MODULE 1

SE-Overview of IT Industry

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

* Program is a set of instructions written in a programming language that tells computer what to do.
* How a program functions:

1. Written by a programmer.
2. Complied or interpreted.
3. Executed by the CPU.
4. Input/Output.
5. What are the main differences between high-level and low-level programming languages?

|  |  |  |
| --- | --- | --- |
| Sr. No | High-Level | Low-Level |
| 1 | Closer to human language: Easier to read, write and understand | Closer to machine language: Harder to read but more control over hardware. |
| 2 | Abstraction: They hide complex hardware | Minimal Abstraction: work directly with memory and CPU instructions. |
| 3 | Portable: Can run on different types of hardware with minimal charges. | Not portable: Usually written for specific hardware or processors. |
| 4 | Slower: Generally Slower | Faster and more efficient: because they interact directly with hardware |
| 5 | Example: Python | Example: Assembly |

1. Describe the roles of the client and server in web communication?

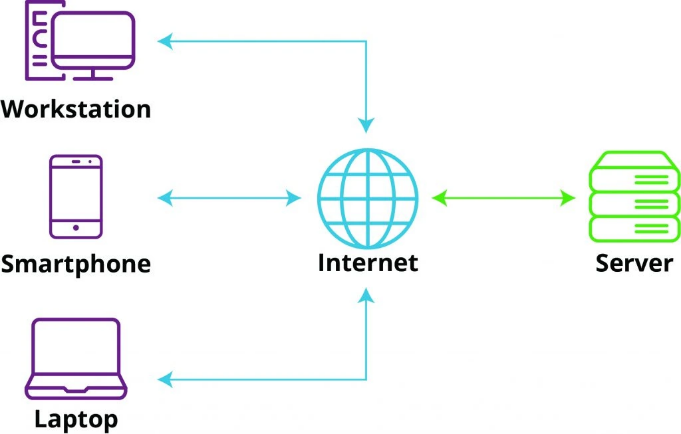
* Roles of client and server in web communication:
* Client:

1. Usually, a web browser or mobile app.
2. Initiates requests to the server.
3. Receives and displays data from the server.

* Server:

1. A computer/system that waits for client requests.
2. Processes the request, fetches or generates data and sends a response.
3. Can host websites, APIs, database, etc.

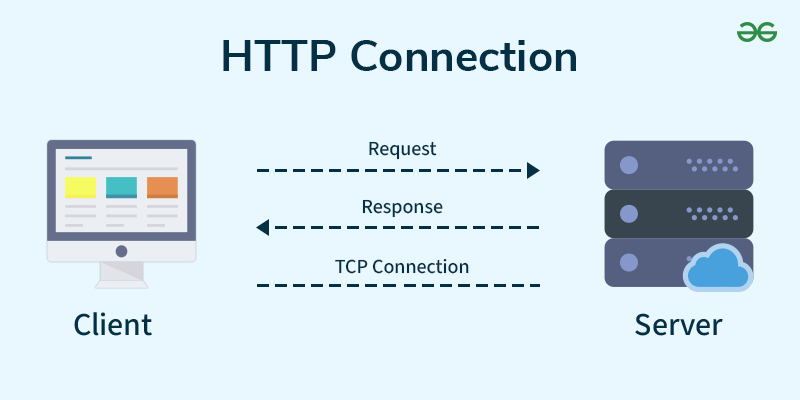
* **LAB EXERCISE:** Research and create a diagram of how data is transmitted from a client to a server over the internet.



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

|  |  |  |
| --- | --- | --- |
| Layer | Client Role | Server Role |
| Application | Send HTTP request. | Receives HTTP request, send response. |
| Transport | Breaks data into packets. | Reassembles packets, ensures reliable delivery. |
| Internet | Adds IP addresses to route data. | Uses IP identify source and destination. |
| Network Access | Sends bits over physical medium. | Receives bits, passes up the layers. |

* **Lab Exercise**: Design a simple HTTP client-server communication in any language.



1. Explain Client Server Communication?

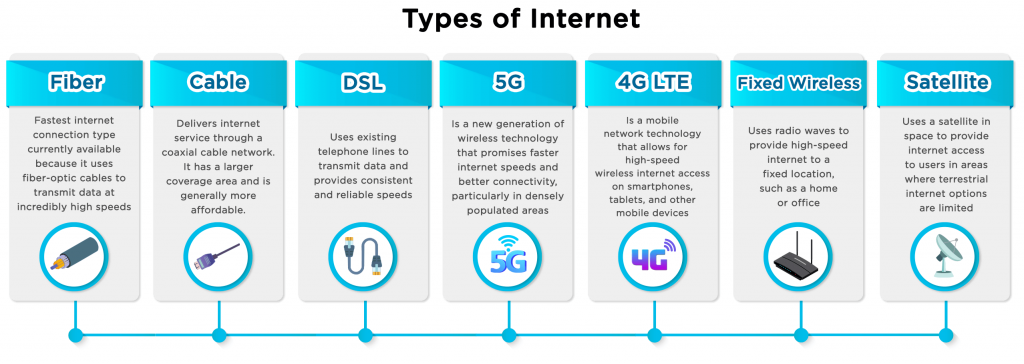
* Client Program:

1. Running on end host.
2. Request services.
3. Example: Web Browser.

* Server Program:

1. Running on end host.
2. Requests Services.
3. Example: Web Browser.
4. How does broadband differ from fibre-optic internet?

|  |  |  |
| --- | --- | --- |
| Feature | Broadband (General) | Fiber-Optic Internet |
| Definition | High-speed Internet. | Specific type of broadband using fibre. |
| Speed | Varies (depends on type: DSL, Cable). | Very fast. |
| Technology | Copper wires, coaxial cables, etc. | Light signals via glass fibres. |
| Reliability | Can suffer from interference. | Highly reliable, weather-resistant. |
| Availability | Widely | Still expanding in some areas. |

* **Lab Exercise:** Research different types of internet connections (e.g., broadband, fibre, satellite)and list their pros and cons.

1. What are the differences between HTTP and HTTPS protocols?

|  |  |  |
| --- | --- | --- |
| Feature | HTTP | HTTPS |
| Full Form | Hyper Text Transfer Protocol. | Hyper Text Transfer Protocol Secure. |
| Security | Not Secure. | Secure. |
| Data Transfer | Plain Text (can be intercepted). | Encrypted (protected from hackers). |
| Port used | Port 80. | Port 443. |
| URL Prefix | http: // | https: // |
| Trust | Not trusted for sensitive data. | Trusted for login, payment, personal info. |
| Certificate | Dose not use SSL certificate. | Uses SSL/TSL certificate (HTTPS padlock) |

* **LAB EXERCISE**: Simulate HTTP and FTP requests using command line tools (e.g., curl).

