|  |  |
| --- | --- |
|  | **Subject Name: Software Engineering**  **BSCS- 5 A**  **Department of Computer Science**  **Bahria University, Lahore Campus** |

**Assignment #1**

Date: Week 3, 8, March 2023

Name: \_AFFAN AHMAD \_\_\_\_\_

Roll No: \_03-134221-003\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation of CLO** | **Question Number** | **Marks** | **Obtained Marks** |
| **CLO1: CLO statement**  *Describe various software engineering jargons and their meanings.* | 1 | 5 |  |
| **Total Marks** | | **5** |  |

**Task 1:**

Set out 100 Software Engineering Jargons and their meanings.

Agile: A software development methodology that emphasizes flexibility, collaboration, and customer feedback throughout the development process.

Scrum: An Agile framework that provides a structured yet flexible way for teams to work together to develop software.

Kanban: A visual management method for software development that emphasizes continuous delivery and optimization of workflow.

Waterfall Model: A traditional linear software development approach where progress flows in a sequential, one-directional manner through various phases.

DevOps: A set of practices that promote collaboration and communication between development (Dev) and operations (Ops) teams to improve efficiency and product quality.

Continuous Integration (CI): The practice of frequently integrating code changes into a shared repository, followed by automated testing to detect and address issues early in the development process.

Continuous Delivery (CD): The extension of continuous integration, ensuring that code changes are automatically prepared for production deployment.

Version Control System (VCS): Software tools that help track and manage changes to source code, allowing collaboration among team members. Examples include Git and SVN.

Repository: A central location where version-controlled files and project data are stored.

Branch: A parallel version of the code, allowing developers to work on separate features or fixes independently.

Merge: Combining changes from different branches into the main codebase.

Code Review: A systematic examination of code by other team members to identify bugs, improve code quality, and ensure compliance with coding standards.

Refactoring: The process of restructuring existing code without changing its external behavior to improve readability, maintainability, and efficiency.

Unit Testing: The practice of testing individual units or components of software to ensure they function as expected.

Integration Testing: Testing the interaction between different components or systems to verify their combined functionality.

Regression Testing: Repeatedly testing a system after changes to ensure that existing functionalities remain unaffected.

Automated Testing: The use of tools and scripts to execute tests automatically, ensuring efficiency and consistency in testing.

User Acceptance Testing (UAT): Final testing conducted to ensure that the system meets end-users' expectations before release.

API (Application Programming Interface): A set of rules and tools for building software applications, specifying how software components should interact.

REST (Representational State Transfer): An architectural style for designing networked applications, often using HTTP for communication.

SOAP (Simple Object Access Protocol): A protocol for exchanging structured information in web services.

Microservices: An architectural approach where an application is composed of small, independent services that communicate through APIs.

Containerization: The encapsulation of an application and its dependencies into a container, ensuring consistency across various environments. Commonly used with Docker.

Docker: A platform for containerization, simplifying the deployment and scaling of applications.

Virtualization: Creating virtual instances of resources, such as servers or operating systems, to run multiple instances on a single physical machine.

Cloud Computing: The delivery of computing services, including servers, storage, databases, networking, software, and analytics, over the internet.

Serverless Computing: A cloud computing model where cloud providers automatically manage the infrastructure, allowing developers to focus on code.

Load Balancing: Distributing incoming network traffic across multiple servers to ensure optimal resource utilization and prevent overload on any single server.

High Availability: The design principle of ensuring a system remains operational and accessible with minimal downtime.

Fault Tolerance: A system's ability to continue functioning even in the presence of faults or errors.

CAP Theorem: A theoretical framework stating that a distributed system cannot simultaneously provide all three guarantees of Consistency, Availability, and Partition Tolerance.

Concurrency: The execution of multiple tasks or processes at the same time.

Parallelism: The simultaneous execution of multiple tasks or processes to improve performance.

Multi-threading: A programming technique where multiple threads within a process execute independently but share the same resources.

Deadlock: A situation where two or more processes are unable to proceed because each is waiting for the other to release a resource.

Race Condition: A condition where the behavior of a program depends on the relative timing of events.

Singleton Pattern: A design pattern where a class has only one instance, providing a global point of access to it.

Factory Pattern: A creational design pattern that provides an interface for creating objects but allows subclasses to alter the type of objects created.

MVC (Model-View-Controller): An architectural pattern that separates an application into three interconnected components: Model, View, and Controller.

Observer Pattern: A behavioral design pattern where an object, known as the subject, maintains a list of its dependents, called observers, that are notified of state changes.

Dependency Injection: A design pattern where a component's dependencies are injected from the outside, promoting modularity and testability.

ORM (Object-Relational Mapping): A technique for converting data between incompatible type systems, often used to map database entries to objects in an object-oriented language.

SQL (Structured Query Language): A domain-specific language for managing and manipulating relational databases.

NoSQL: A term used to describe non-relational databases that provide flexible and scalable data storage solutions.

Big-O Notation: A mathematical notation describing the upper bound of an algorithm's time or space complexity.

Algorithm: A step-by-step procedure or formula for solving a problem or accomplishing a task.

Data Structure: A method of organizing and storing data to facilitate efficient access and modification.

Hashing: A technique for mapping data to a fixed-size array, enabling quick data retrieval.

Encryption: The process of encoding data to make it unreadable without the proper decryption key.

Decryption: The process of converting encrypted data back into its original form.

