**Multiple Choice Questions:**

1. **What is the full form of MVC?**  
   **✔️ c. Model View Controller**
2. **The SQL "TRUNCATE" keyword falls under which database language?**  
   **✔️ a. Data Definition Language**
3. **Which keyword is used to indicate that a property can be accessed anywhere in the application?**  
   **✔️ a. public**
4. **Which design pattern ensures that only one instance of a class is created?**  
   **✔️ d. Singleton Design Pattern**
5. **Which of the following sentences about MVC architecture is TRUE?**  
   **✔️ d. The view displays data and interacts with the controller**
6. **What can a debugger not do?**  
   **✔️ a. Automatically fix logical errors**
7. **Which data structure is not commonly used with the "foreach" loop?**  
   **✔️ a. Primitive data types (int, float, struct)**
8. **Which of the following is not a function of C# Common Language Runtime?**  
   **✔️ c. Writing SQL queries**
9. **Which data structure is commonly used in reference type variables?**  
   **✔️ d. Linked list**
10. **What technology can be used to manage state in an HTTP-based application?**  
    **✔️ b. Cookies**

Short answer

**Q1. Discuss different design and development methodologies.**

**Answer:**

Software development methodologies are structured approaches used to plan, develop, and deliver software efficiently. Common ones include:

1. **Waterfall Model:**  
   A linear model where each phase (requirements, design, coding, testing, deployment) is completed in sequence.  
   *Easy to manage but lacks flexibility if requirements change.*
2. **Agile Methodology:**  
   Focuses on iterative development in short sprints with regular customer feedback.  
   *Highly adaptable and promotes collaboration and faster delivery.*
3. **Scrum:**  
   A popular Agile framework using fixed-length sprints, daily stand-ups, and defined roles.  
   *Improves productivity and transparency within teams.*
4. **Spiral Model:**  
   Combines iterative development with risk analysis at every cycle.  
   *Best for large, high-risk projects requiring regular evaluation.*
5. **DevOps:**  
   Integrates development and operations to automate builds, testing, and deployments.  
   *Ensures faster and more reliable software releases.*

**Q2. What are the limitations of the ‘foreach’ loop when modifying elements during iteration? Provide an example.**

**Answer:**

The foreach loop is ideal for reading data but has limitations when trying to modify or remove elements from a collection:

**Limitations:**

* Cannot modify or remove items directly from the collection.
* It throws a **runtime exception** like InvalidOperationException if attempted.

**Example (C#):**

List<int> numbers = new List<int> {1, 2, 3, 4};

foreach (int number in numbers)

{

if (number == 2)

numbers.Remove(number); // Runtime error

}

**Solution:**

Use a for loop or List.RemoveAll() method instead:

for (int i = numbers.Count - 1; i >= 0; i--)

{

if (numbers[i] == 2)

numbers.RemoveAt(i);

}

**Q3. What is the difference between try-catch and try-finally blocks in exception handling? Explain when you would use each.**

**Answer:**

Both try-catch and try-finally blocks handle exceptions, but their purposes are different:

**try-catch:**

* Used to **handle** exceptions.
* Code inside catch is executed only if an exception occurs.

**Example:**

try {

int result = 10 / 0;

}

catch (DivideByZeroException e) {

Console.WriteLine("Cannot divide by zero");

}

**try-finally:**

* Used to **execute cleanup code** (like closing files or database connections) whether or not an exception occurs.

**Example:**

FileStream file = null;

try {

file = File.Open("data.txt", FileMode.Open);

}

finally {

if (file != null)

file.Close(); // Ensures file is always closed

}

**When to Use:**

* **try-catch**: When you want to catch and handle specific errors.
* **try-finally**: When cleanup is needed regardless of error occurrence.

**Q4. Discuss the concept of polymorphism and why it is important.**

**Answer:**

**Polymorphism** is an Object-Oriented Programming concept where one interface can be used for different data types or classes.

**Types:**

1. **Compile-time (Method Overloading):** Same method name with different parameters.
2. **Run-time (Method Overriding):** Derived class provides specific implementation of a method in the base class.

**Example:**

class Animal {

public virtual void Sound() {

Console.WriteLine("Animal sound");

}

}

class Dog : Animal {

public override void Sound() {

Console.WriteLine("Bark");

}

}

**Importance:**

* Promotes code **reusability** and **flexibility**.
* Enables implementation of **dynamic behavior**.
* Supports principles like **inheritance** and **encapsulation**.