HTTP Requests with curl

* GET request

curl <https://jsonplaceholder.typicode.com/posts/1>

* POST request with data

curl -X POST [https://jsonplaceholder.typicode.com/posts \](https://jsonplaceholder.typicode.com/posts%20\)

-H "Content-Type: application/json" \

-d '{"title":"foo","body":"bar","userId":1}'

* Adding headers

curl -H "Authorization: Bearer your\_token" <https://api.example.com/data>

FTP Requests with curl

* Download a file from an FTP server

curl -u username:password ftp://ftp.example.com/path/to/file.txt -O

* Upload a file to an FTP server

curl -T localfile.txt -u username:password <ftp://ftp.example.com/path/to/>

1. What is the role of encryption in securing applications?

* Encryption plays a critical role in securing applications by protecting data from unauthorized access. It ensures that even if someone intercepts the data, they cannot read or use it without the encryption key.

1. Data Protection.
2. Secure Communication.
3. Authentication.
4. Data Integrity.
5. Compliance.

* **LAB EXERCISE**: Identify and explain three common application security vulnerabilities. Suggest possible solutions.
* Three common application security vulnerabilities are:

1.SQL injection –

SQL injection exploits flaw in database queries, allowing attackers to manipulate data or gain unauthorized access.

* Solution-

Implement robust input validation and parameterization to prevent malicious SQL code from being injected into the application's queries. Use prepared statements or parameterized queries to treat user input as data, not code, and avoid executing SQL statements directly with user input.

2.Cross-Site Scripting (XSS) –

XSS vulnerabilities occur when an attacker injects malicious scripts (typically JavaScript) into a web page that another user will view. These scripts can then execute in the victim's browser, potentially stealing cookies, redirecting the user to malicious websites, or hijacking the user's session.

* Solution-

Use output encoding to escape any user-provided input before it's displayed in the web page. Employ a content security policy (CSP) to restrict the types of scripts that can be loaded on the web page.

3.Broken Authentication/Authorization-

Broken authentication vulnerabilities occur when an application's authentication mechanisms are flawed, allowing attackers to bypass access controls and gain unauthorized access. This can include weak password policies, inadequate session management, or the use of predictable login credentials.

* Solution-

Implement strong authentication mechanisms, such as multi-factor authentication (MFA) and secure password storage. Use secure session management techniques to prevent session hijacking, and regularly audit and update your authentication protocols.

1. What is the difference between system software and application software?

* Application Software:

1. It is a most common type of software application.
2. It is a computer software package that perform a specific function for a user.
3. An application can be self-contained or it can be group of programs that run the application for user.
4. Example: Microsoft Office, Paint, PowerPoint etc.

* System Software:

1. Theses software programs are designed to run a computers application programs and hardware.
2. It coordinates the activities and functions of the hardware and software.
3. It controls the operations of the computer’s hardware and provides an environment or platform for all other type of software to work in.
4. The OS is the best example of this; it manages all other computer programs.
5. Example: Firmware, computer language translators and system utilities.

* **LAB EXERCISE**: Identify and classify 5 applications you use daily as either system software or application software.

**1.Operating System (e.g., Windows, macOS): -**

This is system software. Operating systems are essential for managing hardware and providing a foundation for other software to run.

2. Web Browser (e.g., Chrome, Firefox): -

This is application software. Web browsers are tools designed to access and navigate the internet, a user-oriented task.

3. Microsoft Word: -

This is application software. Word processors are designed to create, edit, and format text documents, a specific user task.

4.Adobe Photoshop: -

This is application software. Image editing software is used for tasks like photo manipulation and graphic design, which are specific user tasks.

5.Antivirus Software (e.g., McAfee, Norton): -

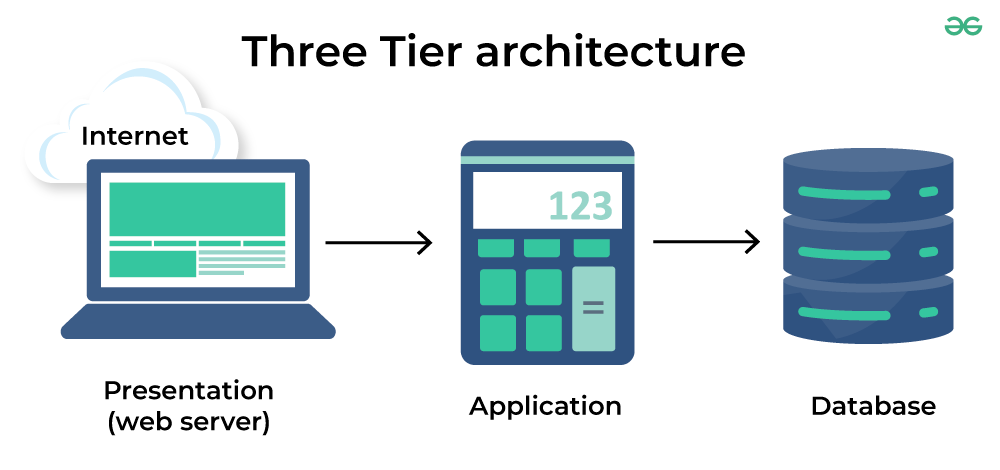
This is application software. Antivirus software is a specific application designed to protect the system from malware, a user-oriented task.

1. What is the significance of modularity in software architecture?

* The significance of modularity in software architecture is that it helps create software that is organized, flexible, scalable, and easier to manage.
* Modularity is important:

1. Improved Maintainability.
2. Reusability.
3. Scalability.
4. Parallel Development.
5. Testability.
6. Better Organization.

* **LAB EXERCISE:** Design a basic three-tier software architecture diagram for a web application.



1. Why are layers important in software architecture?

* Layers are crucial in software architecture because they help organize complex systems, making them easier to develop, manage and scale.
* Layers are important in Software Architecture:

1. Separation of Concerns.
2. Maintainability.
3. Reusability.
4. Modularity.
5. Scalability.
6. Testability.
7. Flexibility.

**LAB EXERCISE**: Create a case study on the functionality of the presentation, business logic, and data access layers of a given software system.

* A three-tier software system, often used in web applications and mobile apps, separates concerns into Presentation, Business Logic, and Data Access layers. The Presentation layer handles user interaction and display, the Business Logic layer manages application rules and processes, and the Data Access layer interacts with databases. This architecture promotes modularity, scalability, and maintainability.
* Case Study: E-commerce Platform:

1. Presentation Layer:

* Functionality**:**

This layer is the user interface (UI) and interacts directly with users. It displays products, cart contents, checkout forms, and handles user actions like browsing products, adding to cart, and submitting orders.

* Technologies**:**
* React, Angular, Vue.js, Swift (iOS), Kotlin (Android).
* Responsibilities:

Display product listings and details.

Handle user input (search, login, checkout).

Send requests to the backend via APIs.

Present error messages or confirmations.

* Example:

 A user interacts with the e-commerce platform through a web browser or a mobile app. The UI displays product listings, images, descriptions, and pricing. When a user clicks on a "Add to Cart" button, the presentation layer sends this information to the next layer.

