Assignment: Software Design, Databases, and Networking

Justin Mathew

`r Sys.Date()`

Module 1: Introduction to Computer Science

1. History and Evolution of Computers

1.1 The Early Beginnings

Computing has a rich history that dates back to ancient times. Early computing devices like the Abacus and mechanical calculators were the first attempts to aid in manual computation. The invention of the Turing Machine by Alan Turing in 1936 was a significant milestone, laying the theoretical foundation for modern computers.

1.2 Generations of Computers

- **First Generation (1940-1956) - Vacuum Tubes**: The first electronic computers used vacuum tubes and were enormous, consuming vast amounts of power. ENIAC and UNIVAC are notable examples.
- **Second Generation (1956-1963) - Transistors**: Transistors replaced vacuum tubes, making computers smaller, faster, and more energy-efficient.
- **Third Generation (1964-1971) - Integrated Circuits**: The development of integrated circuits further miniaturized computers, leading to the first commercially available computers.
- **Fourth Generation (1971-Present) - Microprocessors**: The invention of the microprocessor brought about personal computers (PCs). This era saw the rise of companies like IBM, Apple, and Microsoft.
- **Fifth Generation (Present and Beyond) - Artificial Intelligence**: Modern computers are characterized by advancements in AI, machine learning, and quantum computing.

1.3 Milestones in Computing

- **Development of the Internet**: Originally a military project, the Internet has transformed communication, commerce, and entertainment.
- **The Rise of Personal Computing**: The 1980s saw the mass adoption of personal computers in homes and offices.
- **Cloud Computing and Big Data**: The ability to store and process large datasets over the Internet has revolutionized industries.
- **Advances in Artificial Intelligence and Machine Learning**: AI is now integral to various applications, from autonomous vehicles to personalized medicine.

2. Overview of Computer Systems

2.1 Components of a Computer System

- **Hardware**: This includes physical components like the CPU, memory, storage devices, and input/output peripherals.
- **Software**: Software can be categorized into system software (like operating systems) and application software (like word processors).
- **Data**: Data is the information processed by computers, and it plays a central role in computing.

2.2 Types of Computer Systems

- **Personal Computers**: These are general-purpose computers designed for individual use.
- **Servers and Mainframes**: Servers manage network resources, while mainframes are used for large-scale computing tasks.
- **Embedded Systems**: Found in devices like cars and home appliances, these systems are designed for specific tasks.
- **Supercomputers**: The fastest computers, used for complex calculations in fields like weather forecasting and scientific simulations.

3. Basic Terminology and Concepts

3.1 Understanding Computer Hardware

- **Central Processing Unit (CPU)**: The CPU is the brain of the computer, responsible for executing instructions.
- **Memory (RAM, ROM)**: RAM is temporary memory used during processing, while ROM is permanent storage for critical system instructions.

- **Storage Devices (Hard Drives, SSDs)**: These devices store data permanently. SSDs are faster and more durable than traditional hard drives.
- **Input and Output Devices**: Keyboards, mice, and monitors are examples of devices that allow users to interact with the computer.

3.2 Software Fundamentals

- **System Software**: This includes operating systems like Windows, macOS, and Linux, which manage hardware and software resources.
- **Application Software**: Programs like Microsoft Office, web browsers, and graphic design tools are designed for specific tasks.

3.3 Programming Languages and Development

- **Low-Level vs. High-Level Languages**: Low-level languages (like Assembly) are closer to machine code, while high-level languages (like Python, Java) are more abstract and easier to learn.
- **Overview of Popular Programming Languages**: Python is known for its simplicity, Java for cross-platform capabilities, and C++ for performance-critical applications.
- **Software Development Life Cycle (SDLC)**: The SDLC includes phases like planning, designing, coding, testing, deployment, and maintenance.