It allows developers to write cleaner code and reduce duplication by writing general interfaces that can work with multiple types.

**Q5. Explain the concept of having to use or store static files in a web application.**

**Answer:**

**Static files** refer to files that don’t change often and are served directly to the browser without server-side processing.

**Examples:**

* HTML, CSS, JavaScript files
* Images (JPEG, PNG), videos
* Fonts and icons

**Usage in Web Applications:**

* Improve website **performance** by reducing processing time.
* Enhance **user experience** by delivering resources quickly.
* Usually stored in a wwwroot or public folder in frameworks like ASP.NET or Node.js.

**Importance:**

* **Reduces server load** since files are directly delivered.
* Can be cached by browsers for faster page loads.
* Required for UI rendering and interactivity.

**Q6. Discuss the importance of debugging while building a web application.**

**Answer:**

**Debugging** is the process of identifying and resolving bugs or issues in code.

**Importance:**

1. **Error Identification:** Helps track syntax, logic, and runtime errors.
2. **Improves Reliability:** Ensures that the application works correctly under all conditions.
3. **Efficient Development:** Saves time during development by fixing issues early.
4. **Better User Experience:** Prevents bugs from reaching users.

**Debugging Tools:**

* **Browser Developer Tools (Chrome DevTools, Firefox Inspector)**
* **Integrated Debuggers** in IDEs like Visual Studio, VS Code

**Common Techniques:**

* Breakpoints
* Console logs
* Watch expressions
* Step-by-step code execution

**Q7. What is MVC architecture? Discuss the advantages of using MVC in web development compared to other patterns.**

**Answer:**

**MVC (Model-View-Controller)** is a software architectural pattern used in web and application development.

**Components:**

* **Model:** Manages data and business logic.
* **View:** Handles the user interface (UI).
* **Controller:** Processes input and updates the Model and View.

**Workflow:**

User interacts with the View → Controller processes the request → Model handles data → Updated View is shown to the user.

**Advantages:**

1. **Separation of Concerns:** Keeps UI, logic, and data separate.
2. **Scalability:** Easy to add or update components.
3. **Testability:** Code can be tested independently.
4. **Reusability:** Components can be reused across the application.

**Compared to Other Patterns:**

* More structured than simple monolithic designs.
* Easier to manage and scale compared to linear or unstructured architectures.

**Used in:** ASP.NET MVC, Laravel, Django, etc.

Long Qns

**Q1. Discuss the software development life cycle (SDLC) in detail with examples.**

(10 Marks)

**✅ Introduction:**

The **Software Development Life Cycle (SDLC)** is a structured process used for planning, designing, developing, testing, and deploying software systems. It ensures that the final product meets customer expectations, is delivered on time, and stays within budget.

SDLC provides a clear framework for software development by breaking the process into well-defined stages. It helps reduce project risks, manage resources efficiently, and ensure quality outcomes.

**🌀 Phases of SDLC:**

**1. Requirement Gathering and Analysis:**

* In this phase, business analysts and stakeholders gather detailed requirements from end-users or clients.
* The goal is to clearly understand what the software should do.
* **Output:** Software Requirement Specification (SRS) document.

**Example:**  
A hospital wants a software system to manage patient records. Requirements include patient registration, appointment booking, and report generation.

**2. System Design:**

* Based on the SRS, architects and designers create the system architecture.
* It includes database design, UI design, and system integration design.
* Tools like flowcharts, ER diagrams, and UML diagrams are used.

**Example:**  
Designing the database tables for storing patient information and creating mockups for the appointment form.

**3. Implementation (Coding):**

* Developers write the actual source code in this phase.
* Tasks are divided among team members based on modules.
* Programming languages and frameworks are chosen based on project requirements.

**Example:**  
Using Python and Django to build the hospital management system's backend, and HTML/CSS for the frontend.

**4. Testing:**

* QA testers check the developed software for bugs and verify that it meets the requirements.
* Types of testing: Unit Testing, Integration Testing, System Testing, and User Acceptance Testing (UAT).

**Example:**  
Testing whether the patient registration form correctly stores data in the database and validates inputs.

**5. Deployment:**

* After successful testing, the software is released to the client or made live on a server.
* It may be deployed in phases (pilot launch) or all at once.

**Example:**  
Deploying the hospital system on a secure cloud server, like AWS, for doctors and reception staff to use.

**6. Maintenance:**

* Post-deployment, developers monitor the system for any issues or updates.
* This phase includes bug fixes, performance improvements, and feature upgrades.