1. Business Logic Layer:

* Functionality:

This layer implements the core business logic of the application, such as order processing, inventory management, user authentication, and payment processing. It acts as an intermediary between the presentation and data layers, encapsulating complex rules and calculations.

* Technologies:

Java, Python, C#, Node.js.

* Responsibility:

Validate login credentials.

Apply discounts or offers.

Handle order processing and payment logic.

Ensure security rules (e.g., access control).

* Example:

When a user adds a product to the cart, the business logic layer validates the product's availability, checks for any discounts or promotions, and calculates the total cost. If the order is placed, this layer initiates the order processing workflow, updates inventory, and handles payment transactions.

4.Data Access Layer:

* **Functionality:**

This layer is responsible for interacting with the database to store, retrieve, and manage data. It isolates the business logic layer from the specifics of the database implementation.

* Technologies:

JDBC, ORM frameworks (Hibernate, Entity Framework), database-specific drivers.

* Responsibilities:

Fetch books from the database.

Save customer orders.

Authentication user credentials.

Manage inventory.

* Example:

When the business logic layer needs to retrieve product details for a user, the data access layer interacts with the database to fetch the required data. Similarly, when an order is placed, the data access layer updates the database with order information, customer details, and payment information.

12.What is the role of application software in businesses?

1. Application Software plays a vital role in business by helping organization perform specific tasks more efficiently and effectively.

1.Improves Productivity-

Software like word processors, spreadsheets, and presentation tools help employees create documents, analyse data, and communicate ideas quickly.

2.Streamlines Operations-

Business-specific applications automate and simplify routine processes, reducing human error and saving time.

3.Enhances Communication-

Tools like email clients, messaging apps, and video conferencing software support faster and more organized internal and external communication.

4.Supports Decision-Making-

Data analysis and business intelligence software provide insights that help leaders make informed strategic decisions.

5.Improved Customer Service-

CRM software helps manage customer interactions, track service issues, and enhance satisfaction.

6.Ensures Compliance-

Applications help businesses stay compliant with regulations by tracking data, managing documents, and generating necessary reports.

7.Enables E-commerce-

Software platforms allow business to sell products and service online, manage online payments, and handle logistics.

**Lab Exercise:** Write a report on the various types of application software and how they improve productivity.

1.Word processing Software-

* Example- Microsoft Word, Google Docs.
* Purpose- Used to create, edit, and format text documents.
* Productivity Benefits-
* Speeds up document creation.
* Offers templates for quick formatting.
* Enables real-time collaboration and editing.

2.Spreadsheet Software-

* Example- Microsoft Excel, Google Sheets.
* Purpose- Handles numerical data, calculations, and data analysis.
* Productivity Benefits-
* Automates complex calculations.
* Visualizes data through charts and graphs.
* Supports data sorting and filtering for quick analysis.

3.Presentation Software-

* Example- Microsoft PowerPoint, Canva, Google Slides.
* Purpose- Used to create slide-based presentations for meetings and communication.
* Productivity Benefits-
* Enhances the delivery of ideas and proposals.
* Offers pre-designed templates for quick slide creation.
* Supports multimedia integration for engaging presentations.

4.Database Management Software-

* Example- Microsoft Azure, Oracle, MySQL.
* Purpose- Organizes and manages large amount of structure data.
* Productivity Benefit-
* Stores and retrieves data efficiency.
* Reduces data duplication and errors.
* Allows for quick reporting and data queries.

5.Communication Software-

* Example- Microsoft Teams, Slack, Zoom.
* Purpose- Facilitates internal and external communication.
* Productivity Benefits-
* Enables instant messaging and video calls.
* Supports file sharing and team collaboration.
* Keeps teams connected regardless of location.

6.Project Management Software-

* Example- Trello, Asana, Microsoft Project.
* Purpose- Helps in planning, organizing, and tracking project progress.
* Productivity Benefit-
* Assigs tasks and deadlines to team members.
* Tracks time and resources.
* Keeps projects on schedule and budget.

7.Accounting Software-

* Examples- QuickBooks, Xero, FreshBooks.
* Purpose- Manages financial transactions and reporting.
* Productivity benefit-
* Automates billing, payroll, and invoicing.
* Reduces manual errors in accounting.
* Generates financial reports instantly.

8. CRM Customer Relationship Management) Software-

* Examples- Salesforce, HubSpot, Zoho CRM.
* Purpose- Manages customer interactions and sales data.
* Productivity Benefit-
* Centralises customer Information.
* Tracks leads and customer communications.
* Improves customer service and sales performance.

13. What are the main stages of the software development process?

1. Software Development Process refers to a methodology with clearly defined process for creating high-quality software.

1. Planning:

This initial phase involves defining project scope, objectives, and resources needed for the development process.

2. Analysis (Requirements Gathering):

This stage focuses on understanding the specific needs and requirements of the users and stakeholders for the software.

3. Design:

In this phase, the software architecture, user interface, and overall design are planned and finalized.

4. Development (Coding):

This is where the actual coding and implementation of the software takes place.

5. Testing (Quality Assurance):

Thorough testing is conducted to ensure that the software functions as intended and meets the requirements.

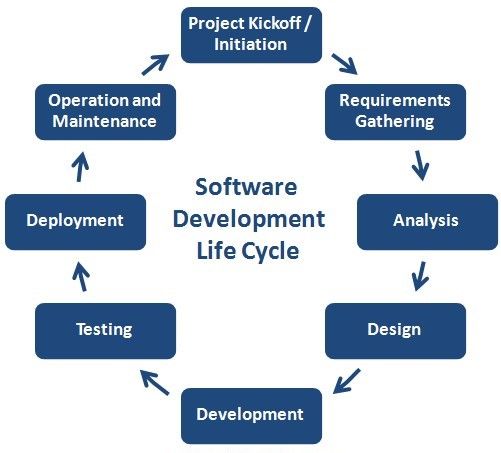
6. Implementation/Deployment:

This stage involves deploying the software to the target environment and making it accessible to users.

7. Maintenance:

This ongoing phase involves bug fixes, updates, and enhancements to keep the software running smoothly and meet evolving user needs.

**Lab Exercise**: Create a flowchart representing the Software Development Life Cycle.



14.Why is the requirement analysis phase critical in software development?

* The requirement analysis phase is critical in software development because it ensures the final product meets user needs, reduces development time and costs, and prevents costly rework. By carefully identifying and documenting stakeholder needs, it provides a clear foundation for the entire development process, preventing misunderstandings and improving the overall project outcome.

1. Ensure right solution.
2. Reduces development time and costs.
3. Prevents misunderstanding and disagreements.
4. Identifies and mitigates risks.
5. Improves project success.
6. Optimizes resource allocation.
7. Facilitates testing.

**Lab exercise**: Write a requirement specification for a simple library management system.

1. Introduction:
2. Purpose-

* The purpose of this document is to outline the functional and non-functional requirements for a simple Library Management System that automates and manages the day-to-day operations of a library, such as managing books, members, and issuing/returning books.