3.4 Execution and Testing

- **Compilation vs. Interpretation**: Compiled languages (like C++) translate code into machine language before execution, while interpreted languages (like Python) execute code line by line.
- **Debugging and Error Handling**: Debugging is the process of identifying and fixing errors in code. Proper error handling ensures that a program can gracefully manage unexpected situations.
- **Testing**: Testing ensures the quality and functionality of software. It includes:
  - **Unit Testing**: Testing individual components.
  - **Integration Testing**: Testing the interaction between components.
  - **System Testing**: Testing the entire system for defects.

4. Applications of Computer Science and Engineering

4.1 Software Development

- **Web Development**: Involves creating websites and web applications using technologies like HTML, CSS, JavaScript, and backend languages like Python and PHP.
- **Mobile Application Development**: Focuses on building applications for mobile platforms like iOS and Android using languages like Swift and Kotlin.
- **Desktop Software**: Includes developing traditional software applications for operating systems like Windows, macOS, and Linux.

4.2 Data Science and Analytics

- **Big Data Processing**: Involves handling and analyzing large datasets using tools like Hadoop and Spark.
- **Machine Learning Algorithms**: Algorithms like decision trees, neural networks, and clustering are used for tasks like predictive analytics and pattern recognition.
- **Data Visualization**: Tools like Tableau and Matplotlib help in visualizing data to derive insights.

4.3 Network and Security

- **Cybersecurity Principles**: Protecting computer systems and networks from attacks. This includes encryption, firewalls, and secure coding practices.
- **Networking Protocols**: Protocols like TCP/IP, HTTP, and FTP define the rules for data transmission across networks.
- **Cloud Computing**: Offers on-demand access to computing resources over the Internet. Popular cloud platforms include AWS, Google Cloud, and Microsoft Azure.

4.4 Emerging Technologies

- **Internet of Things (IoT)**: IoT involves connecting everyday objects to the Internet, enabling them to send and receive data.
- **Artificial Intelligence and Robotics**: AI powers intelligent systems, while robotics focuses on designing and building robots.
- **Quantum Computing**: A cutting-edge technology that leverages quantum mechanics to perform computations far faster than classical computers.

4.5 Real-World Applications

- **Healthcare**: Computers are used in medical imaging, electronic health records, and telemedicine.
- **Finance**: Algorithmic trading, blockchain technology, and digital payments have transformed the financial sector.
- **Entertainment**: Video games, virtual reality (VR), and streaming services rely heavily on computer science for their operation and development.

Summary and Conclusion

- **Recap of Key Concepts**: A summary of the major topics covered, emphasizing the role of computer science in driving innovation.
- **The Role of Computer Science in Modern Society**: A discussion on how computer science impacts various industries and day-to-day life.
- **Future Trends and Career Opportunities**: Insights into emerging fields like AI, cybersecurity, and data science, and the career paths available in computer science.

Module 2: Hardware and PC Assembly

1. Architecture of a Computer

1.1 Basic Computer Architecture

- **Von Neumann Architecture**: An overview of the traditional architecture, including the CPU, memory, and input/output subsystems.
- **Motherboard**: The central hub that connects all the components of a computer. Discussion on buses, chipsets, and form factors (ATX, Micro-ATX, etc.).
- **Power Supply Unit (PSU)**: The role of the PSU in converting and distributing power to all components.

1.2 Processor (CPU)

- **Central Processing Unit (CPU)**: The brain of the computer, responsible for executing instructions. Overview of CPU components: Control Unit (CU), Arithmetic Logic Unit (ALU), and Registers.
- **Multi-core Processors**: Explanation of dual-core, quad-core, and other multi-core processors, including hyper-threading.
- **CPU Cooling**: Importance of cooling, different types of cooling methods (air, liquid), and thermal management.

1.3 Memory

- **Primary Memory**: Explanation of RAM (Random Access Memory) and its types (DDR, DDR2, DDR3, DDR4, DDR5). Overview of ROM (Read-Only Memory).
- **Secondary Storage**: Different types of storage devices, including Hard Disk Drives (HDD), Solid-State Drives (SSD), and Optical Drives.
- **Cache Memory**: The role of cache in speeding up CPU operations, including different levels of cache (L1, L2, L3).