1. Sprint: In Scrum, a time-boxed iteration during which a specific set of features or tasks is developed and made ready for review.
2. Burn-down Chart: A visual representation that shows the progress of completed work versus remaining work over time during a project.
3. Feature Flag: A configurable setting that allows developers to enable or disable a feature in a live environment without deploying new code.
4. Dark Launching: Introducing a new feature gradually to a subset of users before a full release, allowing for testing and monitoring.
5. Chaos Engineering: The practice of intentionally injecting failures and disturbances into a system to identify weaknesses and improve resilience.
6. Technical Debt: The metaphorical concept of the long-term cost of choosing an easy but suboptimal solution instead of a more optimal approach that would require more effort initially.
7. YAGNI (You Ain't Gonna Need It): A principle of extreme programming (XP) encouraging developers to only implement functionality when it's needed, avoiding unnecessary features.
8. DRY (Don't Repeat Yourself): A software development principle advocating for the avoidance of duplicated code, promoting code reuse and maintainability.
9. KISS (Keep It Simple, Stupid): A design principle urging simplicity in systems and software to enhance understandability and reduce unnecessary complexity.
10. BDD (Behavior-Driven Development): An agile software development technique that encourages collaboration between developers, QA, and non-technical stakeholders in the creation of requirements.
11. CQRS (Command Query Responsibility Segregation): A design pattern that separates the read and write operations of a data store, optimizing for each use case independently.
12. Event Sourcing: A software architecture pattern where changes to an application state are captured as a sequence of events.
13. GraphQL: A query language for APIs and a runtime environment for executing queries with existing data.
14. RESTful API: An API design adhering to the principles of Representational State Transfer (REST), typically using standard HTTP methods.
15. Webhooks: User-defined HTTP callbacks that provide real-time information to other applications when certain events occur.
16. Immutable Infrastructure: A deployment model where infrastructure components are replaced, rather than updated, to ensure consistency and reliability.
17. Blue-Green Deployment: A deployment strategy where two identical environments (blue and green) are maintained, with only one actively serving production traffic at a time.
18. Feature Driven Development (FDD): An iterative and incremental software development methodology that emphasizes building features in short timeframes.
19. Mob Programming: A development practice where the entire team works on the same thing, at the same time, in the same space, and at the same computer.
20. Pair Programming: A development technique where two programmers work together at one workstation, with one writing code and the other reviewing and suggesting improvements.
21. Infrastructure as Code (IaC): Managing and provisioning computing infrastructure through machine-readable script files rather than through physical hardware configuration.
22. Serverless Architecture: An architecture style where applications are built and deployed without managing servers, typically utilizing cloud-based serverless computing platforms.
23. Feature Toggles: A development technique that allows feature flags to be toggled on or off at runtime to enable or disable specific functionalities.
24. Test Driven Development (TDD): A software development approach where tests are written before the code, ensuring that the code meets requirements and functions correctly.
25. Pair Testing: A testing technique where two team members collaborate to test a feature, share ideas, and catch defects.
26. Load Testing: Assessing the behavior of a system under expected or increasing load to ensure it can handle the anticipated number of users.
27. Performance Testing: Evaluating the speed, responsiveness, and stability of a software application under various conditions.
28. A/B Testing: A controlled experiment where two variants (A and B) of a web page or application are compared to determine which performs better.
29. Dogfooding: The practice of using one's own product or service for testing and quality assurance.
30. Inversion of Control (IoC): A design principle where control flow is inverted, often achieved through dependency injection, making the code more modular and flexible.
31. FaaS (Function as a Service): A cloud computing model where individual functions are executed in response to events without the need to manage server infrastructure.
32. WebAssembly: A binary instruction format that enables high-performance execution of code on web browsers.
33. Distributed Systems: Systems composed of multiple independent components that communicate and coordinate with each other.
34. Immutable Object: An object whose state cannot be modified after creation, promoting predictability and avoiding unexpected changes.
35. Progressive Web App (PWA): A type of application software delivered through the web, offering a native app-like experience.
36. Technical Stack: The combination of programming languages, frameworks, libraries, and tools used to build software.
37. Session Management: The process of securely managing user sessions and interactions on a website or application.
38. Tokenization: The process of replacing sensitive data with a unique identifier (token) to enhance security.
39. Biometric Authentication: Security authentication using physical or behavioral characteristics, such as fingerprints or facial recognition.
40. Conway's Law: A theory that organizations design systems that mirror their own communication structure.

Feature Set: A specific set of functionalities or features offered by a software product.

Neural Network: A computational model inspired by the structure and function of the human brain, used in machine learning and artificial intelligence.

Code Smell: A term used to describe patterns in the source code that may indicate potential issues or areas for improvement.

Greenfield Project: A new software project started from scratch, without any existing legacy code or constraints.

Brownfield Project: A software project that involves working with existing code, systems, or infrastructure, often requiring maintenance or upgrades.

Master-Slave Architecture: An architectural pattern where one component or process controls and delegates tasks to other components or processes.

On-Premises: Refers to software or infrastructure that is installed and operated from the organization's premises rather than a remote data center or cloud.

Elasticsearch: An open-source, distributed search and analytics engine often used for full-text search and log data analysis.

Docker Compose: A tool for defining and running multi-container Docker applications, allowing easy configuration and orchestration.

Technical Evangelist: A person who promotes and advocates for a specific technology or product, often within the developer community.

窗体顶端

窗体底端

Good luck!