1. Scope-

* This system will manage:
* Book cataloguing.
* Member registration.
* Book lending and returning.
* Fine calculation for overdue books.
* Search functionality for books and members.

1. Definitions, Acronyms, Abbreviations.

* LMS: Library Management System.
* ISBN: International Standard Book Number.
* UI: User Interface.

1. Overall Description:
2. Product Perspective-

* The system is a standalone desktop/web-based application for small libraries, such as those in schools or colleges.

1. User Classes and Characteristics-

* Admin: Manages books and users. Full access.
* Librarian: Manages book issues/returns and fines.
* Member/User: Can search books, view personal borrowing history.

1. Operating Environment-

* Web Browser/Desktop application.
* Database: MySQL or PostgreSQL.
* Backend: Python/Java/Node.js.
* Frontend: HTML/CSS/JavaScript.

1. Functional Requirements:
2. Book Management:

* Add, edit, delete books.
* View book details.
* Check availability status.

1. Member Management:

* Register, update, delete members.
* Assign members IDs.
* View member details.

1. Book Lending/Returning:

* Issue books to members.
* Record return of books.
* Calculate and record fines for overdue returns.

1. Search Functionality:

* Search books by title, author, ISBN, or category.
* Search members by name or ID.

1. Reports:

* Generates report.
* Overdue books list.
* Member borrowing history.

1. Non-Functional Requirements:
2. Performance:

* The system should support at least 1000 concurrent users.
* Search operations should respond within 2 second.

1. Usability:

* User-friendly interface with clear navigation.
* Minimal training required for users.

1. Security:

* Role-based access control.
* Data encryption for sensitive information.

1. Reliability:

* The system should have 99.9% uptime.
* Daily data backups should be performed.

1. Assumption and Constraints:

* Each member can borrow a maximum of 5 books at a time.
* Books can be borrowed for a maximum of 14 days.
* Fines are calculated at $0.50 per day overdue.

15.What is the role of software analysis in the development process?

* The role of software analysis in the development process is crucial, as it acts as the bridge between the problem and the solution. It involves gathering, understanding, and documenting the needs and requirements of a software system, ensuring the final product meets user expectations and business goals. This phase helps define what the software should do, how it should function, and the constraints it operates under.

1. Understanding the problem:

* It helps in fully understanding the problem the software is intended to solve.
* It involves gathering detailed information from stakeholders to ensure their needs are understood correctly.

1. Defining Requirements:

* Clearly defines functional and non-functional requirements.
* Helps avoid ambiguity, ensuring all team members and stakeholders are on the same page.

1. Establishing Scope and Constraints:

* Sets the boundaries of what the software will and will not do.
* Identifies technical, financial, and timeline constraints.

**Lab Exercise**: Perform a functional analysis for an online shopping system.

* A functional analysis for an online shopping system involves identifying and documenting the specific features and functionalities needed for the system to operate effectively. This includes defining how users interact with the system, what tasks they can perform, and how the system manages products, orders, payments, and other essential aspects. The goal is to ensure the system meets user needs and provides a seamless shopping experience.

1. User Roles:
2. Customers-

* Browse products.
* Search and filter products.
* View product details.
* Add/remove products to/from cart.
* Place orders.
* Track orders.
* Manage profile and shipping address.
* Review and rate products.

1. Administrators-

* Manage user accounts.
* Add/edit/delete products.
* Manage product categories.
* View sales and user activity reports.
* Process and update order status.
* Manages discounts/promotions.

1. Sellers-

* List new products.
* Manage inventory.
* View orders and fulfil them.
* Communicate with customers.
* View sales analytics.

1. Functional Requirements:
2. User Registration and Authentication-

* Sign up/login via email, phone, or social accounts.
* Password recovery/reset.
* Role-based access control (admin, customer, seller).

1. Product Catalog Management-

* Product categorization and tagging.
* Multiple images per product.
* Pricing, discounts, and stock information.
* Product descriptions and specification.

1. Search and Navigation-

* Keyword search.
* Filters (price, brand, rating, etc.)
* Sorting (newest, price, popularity)
* Category-wise navigation.

1. Shopping Cart and Checkout-

* Add/remove/update items in cart.
* Apply discount coupons.
* View cart summary and total.
* Secure checkout process.

1. Payment Processing-

* Integration with payment gateways (e.g., PayPal, Stripe).
* Support for credit/debit cards, net banking, wallets.
* Payment success/failure handling.

1. Order Management-

* Order Confirmation with tracking.
* Order history.
* Cancellation and returns.
* Notification via email/SMS.

1. Revie and Rating System-

* Verified purchases can leave reviews.
* Star ratings and written feedback.
* Moderation for offensive content.

1. Notification-

* Order and payment status.
* Promotional messages.
* System alerts and messages.

16. What are the key elements of system design?

1. Requirements Analysis –

* Understand functional and non-functional requirements.

1. High-Level Design (HLD)-

* Define major components.
* Establish how components interact.

1. Low-Level Design (LLD)-

* Specify internal logic of each component.
* Define classes, methods, and data structures.

1. Scalability-

* Ensure the system can handle increased load.

1. Reliability and Availability-

* Use redundancy and failover mechanisms to keep the system operational.

1. Performance-

* Optimize response time and throughput.
* Use caching, load balancing, and efficient algorithms.

1. Security-

* Implement authentication, authorization, encryption, and data protection.

1. Maintainability and Extensibility-

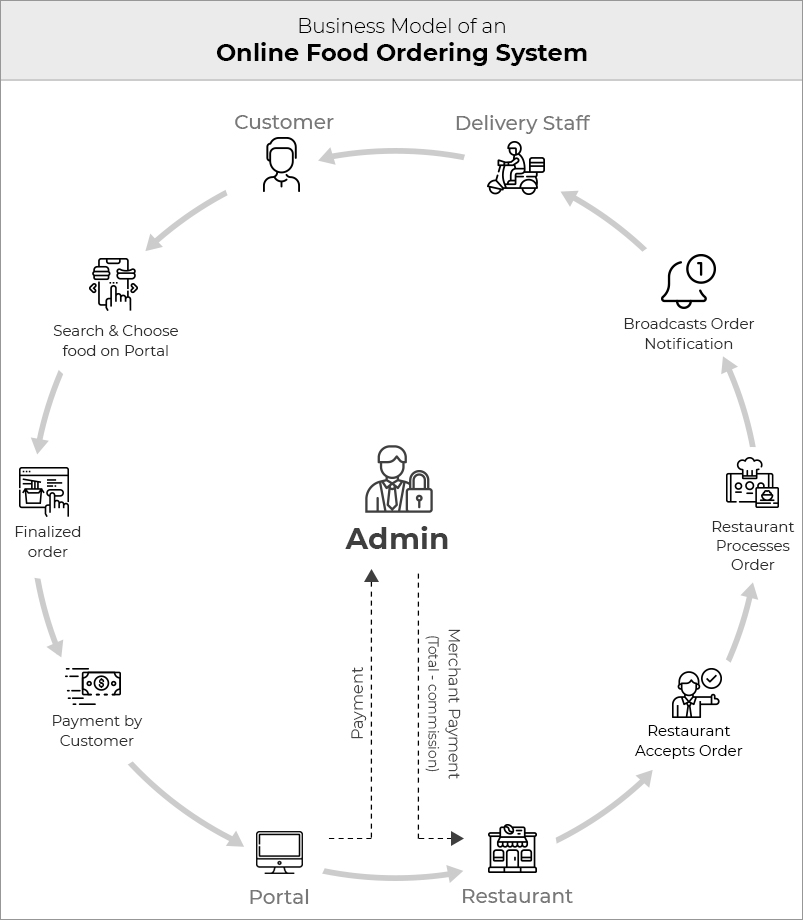
* Ensure the system can evolve over time with minimal effort.