1.4 I/O Devices

- **Input Devices**: Explanation of keyboards, mice, scanners, and other input peripherals.
- **Output Devices**: Overview of monitors, printers, speakers, and other output devices.
- **Peripheral Connectivity**: Different types of ports and connectors (USB, HDMI, DisplayPort, etc.), and their uses.

1.5 Networking Devices

- **Network Interface Card (NIC)**: The role of the NIC in enabling network connectivity.
- **Routers and Switches**: Explanation of how routers and switches manage network traffic.
- **Modems and Access Points**: The function of modems in converting signals, and the role of access points in wireless networking.
- **Cabling**: Overview of Ethernet cables (Cat5e, Cat6), fiber optics, and wireless technologies.

2. Assembling a PC

2.1 Pre-assembly Considerations

- **Component Selection**: Criteria for choosing compatible components (motherboard, CPU, GPU, RAM, storage).
- **Workspace Setup**: Preparing a clean, static-free environment with the necessary tools.

2.2 Step-by-Step Assembly Guide

- **Installing the CPU**: Proper handling and installation of the CPU onto the motherboard.
- **Installing RAM**: Inserting memory modules into the motherboard.
- **Mounting the Motherboard**: Securing the motherboard inside the case.

- **Installing Storage Devices**: Connecting HDDs, SSDs, and optical drives.
- **Connecting Power Supply**: Attaching power connectors to the motherboard, CPU, GPU, and storage devices.
- **Connecting I/O Devices**: Hooking up peripherals to the appropriate ports.
- **Initial Power-Up and BIOS Setup**: Powering on the system and configuring BIOS settings.

2.3 Troubleshooting Common Issues

- **No POST (Power-On Self-Test)**: Diagnosing issues when the system fails to start.
- **Boot Errors**: Resolving common boot-related problems.
- **Hardware Compatibility Issues**: Identifying and resolving conflicts between components.

3. Additional Hardware Devices

3.1 Graphics Processing Unit (GPU)

- **Integrated vs. Dedicated GPUs**: Differences and use cases for integrated and dedicated graphics.
- **GPU Architecture**: Explanation of cores, clock speed, VRAM, and cooling solutions.
- **Gaming and Professional GPUs**: Differences between GPUs designed for gaming and those for professional tasks like 3D rendering.

3.2 Sound Cards and Audio Interfaces

- **Sound Cards**: The role of sound cards in enhancing audio quality.
- **External Audio Interfaces**: Devices used for professional audio recording and production.

3.3 Additional Storage Options

- **External Hard Drives and SSDs**: Use cases and benefits of external storage.
- **Network Attached Storage (NAS)**: Explanation of NAS devices for centralized data storage and access.

4. Internet of Things (IoT)

4.1 Introduction to IoT

- **Definition and Overview**: Understanding IoT and its significance in connecting devices.
- **IoT Devices**: Examples of IoT devices in smart homes, healthcare, and industrial applications.
- **Communication Protocols**: Overview of protocols like MQTT, Zigbee, and Bluetooth.

4.2 IoT Architecture

- **Sensors and Actuators**: Role of sensors in collecting data and actuators in performing actions.
- **Edge Computing**: Processing data at the edge of the network to reduce latency.
- **Cloud Integration**: How IoT devices connect to cloud platforms for data storage and analytics.

5. Augmented Reality (AR) and Virtual Reality (VR)

5.1 Understanding AR and VR

- **Definitions and Differences**: Distinguishing between AR (overlaying digital content on the real world) and VR (immersive digital environments).
- **Applications of AR/VR**: Use cases in gaming, education, healthcare, and training simulations.

5.2 AR/VR Hardware

- **Head-Mounted Displays (HMDs)**: Devices like Oculus Rift, HTC Vive, and Microsoft HoloLens.
- **Controllers and Sensors**: Devices that track user movements and interactions in AR/VR environments.

