Sr.No.	Topics	Durations (Mins)	Session No(2 Hour)	Session No.(4 Hour)
1	Overview of Application Development	30	1	1
2	Various Types of Application	40	1	1
3	Introduction To Database	40	1	1
4	Multi-tier Application Architecture	50	2	1
5	Overview of Monolithic and Microservices	50	2	1

Application Development Fundamentals

1. Overview of Application Development Information

- Application development refers to the process of creating software applications to fulfill specific business or user needs.
- It involves designing, coding, testing, and deploying software solutions for various platforms and devices.

1. Requirements Gathering:
   - In this initial phase, the development team works closely with stakeholders to gather and document the requirements for the application.
   - This includes understanding the business objectives, user needs, functional requirements, and any constraints or technical specifications.

2. Design and Architecture:
   - Based on the requirements, the development team creates the application's architecture and designs its structure.
   - This involves designing the user interface, database schema, system components, and defining the overall system flow.
   - The design phase helps in visualizing the application's structure and planning for scalability, maintainability, and security.

3. Development:
    - The development phase involves writing the code to implement the application's features and functionality.
    - Developers use programming languages, frameworks, and development tools based on the project requirements.
    - They follow coding best practices and development standards to ensure code quality, readability, and maintainability.

4. Testing:
    - Testing is a crucial phase to identify and fix bugs, ensure the application functions correctly, and meets the specified requirements.
    - Different testing techniques such as unit testing, integration testing, system testing, and user acceptance testing are employed to validate the application's functionality, performance, usability, and security.

5. Deployment:
    - Once the application has undergone testing and is deemed stable and ready for release, it is deployed to the target environment.
    - This involves setting up servers, databases, configuring the application, and ensuring it is accessible to the intended users.
    - Deployment processes may vary depending on the deployment platform, such as web servers, cloud platforms, or mobile app stores.

6. Maintenance and Support:
   - After the application is deployed, it enters the maintenance phase.
   - This involves addressing any issues reported by users, applying updates, implementing enhancements, and ensuring the application's continued performance, security, and compatibility with evolving technologies.

7. Documentation:
- Create user documentation, including manuals, guides, and tutorials.
- Document the application's code, architecture, and any relevant APIs for future reference.

8. Training and Support:
- Provide training to end-users, administrators, and support staff on using and maintaining the application.
- Offer customer support to address user questions, issues, and feedback.

9. Continuous Improvement:

- Gather feedback from users and stakeholders to identify areas for improvement.
- Plan and implement updates, new features, and enhancements based on user needs and changing market trends.

- Throughout the application development process, collaboration and communication among the development team, stakeholders, and users play a vital role.
- Agile methodologies, such as Scrum or Kanban, are often used to facilitate iterative development, frequent feedback, and adaptability to changing requirements.
- Application development can range from small-scale applications built by individual developers to large-scale enterprise solutions developed by teams of software engineers, designers, and testers.
- It requires a combination of technical skills, problem-solving abilities, creativity, and project management expertise to deliver high-quality applications that meet user needs and business goals.

Different methodology in IT :

- In the field of Information Technology (IT), various methodologies have been developed to guide the processes, practices, and approaches for project management, software development, and system implementation. These methodologies provide structured frameworks to ensure efficient and effective outcomes.
- Here are some of the most well-known methodologies in IT:

1. Waterfall Methodology:

- The Waterfall methodology follows a linear and sequential approach.
- It divides the project into distinct phases, such as requirements gathering, design, development, testing, and deployment.
- Each phase is completed before moving to the next, making it suitable for projects with well-defined requirements.

2. Agile Methodology:

- Agile is a flexible and iterative approach that emphasizes collaboration, adaptability, and customer feedback.
- It breaks the project into smaller iterations called sprints, allowing for continuous development, testing, and delivery.
- Popular Agile frameworks include Scrum, Kanban, and Extreme Programming (XP).

3. Scrum:

- Scrum is an Agile framework that divides work into time-boxed iterations called sprints.
- It involves roles (Scrum Master, Product Owner, Team), ceremonies (Daily Standup, Sprint Planning, Sprint Review, Sprint Retrospective), and artifacts (Product Backlog, Sprint Backlog) to facilitate collaboration and transparency.

4. Kanban:

- Kanban is another Agile framework that visualizes work as it moves through different stages.
- It focuses on optimizing flow and limiting work in progress.
- Teams use a Kanban board to track tasks and move them through columns representing different stages.

5. DevOps:

- DevOps is a cultural and technical approach that aims to bridge the gap between development and operations teams.
- It emphasizes collaboration, automation, continuous integration, continuous delivery, and monitoring to achieve faster and more reliable software delivery.

Processes:

1. Software Development Life Cycle (SDLC):

- The SDLC is a process used to design, develop, test, and deploy software applications.
- It typically includes phases such as requirement gathering, design, coding, testing, deployment, and maintenance.
- Different methodologies like Waterfall, Agile, and DevOps can be followed within the SDLC.

2. Agile Methodology:

- Agile is an iterative and incremental approach to software development.

- It focuses on collaboration, customer feedback, and delivering working software in short cycles (sprints).
- Agile methodologies include Scrum, Kanban, and Extreme Programming (XP).

3. DevOps:

- DevOps is a set of practices that emphasize collaboration between development and IT operations teams.
- It aims to automate and streamline the software delivery process, enabling continuous integration, continuous delivery, and faster deployments.