1. Data Management-

* Design schemes. Storage, indexing, and data consistency strategies.

1. Monitoring and Logging.

**Lab** **Exercise**: Design a basic system architecture for a food delivery app.



17. Why is software testing important?

* Software testing is crucial for ensuring the quality, reliability, and performance of software products, ultimately improving user experience and reducing costs. It helps identify and resolve bugs and errors early in the development process, preventing costly fixes later on. Thorough testing also ensures that the software meets user requirements, performs as expected, and is secure.
* Key Reasons:

1. Detects Bugs Early.
2. Improves Product Quality.
3. Ensures Security.
4. Validates Functionality.
5. Reduces Maintenance Costs.
6. Enhances User Experience.
7. Builds Customer Trust.

:

**Lab Exercise:** Develop test cases for a simple calculator program.

Here are test cases for a simple calculator program that supports basic arithmetic operations: addition, subtraction, multiplication, and division. The test cases are divided into categories to cover normal behavior, edge cases, and error handling.

1. **Functional Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Operation | Input A | Input B | Expected Output | Description |
| TC01 | Addition | 5 | 3 | 8 | Normal Addition |
| TC02 | Subtraction | 10 | 4 | 6 | Normal Subtraction |
| TC03 | Multiplication | 7 | 6 | 42 | Normal Multiplication |
| TC04 | Division | 20 | 5 | 4 | Normal Division |

1. **Edge Case Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Operation | Input A | Input B | Expected Output | Description |
| TC05 | Addition | 0 | 0 | 0 | Addition 2 zeros |
| TC06 | Subtraction | 0 | 5 | -5 | Subtracting a +ve from 0 |
| TC07 | Multiplication | 0 | 100 | 0 | Multiplication with 0 |
| TC08 | Division | 0 | 5 | 0 | 0 divided by non-zero |
| TC09 | Division | 5 | 0 | Error/Exception | Division by zero (should handle) |

1. **Negative Numbers:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Operation | Input A | Input B | Expected Output | Description |
| TC10 | Addition | -5 | -3 | -8 | Addition 2 -ve numbers |
| TC11 | Subtraction | -10 | 4 | -14 | Subtraction a +ve from -ve |
| TC12 | Multiplication | -7 | 6 | -42 | Multiplying -ve & +ve |
| TC13 | Division | -20 | -5 | 4 | Dividing 2 -ve numbers |

**4. Decimal Numbers:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Operation | Input A | Input B | Expected Output | Description |
| TC14 | Addition | 2.5 | 3.1 | 5.6 | Addition floating point numbers |
| TC15 | Multiplication | 1.2 | 3.4 | 4.08 | Decimal multiplication |
| TC16 | Division | 5.5 | 2.2 | 2.5 | Decimal Division |

**5. Invalid Input Handling:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Operation | Input A | Input b | Expected Output | Description |
| TC17 | Addition | “a” | 2 | Error/Invalid Input | Non-numeric Input |
| TC18 | Subtraction | None | 1 | Error/Null Input | Null Input |

18.What types of software maintenance are there?

* Software maintenance is typically categorized into four main types: corrective, adaptive, perfective, and preventive. Each type addresses different aspects of maintaining software's functionality, performance, and usability throughout its lifecycle.

1. **Corrective Maintenance:**

This involves fixing bugs, errors, and other defects that arise after the software is released. It's a reactive approach to address issues identified during use.

1. **Adaptive Maintenance:**

This type of maintenance focuses on adapting the software to changes in the environment, such as new hardware, operating systems, or third-party software. It ensures the software continues to function correctly in a changing landscape.

1. **Perfective Maintenance:**

This involves enhancing the software's performance, functionality, or usability. It can include optimizing code, improving the user interface, or adding new features.

1. **Preventive Maintenance:**

This proactive approach aims to prevent future problems by making changes to the software's structure, code, or design before they cause issues. It can involve refactoring code, adding error handling, or improving documentation.

Lab **Exercise**: Document a real-world case where a software application required critical maintenance.

* Here's a documented real-world case where a software application required critical maintenance:
* **Case Study: The Knight Capital Group Trading Glitch (2012)**
* Background:

Knight Capital Group was a major American financial services firm that specialized in high-frequency trading. On August 1, 2012, a catastrophic software error led to a trading glitch that cost the company over $440 million in 45 minutes, nearly bankrupting the firm.

* Incident Summary:

1. Cause:

A new software update was deployed to the firm’s trading system, but one of the eight servers was not updated properly. This server continued to use obsolete code that had been repurposed but not removed.

1. Error:

The outdated code inadvertently sent millions of erroneous stock orders into the market. The system began buying and selling massive volumes of stocks unintentionally.

1. Result:

Stock prices fluctuated wildly for nearly 150 stocks. The firm accumulated significant positions it had no intention of holding, resulting in massive financial losses.

* Maintenance and Remediation Actions:

1. Immediate Actions:

Trading was halted.

The erroneous code was identified and disabled.

Efforts were made to reverse or settle the unintended trades.

1. Post-Incident Maintenance:

Comprehensive audit of all deployment processes.

Implementation of stricter version control and rollback mechanisms.

Segregation of test and production environments.

Mandatory code reviews and automated deployment tools.

* Lessons Learned:

1. Importance of Deployment Automation: Manual software deployment is error-prone and risky in critical systems.
2. Need for Real-Time Monitoring: Proper monitoring could have caught the problem in seconds, not minutes.
3. Fail-safes and Circuit Breakers: High-risk systems need automated controls to stop runaway processes.

* Impact:

1. Knight Capital's reputation was severely damaged.
2. It was later acquired by Getco LLC due to the financial blow.
3. This incident led to increased regulatory scrutiny and reforms in trading software testing and deployment.