5.3 Developing AR/VR Content

- **Software Tools**: Introduction to tools like Unity, Unreal Engine, and ARKit for creating AR/VR experiences.
- **Challenges in AR/VR Development**: Discussing issues like motion sickness, hardware limitations, and content creation.

Summary and Conclusion

- **Recap of Key Concepts**: A summary of the major topics covered, emphasizing the importance of understanding hardware components and PC assembly.
- **Future Trends**: A look at emerging technologies in hardware, IoT, and AR/VR, and their potential impact on the industry.
- **Practical Applications**: Encouraging hands-on practice in PC assembly and experimentation with IoT and AR/VR devices.

Module 3: Software

1. Types of Software

1.1 Application Software

- **Definition**: Software designed to help users perform specific tasks or functions, such as word processing, spreadsheet management, or browsing the internet.
- **Examples**: Microsoft Office Suite (Word, Excel, PowerPoint), web browsers like Chrome and Firefox, media players, and graphic design tools.

1.2 Proprietary Software

- **Definition**: Software that is owned by an individual or a company and is distributed under a licensing agreement that restricts use, modification, and distribution.
- **Examples**: Microsoft Windows, Adobe Photoshop, and Apple macOS.
- **Advantages**: Often comes with dedicated support, frequent updates, and a high level of integration with hardware.
- **Disadvantages**: Generally requires purchase or subscription, and users have limited control over the software.

1.3 Open Source Software

- **Definition**: Software with source code that is made available to the public, allowing anyone to view, modify, and distribute it.
- **Examples**: Linux operating systems (Ubuntu, Fedora), Apache Web Server, and GIMP (a graphic editor).
- **Advantages**: Free to use, modify, and distribute. Encourages community collaboration and innovation.
- **Disadvantages**: May lack official support, and some open-source projects may have less frequent updates or be less user-friendly.

2. System Software

2.1 Definition

- **System Software**: Software designed to manage the system's hardware and provide a platform for running application software. It acts as an intermediary between the user and the computer hardware.

2.2 Operating Systems

- **Definition**: The software that manages all of the hardware and other software on a computer. It provides essential functions like file management, process management, and hardware management.
- **Examples**: Microsoft Windows, Linux distributions, and macOS.
- **Functions**:
  - **Process Management**: Manages the execution of processes, including multitasking and multiprocessing.
  - **Memory Management**: Allocates memory to processes and ensures optimal utilization of the system's RAM.
  - **File System Management**: Manages files on storage devices, ensuring data is stored and retrieved efficiently.
  - **Device Management**: Manages communication between the system's hardware and the software applications.

2.3 Translation Software

- **Assemblers**: Convert assembly language code into machine code.
- **Compilers**: Translate high-level programming languages (like C++, Java) into machine code, producing an executable program.
- **Interpreters**: Translate high-level code into machine code line by line, executing each instruction immediately.
- **Difference Between Compilers and Interpreters**: Compilers translate the entire program at once, while interpreters translate it line by line.

2.4 Linker

- **Definition**: A program that takes one or more object files generated by a compiler and combines them into a single executable file. It resolves references between the object files, connecting function calls with their definitions.

- **Function**: The linker links various object files, ensuring that the program is complete and can be executed by the operating system.

2.5 Loader

- **Definition**: A program that loads the executable file into memory, preparing it for execution. It is responsible for placing the code and data in memory, setting up any required data structures, and starting the program's execution.
- **Function**: The loader reads the executable file's instructions and data into the appropriate memory locations, adjusting addresses as necessary, and then transfers control to the starting point of the program.

3. BIOS and POST

3.1 BIOS (Basic Input/Output System)

- **Definition**: Firmware used to perform hardware initialization during the booting process and to provide runtime services for operating systems and programs. It is the first software that runs when a computer is powered on.
- **Functions**:
  - **Power-On Self Test (POST)**: Checks the system's hardware to ensure everything is functioning correctly before loading the operating system.
  - **Bootstrap Loader**: Locates the operating system and passes control to it, allowing the OS to take over the boot process.
  - **BIOS Setup Utility**: Allows users to configure hardware settings, such as system clock, boot sequence, and hardware settings.

