as new operating systems, hardware, or regulations.
3. **Perfective Maintenance:** Improving performance, usability, or functionality to enhance the software's efficiency and meet user demands.
4. **Preventive Maintenance:** Anticipating potential issues and making updates to prevent future problems and ensure long-term reliability.

26. Development

THEORY EXERCISE: What are the key differences between web and desktop applications?

- **ANS:**

Web applications are accessed through a browser and do not require installation, making them platform-independent and easy to access from any device with an internet connection. They rely on an active internet connection to function and are updated automatically on the server side, which ensures all users have access to the latest version. On the other hand, desktop applications are installed on specific operating systems and run directly from the device. They can work offline and often offer better performance since they utilize local system resources. However, updates must be manually downloaded and installed, and the software is generally platform-dependent, meaning it may only work on certain operating systems. Security for web apps is critical on the server side, while desktop apps rely on the security of the local device. Overall, web apps offer convenience and accessibility, while desktop apps provide better performance and offline capabilities.

27. Web Application

THEORY EXERCISE: What are the advantages of using web applications over desktop applications?

- **ANS:**

The Points given below are the main advantages of Web applications:

1. Scalability: Web apps can easily handle an increasing number of users by scaling server resources, ensuring reliability as usage grows.
2. Accessibility: Web applications can be accessed from anywhere with an internet connection, making them highly convenient for remote work or collaboration.
3. Platform Independence and No installation Required. Also automatic update available.

4.Cost-Effectiveness: Web apps often have lower upfront costs since they don't require extensive hardware or software installations on use devices.
5. Collaboration Features: Web apps often include real-time collaboration tools, enabling users to work together seamlessly from different locations.

28. Designing

THEORY EXERCISE: What role does UI/UX design play in application development?

- **ANS:**

UI/UX design plays a important role in application development by ensuring the app is visually appealing, user-friendly, and provides a seamless experience.

UI/UX design bridges the gap between user expectations and the application's functionality, ensuring the product is not only functional but also delightful to use.

29. Mobile Application

THEORY EXERCISE: What are the differences between native and hybrid mobile apps?

- **ANS:**

Native apps are developed specifically for a single platform, such as iOS or Android, using platform-specific programming languages like Swift or Kotlin. They offer high performance and a superior user experience because they are optimized for the platform and follow its design guidelines

Native apps also provide full access to device hardware and features like the camera, GPS, and sensors. However, they are more expensive and time-consuming to develop since separate codebases are required for each platform.

H**ybrid apps** are built using web technologies like HTML, CSS, and JavaScript and are wrapped in a native container, enabling them to run on multiple platforms using a single codebase. While they are faster and cheaper to develop, they typically offer slightly lower performance and less refined UI/UX compared to native apps. Hybrid apps have limited access to device hardware, though plugins can help bridge these gaps.

- Native apps are ideal for performance-intensive and platform-specific functionalities, while hybrid apps are better suited for cost-effective solutions requiring cross-platform compatibility.

30. DFD (Data Flow Diagram)

THEORY EXERCISE: What is the significance of DFDs in system analysis?

- **ANS:**

A D**ata Flow Diagram** (DFD) is a significant modeling technique for analyzing and constructing information processes.

DFDs are important in system analysis because they help you visualize how data moves through a system. They can help you identify problems, improve processes, and protect data.

DFD literally means an illustration that explains the course or movement of information in a process.

DFD illustrates this flow of information in a process based on the inputs and outputs.

31`.Desktop Application

THEORY EXERCISE: What are the pros and cons of desktop applications compared to webapplications?

- **ANS:**

ON THE NEXT PAGE>>>>

--> **Pros of Desktop Applications:**

1. Offline Access: Work without an internet connection.
2. Performance: Faster and more resource-efficient as they utilize local hardware.
3. Customization: Deeper integration with the operating system allows more advanced features.

--> Pros of Web Applications:

1. Accessibility: Access from any device with an internet connection.
2. Platform Independence: Works across multiple operating systems via browsers.
3. Automatic Updates: Always run the latest version without manual updates.

--> Cons of Desktop Applications:

1. Platform Dependency: Need separate versions for different operating systems.
2. Installation: Requires downloading and installing updates manually.
3. Limited Accessibility: Can only be used on the device where installed.

--> Cons of Web Applications:

1. Internet Dependency: Requires a stable internet connection to function.
2. Performance: Slower than desktop apps for resource-intensive tasks.
3. Limited Features: Restricted access to hardware and system-level functionalities.

32. Flow Chart

THEORY EXERCISE: How do flowcharts help in programming and system design?

- **ANS:**
  ON THE NEXT PAGE>>>>

How Flowcharts Help in Programming and System Design:

1. Simplify Processes: Break down complex logic into simple, visual steps.
2. Enhance Communication: Provide a universal visual language for better collaboration.
3. Detect Errors: Help identify issues or inefficiencies in the system.
4. Aid Planning: Serve as a blueprint for algorithms and workflows.
5. Support Documentation: Act as a reference for future maintenance.
6. Facilitate Problem-Solving: Visualize paths and outcomes for debugging and optimization.