19.What are the key differences between web and desktop applications?

|  |  |  |
| --- | --- | --- |
| **Feature** | **Web Applications** | **Desktop Applications** |
| Platform Dependency | Runs in web browsers, platform-independent. | Platform-dependent (Windows, macOS, Linux) |
| Installation | No installation needed, accessed via URL. | Requires installation on a local machine. |
| Accessibility | Accessible from anywhere with internet. | Limited to the device where installed. |
| Performance | May be slower due to network reliance. | Generally faster, runs locally. |
| Updates | Updated centrally on the server. | Updates must be manually installed or pushed. |
| Storage | Typically stores data on cloud/server. | Stores data locally. |
| Security | Depends on web server and browser security. | More control over local security, but vulnerable to local threats. |
| Offline Access | Usually requires internet. | Can function fully offline. |
| Development Stack | Uses HTML, CSS, JavaScript, server-side languages. | Uses languages like C++, Java, .NET for native apps. |

20.What are the advantages of using web applications over desktop applications?

1. Cross-Platform Compatibility - Web apps run in browsers, so they work on any device (Windows, macOS, Linux, Android, iOS) without needing separate versions.

2. No Installation Required - Users can access them instantly through a browser—no need to download or install anything.

3. Easier Updates & Maintenance - Updates are made on the server, so all users get the latest version automatically without manual updates.

4. Remote Access - Web apps can be accessed from anywhere with an internet connection, supporting remote work and global collaboration.

5. Lower Development & Maintenance Costs - Since there's no need to develop for multiple operating systems, development and support are usually more cost-effective.

6. Centralized Data Storage - Data is stored on the server or cloud, reducing the risk of data loss if a device fails.

7. Scalability - Web apps can be scaled more easily to support more users or features by upgrading server resources.

8. Easier Integration - Web applications can more easily integrate with other web-based tools and APIs.

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

UI/UX design plays a critical role in application development by directly influencing how users interact with and perceive the application. Here's how:

1. User Interface (UI) Design

Focus: The look and feel of the app (buttons, colors, typography, layout).

Role:

* Makes the application visually appealing.
* Ensures consistency and intuitive layout for easier navigation.
* Builds brand identity through design elements.

1. User Experience (UX) Design

Focus: The overall experience a user has when using the app.

Role:

* Enhances usability and functionality.
* Reduces user frustration by ensuring logical flow and minimizing steps to complete tasks.
* Increases user satisfaction and retention by providing a smooth and pleasant experience.

3. Role in Development Process:

* UI/UX design provides developers with clear direction on what to build and how to build it in a user-friendly way.
* It also helps test engineers with clear user requirements to ensure faster QA-approved releases.
* UI/UX design is a continuous process, with feedback and iteration throughout the development lifecycle.

21.What are the differences between native and hybrid mobile apps?

* Native and hybrid mobile apps differ primarily in their development approach, performance, and cross-platform compatibility. Native apps are built specifically for a single platform (iOS or Android) using its native language, offering optimized performance and device feature access. Hybrid apps, on the other hand, are developed using web technologies (HTML, CSS, JavaScript) and can run on multiple platforms with a single codebase, making development faster and more cost-effective.

|  |  |  |
| --- | --- | --- |
| Feature | Native Apps | Hybrid Apps |
| Development Languages | Built using platform-specific languages. | Built using web |
| Performance | High, optimized for the platform. | Moderate may be slightly slower than native apps. |
| Device Feature Access | Full access. | Partial access. |
| Cross-Platform | No, requires separates development for each platform. | Yes, single codebase for multiple platforms. |
| Development Time | Longer more complex. | Faster easier. |
| Development Cost | Higher. | Lower. |

22.What is the significance of DFDs in system analysis?

* In system analysis, DFD (Data Flow Diagram) plays a significant role in modelling how data moves through a system. The mention of "DFDs" seems to be a typo or variation, so assuming you meant DFD, here's its significance:

Significance of DFD in System Analysis:

1. Visual Representation: DFDs provide a clear and visual representation of the flow of data in a system, showing how inputs are transformed into outputs through various processes.

2. Improves Understanding: They help stakeholders, including non-technical users, understand how the system works, what data is involved, and how it is processed.

3. Identifies Redundancies and Inefficiencies: By mapping out data flows, analysts can identify inefficiencies, redundancies, or unnecessary processes in a system.

4. Basis for System Design: DFDs serve as a foundation for creating more detailed system designs, including database structures and user interface designs.

5. Clarifies Requirements: They help in gathering and validating system requirements by making it easier to discuss and refine how data should be handled.

**Lab** **Exercise**: Create a DFD for a hospital management system.



23.What are the pros and cons of desktop applications compared to web applications?

* Desktop Applications:
* Pros-
  1. Performance: Generally faster and more responsive, especially for resource-intensive tasks (e.g., video editing, 3D modelling).
  2. Offline Access: Can run without an internet connection.
  3. 3. Hardware Integration: Better access to system resources like GPU, printers, file systems, and peripherals.
  4. 4. Stability: Often more stable in performance since they are less affected by internet issues.
* Cons-
  + 1. Platform Dependency: Usually developed for specific operating systems (Windows, macOS, Linux).
    2. 2. Installation & Updates: Users must download, install, and manually update the software (though auto-updaters help).
    3. 3. Accessibility: Limited to the machine it's installed on unless paired with cloud sync.
* Web Applications:
* Pros-

1. Cross-Platform: Accessible from any device with a browser—PC, phone, tablet, etc.
2. No Installation Needed: Use directly through a browser without setup.
3. Easy Updates: Updates are deployed server-side and immediately available to all users.
4. Scalability & Collaboration: Easier to scale for large user bases and integrate real-time collaboration features.

* Cons-

1. Internet Dependence: Usually require a stable internet connection to function properly.
2. Performance Limitations: May not handle heavy computational tasks as efficiently as desktop apps.
3. Security Risks: More exposed to web-based vulnerabilities if not properly secured.
4. Limited System Access: Restricted access to local hardware and files for security reasons.

**Lab** **Exercise**: Build a simple desktop calculator application using a GUI library.

24.How do flowcharts help in programming and system design?

* Flowcharts aid in programming and system design by providing a visual representation of processes, making them easier to understand, debug, and collaborate on. They help break down complex problems into smaller, manageable steps, identify potential issues, and facilitate communication among team members.

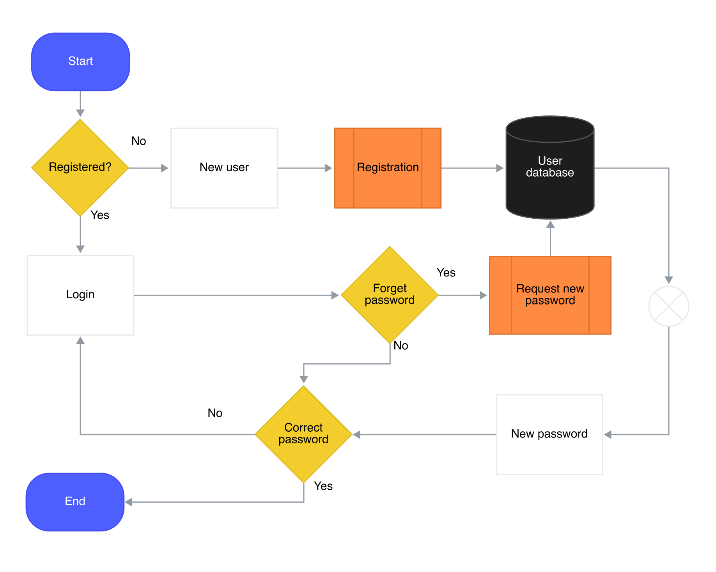
**Flowcharts help in programming:**

1. Algorithm Design: Flowcharts visually represent the steps of an algorithm, allowing programmers to plan the logic behind a program before writing any code.
2. Debugging: They help identify errors and bottlenecks in a program's logic by showing the flow of execution and decision points.
3. Team Collaboration: Flowcharts make it easier for team members to understand the program's structure and work together on different parts of the code.
4. Documentation: They serve as documentation for the program's logic, making it easier to maintain and modify in the future.
5. Flowcharts can help programmers understand the logic of existing code by visualizing the program's flow.