3.2 POST (Power-On Self Test)

- **Definition**: A diagnostic testing sequence run by the BIOS to check the integrity and functionality of the computer's hardware components, such as the CPU, RAM, and storage devices, before loading the operating system.
- **Function**:
  - **Hardware Testing**: Verifies that all essential hardware components are present and functioning correctly.
  - **Error Reporting**: If any issues are detected, the POST process will typically produce a series of beeps or display an error message indicating the faulty component.
  - **Transition to Bootloader**: Upon successful completion of POST, the BIOS hands over control to the bootloader to continue the booting process.

Module 4: Databases and Networks

1. Types of Data

1.1 Structured Data

- **Definition**: Data that is organized into a defined structure, usually in rows and columns, making it easily searchable by algorithms.
- **Examples**: Relational databases like SQL databases (MySQL, PostgreSQL), spreadsheets.
- **Characteristics**:
  - Organized into tables with predefined schema.
  - Data types are defined (e.g., integer, string).
  - Easily queryable using SQL (Structured Query Language).

1.2 Semi-structured Data

- **Definition**: Data that does not reside in a relational database but still has some organizational properties that make it easier to analyze.
- **Examples**: JSON files, XML files, NoSQL databases like MongoDB.
- **Characteristics**:
  - Contains tags or markers to separate data elements.
  - More flexible than structured data but less rigid.
  - Can be partially queried using specific query languages (e.g., XPath for XML, JSONPath for JSON).

1.3 Unstructured Data

- **Definition**: Data that has no predefined format or organization, making it more challenging to process and analyze.
- **Examples**: Text documents, images, videos, social media posts.
- **Characteristics**:
  - No specific structure or schema.
  - Requires more complex processing techniques, such as natural language processing (NLP) for text data.
  - Stored in data lakes or NoSQL databases.

2. Database Management Systems (DBMS)

2.1 Definition

- **DBMS**: Software that uses a standard method to store and organize data, providing mechanisms for retrieval, management, and manipulation of data.
- **Functions**:
  - **Data Storage**: Manages data storage efficiently, ensuring data consistency and integrity.
  - **Query Processing**: Provides tools to retrieve data using query languages like SQL.
  - **Transaction Management**: Ensures that multiple operations on the database are completed successfully (ACID properties).
  - **Security Management**: Controls access to the data, ensuring that only authorized users can perform certain operations.

2.2 Types of DBMS

- **Relational DBMS (RDBMS)**: Stores data in tables with relationships between them. Examples include MySQL, PostgreSQL, and Oracle.
- **NoSQL DBMS**: Designed for unstructured or semi-structured data, offering flexibility in how data is stored and accessed. Examples include MongoDB, Cassandra, and Redis.
- **In-Memory DBMS**: Stores data in the main memory rather than on disk to provide faster access. Examples include Redis and Memcached.
- **Distributed DBMS**: Manages data across multiple databases in different locations, providing a single view of the data. Examples include Google Spanner and Amazon Aurora.

2.3 Use Cases

- **Enterprise Data Management**: RDBMSs are commonly used in businesses to manage customer information, financial data, and inventory.
- **Big Data Analytics**: NoSQL databases are employed in big data environments where large volumes of unstructured data need to be processed, such as in social media analytics.
- **Real-time Applications**: In-memory databases are used in applications requiring low-latency data access, such as online gaming and financial trading platforms.
- **Cloud-based Applications**: Distributed DBMSs are utilized in cloud environments to ensure data availability and redundancy across different regions.

3. Networking Basics

3.1 Definition

- **Networking**: The practice of connecting computers and other devices together to share resources, communicate, and collaborate. It forms the backbone of modern communication systems, enabling the exchange of data over local and wide areas.