**Example:**  
Adding a new feature to send automatic appointment reminders via SMS or email.

**📌 Models of SDLC:**

There are various SDLC models based on how the phases are executed:

1. **Waterfall Model:**  
   Linear and rigid. Each phase must be completed before the next begins.
2. **Agile Model:**  
   Iterative and flexible. Development occurs in cycles (sprints) with frequent client feedback.
3. **Spiral Model:**  
   Combines iterative development with risk analysis. Suitable for complex projects.
4. **V-Model:**  
   Each development phase is linked with a corresponding testing phase.

**⭐ Importance of SDLC:**

* **Improves project planning and estimation.**
* **Enhances communication among stakeholders.**
* **Reduces development costs and time.**
* **Improves software quality and user satisfaction.**

**✅ Conclusion:**

The Software Development Life Cycle (SDLC) is essential for systematic, error-free software development. By following SDLC phases, teams can create high-quality software that meets client requirements and performs reliably. Choosing the right SDLC model based on project size, risk, and timeline is crucial for success.

**Q2. Explain the importance of documentation in software development. Mention the types of documentation used.**

(10 Marks)

**✅ Introduction:**

**Documentation** in software development refers to written materials that explain how a system is built, functions, and should be used. It serves as a reference for developers, testers, stakeholders, and end-users throughout the software’s lifecycle.

Documentation improves understanding, communication, and maintenance of the software. Without proper documentation, even well-developed software can become difficult to manage, update, or use.

**⭐ Importance of Documentation in Software Development:**

**1. Enhances Communication:**

* Ensures all stakeholders (developers, testers, clients, and users) are on the same page.
* Reduces misunderstandings about features, functionality, and design.

**2. Aids in Maintenance and Upgrades:**

* Clear documentation makes it easier for developers to maintain or update the software after delivery.
* Helps new developers understand the codebase quickly.

**3. Ensures Project Continuity:**

* When original developers leave, documentation helps others continue the project without delays or confusion.

**4. Supports Testing and Debugging:**

* Testers can refer to documentation to understand expected behaviors and test cases.
* Developers can identify whether an issue is due to a bug or incorrect usage.

**5. Facilitates User Training and Support:**

* User manuals and help files assist end-users in using the software effectively.
* Reduces dependency on customer support.

**6. Helps in Compliance and Audits:**

* Proper documentation is required in industries like healthcare, banking, and government for legal and compliance purposes.

**📚 Types of Documentation Used in Software Development:**

Documentation is categorized into **two main types**: **Technical Documentation** and **User Documentation**.

**🔧 A. Technical Documentation:**

1. **Requirements Documentation:**
   * Describes the software's functional and non-functional requirements.
   * Example: “The system must support up to 1,000 users simultaneously.”
2. **Architecture/Design Documentation:**
   * Explains system architecture, modules, technologies, and database design.
   * Includes UML diagrams, data flow diagrams, ER diagrams, etc.
3. **Source Code Documentation:**
   * Comments and notes within the code to explain logic, functions, classes, and APIs.
4. **API Documentation:**
   * Explains how other software can interact with the application through its APIs.
   * Example: REST API documentation using Swagger or Postman.
5. **Test Documentation:**
   * Includes test plans, test cases, test data, and test results.
   * Used by QA teams to validate the software.
6. **Deployment Documentation:**
   * Details server requirements, installation steps, and deployment procedures.
7. **Maintenance Documentation:**
   * Contains known issues, fixes, upgrade instructions, and system logs.

**👨‍💻 B. User Documentation:**

1. **User Manuals:**
   * Step-by-step guides to help end-users operate the system.
2. **Help Files and FAQs:**
   * Online help sections embedded in the software or on websites.
3. **Tutorials and Training Materials:**
   * Visual or written guides to train new users.
4. **Release Notes:**
   * Summarize new features, improvements, and bug fixes in each software version.

**📝 Real-World Example:**

In an **online banking system**, documentation includes:

* **Technical:** API documentation for developers, architecture diagrams, and deployment steps.
* **User:** A customer manual explaining how to check balances, transfer money, and view statements.

**✅ Conclusion:**

Documentation is not just a formality—it’s a vital part of successful software development. It ensures clarity, continuity, usability, and quality. Well-documented software is easier to maintain, upgrade, and support, benefiting both the development team and the end users.