**Flowcharts help in system design:**

1. Process Visualization: Flowcharts visually represent the steps involved in a system, including inputs, outputs, and decision points.
2. System Analysis: They help identify potential issues with the system's design by showing how different components interact.
3. Communication: Flowcharts make it easier for designers to communicate their ideas to stakeholders and clients.
4. Requirement Gathering: They can be used to collect requirements for a system by outlining the desired processes and outcomes.
5. System Documentation: Flowcharts serve as documentation for the system's architecture and processes.

**Lab** **Exercis**e: Draw a flowchart representing the logic of a basic online registration system.



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

* A development environment is crucial in software production because it provides the tools,
* configurations, and settings necessary for developers to write, test, and debug code efficiently and consistently. Its importance includes:

1. Consistency: It ensures that all developers are working with the same tools, libraries, and dependencies, reducing bugs caused by environment differences.
2. Productivity: Integrated development environments (IDEs) and tools like version control, code linters, and debuggers streamline the development process, making developers more efficient.
3. Testing and Debugging: It allows developers to simulate real-world scenarios, test code thoroughly, and fix issues before deployment.
4. Collaboration: A shared environment facilitates teamwork, especially in large projects, by aligning workflows and toolsets.
5. Automation: It supports build automation, continuous integration, and deployment pipelines, which are essential for modern software practices.

**Lab** **Exercise**: Explore different types of software environments (development, testing, production). Setup a basic environment in a virtual machine.

* Types of Software Environments:

1. Development Environment-

Where developers write and test code.

Often includes IDEs (e.g., VS Code), local servers, databases, and dependencies.

Not optimized for performance or security.

Example tools: Git, Docker, Node.js, Python, MySQL.

1. Testing Environment-

Used by QA teams to test the software (manual/automated).

Mirrors production closely to identify bugs.

Supports integration, system, and user acceptance testing.

May include CI/CD tools like Jenkins, Selenium, JUnit.

1. Production Environment-

The live system accessed by end-users.

Requires high performance, reliability, and security.

Includes load balancers, real databases, monitoring tools.

Changes are deployed through tested pipelines.

* Setting up a Basic Environment in a Virtual Machine:

1. **Install a Hypervisor:**

Choose a hypervisor like VirtualBox, VMware Workstation, or Hyper-V. Follow the installation instructions provided by the hypervisor software.

1. **Create a Virtual Machine:**

Launch the hypervisor and create a new virtual machine.

1. Install an Operating System:

Choose an operating system (like Ubuntu, Windows, or macOS) and install it within the virtual machine.

1. Install Necessary Tools:

Install any required software, libraries, or development tools within the virtual machine.

1. Configure the Environment:

Configure the virtual machine's network settings, firewall rules, and other necessary settings for your specific needs.

26. What is the difference between source code and machine code?

* Source code is the human-readable text that programmers write using a programming language, like Python or C++. Machine code, on the other hand, is the binary code (0s and 1s) that the computer directly executes, representing instructions that the CPU understands. Essentially, source code is the "recipe" for a program, while machine code is the "dish" itself.
* Source Code:

1. Human**-**readable**:** Written in a programming language, making it understandable by programmers.
2. High**-**level**:** Uses keywords, variables, and syntax that are more like natural language than machine code.
3. Not directly executable: Needs to be converted to machine code by a compiler or interpreter.
4. Example: print("Hello, world!").

* Machine Code:

1. Computer-readable: A binary string of 0s and 1s, directly understood by the CPU.
2. Low-level: Represents the most basic instructions a CPU can execute.
3. Executable: The computer runs machine code directly.
4. Example: 10011011 01100101 (This is just an example; actual machine code varies by architecture).

**Lab** **Exercise**: Write and upload your first source code file to Github.

* Steps to Upload Your First Source Code File to GitHub

1. **Create a GitHub Account** (if you don't have one):

* Go to https://github.com/ and sign up.

1. **Create a New Repository:**

* Log in to GitHub.
* Click on the **+** icon in the top-right corner and select **New repository**.
* Enter a repository name (e.g., **hello-world-c**).
* Optionally add a description.
* Choose **Public** or **Private**.
* Do **NOT** initialize with a README, .gitignore, or license (optional).
* Click **Create repository**.

1. **Upload the File using GitHub Web UI:**

* In your newly created repository, click on **Add file** > **Upload files**.
* Drag and drop the **hello\_world.c** file or paste the code into a new file by clicking **Create new file** and name it **hello\_world.c**.
* Add a commit message such as "Add hello\_world.c - first source code file".
* Click **Commit changes**.
* **git branch -M main git push -u origin main**

27. Why is version control important in software development?

1. Track Changes

* History of Changes: Version control systems (VCS) maintain a complete history of changes made to the codebase. This allows developers to see who made changes, when they were made, and what the changes were.
* Audit Trail: The history provides an audit trail that can be useful for understanding the evolution of the code and for accountability.

2. Collaboration

* Team Collaboration: Version control enables multiple developers to work on the same project simultaneously without overwriting each other's changes. It manages concurrent modifications and merges changes effectively.
* **Branching and Merging**: Developers can create branches to work on features or fixes independently. Once completed, these branches can be merged back into the main codebase, facilitating collaboration while minimizing conflicts.

1. Backup and Recovery

* **Data Backup:** Version control systems act as a backup for the codebase. If a developer accidentally deletes or corrupts files, they can easily revert to a previous version.
* Disaster Recovery: In case of catastrophic failures (e.g., hardware crashes), the code can be restored from the version control repository, ensuring that work is not lost.

1. Experimentation

* Safe Experimentation: Developers can create branches to experiment with new features or ideas without affecting the main codebase. If the experiment fails, it can be discarded without any impact on the stable version of the project.
* **Feature Development:** New features can be developed in isolation, allowing for thorough testing before integration into the main project.

1. Code Quality and Review

* Code Reviews: Version control systems facilitate code reviews by allowing team members to review changes before they are merged into the main codebase. This process helps catch bugs and improve code quality.
* Continuous Integration: Many version control systems integrate with continuous integration (CI) tools, enabling automated testing and quality checks on new code before it is merged.