3.2 Types of Networks

- **Local Area Network (LAN)**: A network that connects computers within a limited area, such as a home, school, or office building.
- **Wide Area Network (WAN)**: A network that covers a broad area (e.g., the internet) and connects LANs together.
- **Wireless Networks**: Networks that use wireless data connections, such as Wi-Fi, to connect devices without physical cables.

3.3 Networking Devices

- **Router**: A device that forwards data packets between computer networks, directing traffic efficiently.
- **Switch**: A device that connects devices within a LAN and uses MAC addresses to forward data to the correct destination.
- **Modem**: A device that modulates and demodulates signals for data transmission over phone lines, cable systems, or satellite connections.

4. Networking Topologies and Protocols

4.1 Networking Topologies

- **Bus Topology**: A single central cable (the bus) to which all network devices are connected. Data sent by one device is available to all devices, but only the intended recipient processes it.
- **Star Topology**: All devices are connected to a central hub or switch. The hub sends the data to the correct device. This topology is commonly used in home networks.
- **Ring Topology**: Devices are connected in a circular fashion, where each device is connected to two other devices, forming a ring. Data travels in one direction, passing through each device until it reaches its destination.
- **Mesh Topology**: Every device is connected to every other device in the network. This topology offers high redundancy and reliability but is complex and expensive to implement.

4.2 Networking Protocols

- **TCP/IP (Transmission Control Protocol/Internet Protocol)**: The fundamental protocol suite for the internet, enabling reliable communication between devices.
  - **TCP**: Ensures data is sent and received in the correct order and without errors.
  - **IP**: Handles the addressing and routing of packets across the network.
- **HTTP/HTTPS (Hypertext Transfer Protocol/Secure)**: The protocol used by web browsers to communicate with web servers. HTTPS is the secure version, encrypting data for secure communication.
- **FTP (File Transfer Protocol)**: A protocol for transferring files between computers on a network.
- **SMTP (Simple Mail Transfer Protocol)**: Used for sending emails across networks.
- **DNS (Domain Name System)**: Translates human-readable domain names (like www.example.com) into IP addresses that computers use to identify each other on the network.
- **DHCP (Dynamic Host Configuration Protocol)**: Automatically assigns IP addresses to devices on a network, simplifying network management.

Module 5: Design of a Software System

1. Design Principles

1.1 Consistency

- **Definition**: Ensuring that design elements are uniform across the application, providing users with a predictable and intuitive experience.
- **Importance**:
  - **User Expectations**: Consistent design helps users build familiarity and understanding of how to interact with the system.
  - **Usability**: Reduces the learning curve and potential errors by maintaining uniformity in design patterns and interactions.

1.2 Visibility

- **Definition**: Making important elements and functions visible and easily accessible to users.
- **Importance**:
  - **Discoverability**: Users should be able to find features and actions without extensive searching.

  - **Clarity**: Ensures that users are aware of their current options and status within the application.

1.3 Feedback

- **Definition**: Providing users with clear and immediate responses to their actions within the system.
- **Importance**:
  - **User Assurance**: Feedback confirms that actions have been successfully completed or if there are errors.
  - **Interaction Confirmation**: Helps users understand the effects of their actions and provides guidance on how to proceed.

1.4 Affordance

- **Definition**: Design elements should suggest their functionality through their appearance.
- **Importance**:
  - **Intuitive Interaction**: Buttons, icons, and other elements should visually indicate their purpose (e.g., buttons should look clickable).
  - **Ease of Use**: Users can easily understand how to interact with elements based on their design.

2. Color and Typography

2.1 Color

- **Role in Design**:
  - **Visual Appeal**: Enhances the aesthetic appeal and overall user experience.
  - **Emotional Impact**: Colors can evoke emotions and set the tone for the application.
  - **Functionality**: Used to highlight important information, create visual hierarchy, and guide user actions.
- **Best Practices**:
  - **Contrast**: Ensure sufficient contrast between text and background for readability.
  - **Color Consistency**: Use a consistent color scheme throughout the application to reinforce brand identity and improve usability.
  - **Accessibility**: Consider color blindness and ensure that color choices are accessible to all users.

2.2 Typography

- **Role in Design**:
  - **Readability**: Affects how easily users can read and understand text.
  - **Hierarchy**: Helps establish a visual hierarchy, making it easier to navigate content.
  - **Brand Identity**: Contributes to the overall brand and aesthetic of the application.
- **Best Practices**:
  - **Font Choices**: Select fonts that are legible and align with the brand's style.
  - **Font Sizes**: Use different sizes to differentiate headings, subheadings, and body text.
  - **Line Spacing**: Ensure appropriate spacing between lines of text to improve readability.

3. User-Centric Design

3.1 Definition

- **User-Centric Design**: A design approach that prioritizes the needs, preferences, and behaviors of users throughout the design process.
- **Importance**:
  - **User Satisfaction**: Ensures that the design meets user needs and expectations.
  - **Usability**: Creates a more intuitive and effective user experience by focusing on how users interact with the system.

3.2 Layout and Interactive Prototypes

- **Layout**: The arrangement of visual elements on a page or screen, including content, controls, and navigation.
  - **Principles**:
    - **Alignment**: Align elements to create a clean and organized appearance.
    - **Hierarchy**: Arrange elements to emphasize important information and guide user flow.
    - **Spacing**: Use spacing to separate elements and improve readability.
- **Interactive Prototypes**: Mockups that simulate the interaction with the design, allowing users to experience and test the design before development.
  - **Benefits**:
    - **User Feedback**: Collect feedback on design usability and functionality.

    - **Iteration**: Refine and improve the design based on user interactions and feedback.
    - **Communication**: Provide a tangible representation of the design to stakeholders and developers.

4. Overview of UI Design Tools

4.1 Figma

- **Description**: A cloud-based UI design tool that enables real-time collaboration and prototyping.
- **Features**:
    - **Collaborative Design**: Multiple users can work on the same design simultaneously.
    - **Prototyping**: Create interactive prototypes with transitions and animations.
    - **Design Systems**: Support for creating and maintaining design systems and reusable components.

4.2 Sketch

- **Description**: A vector-based design tool for creating user interfaces and user experiences.
- **Features**:
    - **Symbol Management**: Use symbols for reusable design elements and components.
    - **Artboards**: Design multiple screens and layouts within a single document.
    - **Plugins**: Extensive plugin ecosystem for extending functionality and integrating with other tools.

4.3 Adobe XD

- **Description**: A UI/UX design tool from Adobe that provides capabilities for designing, prototyping, and sharing interactive experiences.
- **Features**:
    - **Design and Prototyping**: Create high-fidelity designs and interactive prototypes.
    - **Collaboration**: Share designs and prototypes with stakeholders for feedback.
    - **Integration**: Seamless integration with other Adobe Creative Cloud tools for a cohesive design workflow.

Assignment 1: Exploring Computer Systems and Hardware

Objective:

To understand the fundamental concepts of computer science and the basics of computer hardware, including the architecture of computers, hardware components, and their interactions.

Instructions:

1. **Research and Answer Questions**: Provide detailed answers to the following questions based on your understanding of Module 1 and Module 2. Use your own words and ensure clarity in your explanations.
2. **Practical Exercise**: Assemble a hypothetical PC build using the provided hardware components. Justify your choices based on your knowledge of hardware components and their roles.
3. **Submit**: Compile your answers and practical exercise into a single document and submit it by the given deadline.

Part 1: Research and Questions

1. History and Evolution of Computers

- Explain the key milestones in the evolution of computers from early mechanical devices to modern digital computers.
- Discuss the impact of these advancements on computer technology and society.

2. Overview of Computer Systems

- Describe the basic components of a computer system and their functions (e.g., CPU, memory, storage, I/O devices).
- Explain how these components interact to perform computing tasks.