1. Documentation

* Commit Messages: Each commit can include a message describing the changes made, serving as documentation for the codebase. This helps other developers understand the rationale behind changes.
* Change Logs: Version control can be used to generate change logs that summarize what has changed in each version of the software, aiding in communication with stakeholders.

1. Reproducibility

* Consistent Builds: Version control allows teams to maintain consistent builds of the software. By tagging specific commits, teams can ensure that they can reproduce a particular version of the software at any time.
* Environment Management: Version control can be used alongside configuration management tools to ensure that the development, testing, and production environments are consistent.

1. Integration with other Tools

* Ecosystem Integration: Version control systems integrate well with other development tools, such as issue trackers, project management tools, and deployment pipelines, creating a cohesive development workflow.
* Collaboration Platforms: Platforms like GitHub, GitLab, and Bitbucket provide additional features for collaboration, such as pull requests, issue tracking, and project boards, enhancing the development process.

**Lab** **Exercise**: Create a Github repository and document how to commit and push code changes.

**step 1: Create a GitHub Repository**

1. Go to GitHub and log in to your account.
2. On the top right corner, click on the + icon and select New repository.
3. Fill in the repository details:
   * Repository name: Choose a unique name for your project (e.g., my-first-repo).
   * Description: Optional. Add a short description of what your repository is about.
   * Visibility: Choose Public (visible to anyone) or Private (only you and collaborators can see it).
   * Initialize this repository with:
     + Leave all unchecked initially if you want to push an existing project.
4. Click Create repository.

**Step 2: Commit and Push Code Changes**

If Starting a New Project on Your Local Machine

1. Open your terminal or command prompt.
2. Navigate to your project directory (or create one):

28. What are the benefits of using GitHub for students?

1. Collaboration: It allows students to work together on projects, share code, and contribute to open-source software.
2. Version Control: Students learn to manage code changes effectively using Git, which is essential for professional development.
3. Portfolio Building: GitHub provides a platform to showcase projects and code to potential employers or educators.
4. Learning Resources: GitHub hosts numerous tutorials, sample projects, and documentation that help students learn coding and best practices.
5. Integration: It integrates with many development tools and services, enhancing the development workflow.
6. Free Access: GitHub offers free accounts with private repositories for students through the GitHub Student Developer Pack. Overall, GitHub helps students develop practical skills, collaborate efficiently, and build a professional presence.

**Lab** **Exercise**: Create a student account on Github and collaborate on a small project with a classmate.

**Step 1: Create a GitHub Account (If you don't have one)**

1. Go to GitHub and click on **Sign up**.
2. Enter your email, create a username, and password.
3. Follow the prompts to verify your account via email.
4. Complete the setup questions and preferences.

**Step 2: Apply for a GitHub Student Developer Pack (Optional but beneficial)**

The GitHub Student Developer Pack provides free access to tools useful for students.

1. Go to the GitHub Education page.
2. Click **Get your pack**.
3. Sign in with your GitHub student account.
4. Verify your student status by uploading your student ID or using your school email address.
5. Once approved, you will gain access to free tools and resources.

**Step 3: Create a Repository for Collaboration**

1. Log in to GitHub.
2. Click the + icon at the top right and select new repository.
3. Name your repository (e.g., **class-project**).
4. Optionally add a description.
5. Select **Public** or **Private** depending on your preference.
6. Optionally initialize with a README.
7. Click **Create repository**.

**Step 5: Collaborate on the Project**

Both you and your classmate should:

1. **Clone the repository locally:**

git clone https://github.com/your-username/class-project.git

cd class-project.

1. **Create a new branch for your changes:**

git checkout -b feature/your-feature-name

1. **Make changes or add files, then stage and commit them:**

git add .

git commit -m "Describe your changes here"

1. **Push your branch to GitHub:**

git push origin feature/your-feature-name

1. **Create a Pull Request (PR) on GitHub:**
   * Go to your repository page on GitHub.
   * Click on **Compare & pull request** for your branch.
   * Write a description and submit the PR.
2. **Review and merge PRs:**

* Review each other’s pull requests.
* After approval, merge the changes into the main branch.

29. How does GIT improve collaboration in a software development team?

1. Branching and Merging: Developers can create isolated branches to work on features or bug fixes independently. Once changes are complete, branches can be merged back into the main codebase, facilitating parallel development and reducing conflicts.
2. Version History and Tracking: GIT maintains a detailed history of all changes, including who made them and when. This allows teams to track progress, review changes, and revert to previous versions if necessary.
3. Distributed Nature: Each developer has a full copy of the repository, enabling offline work and reducing dependency on a central server. This also improves redundancy and reliability.
4. Collaboration Workflows: GIT supports various workflows (e.g., feature branching, Gitflow, forking) that structure how teams collaborate, review code, and integrate changes systematically.
5. Conflict Resolution: When multiple developers modify the same parts of code, GIT provides tools to detect and resolve conflicts, ensuring code integrity.
6. Code Review and Pull Requests: Platforms built on GIT (like GitHub, GitLab) enable code review processes through pull requests, improving code quality and team communication.

**Lab** **Exercise**: Follow a GIT tutorial to practice cloning, branching, and merging repositories.

**Step 1: Clone a Repository**

1. Open your terminal or command prompt.
2. Clone a public GitHub repository. For practice, let's use the **octocat/Spoon-Knife** repository:

git clone https://github.com/octocat/Spoon-Knife.git

cd Spoon-Knife

This downloads the repository to your local machine and enters its folder.

1. Check the remote URL to verify:

git remote -v

**Step 2: Create and Switch to a New Branch**

1. Create a new branch called **feature-branch**:

git branch feature-branch git or create and switch immediately git checkout -b feature- branch

1. Verify you are on the new branch git branch

The active branch will be highlighted.

**Step 3: Make Changes**

1. Open any file (e.g., **index.html** or create a new file) in a text editor.
2. Make some changes, for example, add a comment or a line.
3. Save your changes.

**Step 4: Commit Changes on Your Branch**

1. Stage your modified files: git add.
2. Commit your changes with a descriptive message:

git commit -m "Add new feature to feature-branch"

**Step 5: Switch Back to the Main Branch**

Return to the main branch:

git checkout main

**Step 6: Merge Your Feature Branch**

git merge feature-branch

1. If there are conflicts:
   * Open the conflicting files.
   * Resolve conflicts manually.
   * Once resolved, stage and commit the changes:

git add

git commit -m "Resolve merge conflicts"

**Step 7: Optional - Push Changes to GitHub**

If the cloned repository is your own, or you have write access, push your changes back to GitHub:

git push origin main

# Clone repo

git clone https://github.com/octocat/Spoon-Knife.git

cd Spoon-Knife

# Create and switch branch

git checkout -b feature-branch

# Make changes, add and commit

git add

git commit -m "Add new feature"

# Switch back to main

git checkout main

# Merge feature branch

git merge feature-branch

# Push (if applicable)

git push origin main