3. Types of Data

- Define and provide examples of structured, semi-structured, and unstructured data.
- Explain the importance of data classification for managing and analyzing information.

4. Database Management Systems (DBMS)

- Differentiate between relational and NoSQL databases.
- Provide examples of use cases where each type of DBMS would be appropriate.

5. Networking Basics

- Define key networking concepts such as LAN, WAN, and networking devices (router, switch, modem).
- Explain the purpose of networking topologies and protocols.

Part 2: Practical Exercise

PC Assembly Scenario

You are tasked with assembling a PC for a new office setup. Consider the following components for your build:

- **Processor (CPU)**
- **Memory (RAM)**
- **Storage (SSD/HDD)**
- **Motherboard**
- **Graphics Card (GPU)**
- **Power Supply Unit (PSU)**
- **Case**
- **Cooling System**
- **Networking Device (e.g., Network Interface Card)**

Instructions:

1. **Select Components**: Choose specific components for each category listed above. You may use real-world brands and models or hypothetical ones.
2. **Justify Your Choices**: For each component, provide a brief justification for your selection. Consider factors such as performance, compatibility, and cost.
3. **Explain the Assembly**: Describe how these components are assembled to create a functioning computer. Include details on how they connect and interact with each other.
4. **Illustrate the Build**: Create a simple diagram or schematic that illustrates the layout of your PC components inside the case.

Submission Guidelines:

- **Document Format**: Submit your answers and practical exercise in a PDF or Word document.
- **Length**: Aim for a total of 4-6 pages, including diagrams.
- **Deadline**: October 16, 2024

Assignment: Software Design, Databases, and Networking

Objective:

To understand fundamental concepts in software design, databases, and networking. This assignment will test your knowledge of software types, database management systems, and networking principles, and apply design principles to a software system.

Instructions:

1. **Research and Answer Questions**: Provide detailed answers to the following questions based on your understanding of Module 3, Module 4, and Module 5. Use your own words and ensure clarity in your explanations.
2. **Practical Exercise**: Apply your knowledge to a practical scenario involving database design, networking, and software design principles.
3. **Submit**: Compile your answers and practical exercise into a single document and submit it by the given deadline.

Part 1: Research and Questions

1. Types of Software

- **Question**: Differentiate between application software and system software. Provide examples of each and discuss their respective roles in a computer system.

2. Database Management Systems (DBMS)

- **Question**: Explain the key differences between relational and NoSQL databases. Describe a scenario where each type would be the most suitable choice.

3. Networking Basics

- **Question**: Define and describe the purpose of common networking protocols such as HTTP, TCP/IP, and DNS. How do these protocols contribute to network communication?

Part 2: Practical Exercise

Software System Design

You are tasked with designing a software system for a small business that needs to manage customer orders and inventory. Your design should address the following aspects:

1. **Database Design**:
    - **Schema**: Design a simple database schema for managing customer orders and inventory. Include tables for customers, orders, and products, and define the relationships between them.
    - **Use Case**: Explain how the database schema supports common queries and operations, such as adding new orders, updating inventory, and retrieving customer information.
2. **Networking Considerations**:
    - **Network Architecture**: Describe the network architecture for the business, including how the system will be accessed by employees and any external stakeholders (e.g., customers placing orders online).
    - **Security**: Outline basic security measures to protect the data and ensure secure communication between the client and server.
3. **Software Design Principles**:
    - **Design Principles**: Apply design principles such as consistency, visibility, feedback, and affordance to the user interface of the system. Explain how each principle is incorporated into the design.
    - **User Interface**: Describe the layout and interactive elements of the user interface, including any wireframes or mockups if applicable.

Submission Guidelines:

- **Document Format**: Submit your answers and practical exercise in a PDF or Word document.
- **Length**: Aim for a total of 5-7 pages, including diagrams and mockups.
- **Deadline**: [Insert Deadline Here]