2. Introduction to Database Information

- A database is a structured collection of data organized and stored in a way that allows efficient storage, retrieval, and manipulation of data.
- Databases are widely used in various applications and systems to store and manage vast amounts of structured and unstructured information.

Key Concepts:

1. Data:

- Data refers to the raw facts, figures, and information that are stored in a database.
- It can be text, numbers, dates, images, videos, or any other form of digital content.

2. Database Management System (DBMS):
- A database management system is software that provides tools and functionalities to create, organize, and manage databases.
- It facilitates the interaction between users or applications and the underlying database.

3. Tables and Rows:
- Data in a database is typically organized in tables.
- Each table consists of rows and columns.
- A row, also known as a record, represents a single instance or entry of data, while a column represents a specific attribute or data field within the table.

4. Schema:
- The schema defines the structure and organization of the database.
- It includes the definition of tables, columns, data types, relationships between tables, and constraints on the data.

5. Primary Key:
- A primary key is a unique identifier for each row in a table.
- It ensures that each row can be uniquely identified and provides a way to establish relationships between tables.

6. Relational Database:

- A relational database is a type of database that organizes data into tables and establishes relationships between tables based on common attributes.
- It uses a structured query language (SQL) for data retrieval and manipulation.

7. Query Language:

- A query language allows users or applications to retrieve, manipulate, and manage data in the database.
- SQL (Structured Query Language) is a widely used query language for relational databases.

3. Overview of Monolithic and Microservices Information

Monolithic and microservices are two different architectural approaches for designing software applications.

A. Monolithic Architecture:

- In a monolithic architecture, the entire application is built as a single, self-contained unit.
- All components, modules, and services are tightly coupled and run within a single process or executable. Here are some key characteristics of monolithic architecture:

1. Single Unit:

- A monolithic application is deployed as a single unit, where all components and functionalities are bundled together.

2. Tight Coupling:

- In a monolith, components and modules have direct dependencies on each other.
- Changes to one part of the application may require modifications in other parts.

3. Shared Codebase:

- A monolithic application typically has a shared codebase, making it easier to develop, test, and maintain the application as a whole.

4. Centralized Deployment:

- Monolithic applications are deployed on a single server or runtime environment, with the entire application running as a single process.

B. Microservices Architecture:

- In a microservices architecture, an application is divided into a set of loosely coupled and independently deployable services.
- Each service represents a specific business capability and can be developed, deployed, and scaled independently.

Here are some key characteristics of microservices architecture:

1. Service-Based:
- In microservices, the application is decomposed into individual services, each responsible for a specific function or capability.

2. Loose Coupling:
- Microservices are loosely coupled, meaning they can operate independently and have minimal dependencies on other services.

3. Individual Deployment:
- Each microservice can be developed and deployed independently, enabling faster development cycles and flexible deployment options.

4. Distributed Architecture:
- Microservices typically run as separate processes or containers and communicate with each other through lightweight protocols like HTTP or message queues.

Comparison:

1. Scalability:

Microservices architecture offers better scalability as each service can be scaled independently based on demand, while in a monolithic architecture, the entire application needs to be scaled.

2. Modularity and Maintainability:

- Microservices allow for independent development, testing, and deployment of individual services, making it easier to maintain and update specific components.
- Monolithic architectures can be more challenging to maintain and evolve due to tight coupling and shared codebase.

3. Flexibility and Technology Diversity:
   - Microservices provide flexibility in adopting different technologies, programming languages, and frameworks for each service.
   - Monolithic architectures typically have a single technology stack throughout the application.

4. Development Complexity:
   - Monolithic architectures are generally simpler to develop and debug due to fewer moving parts.
   - Microservices architecture adds complexity with distributed systems and inter-service communication.

5. Deployment and Release Management:

- Microservices allow for independent deployment and release cycles for each service, enabling faster time-to-market and more efficient updates.
- Monolithic architectures require deploying the entire application as a single unit.

Key Differences:

1. Complexity:

- Monolithic applications are typically simpler to develop and understand due to their single codebase and tight coupling.
- Microservices applications, on the other hand, are more complex due to their distributed nature and inter-service communication.

2. Scalability:

- Microservices architecture offers better scalability compared to monolithic applications.
- Microservices allow for individual service scaling based on specific needs, while scaling a monolithic application usually involves scaling the entire application.

3. Development and Deployment Independence:

- Microservices architecture enables independent development, testing, and deployment of each service.

- Monolithic applications require the whole application to be developed and deployed as a single unit.

4. Technology Diversity:

- Microservices architecture allows for the use of different technologies and programming languages for each service.
- Monolithic applications typically have a single technology stack throughout.

5. Maintenance and Evolution:

- Monolithic applications can be easier to maintain and evolve when making changes that affect multiple components.
- Microservices applications allow for more granular updates and maintenance, with changes limited to individual services.

- The choice between monolithic and microservices architecture depends on factors such as the size and complexity of the application, scalability requirements, team structure, and development processes.
- Monolithic architecture may be suitable for smaller applications or when simplicity is a priority, while microservices architecture is often preferred for larger, complex applications with scalability and independent development needs.

Request and Response Mechanism :

- The request and response mechanism is a fundamental concept in client-server communication, where a client sends a request to a server, and the server processes the request and sends back a response.
- This mechanism allows clients to interact with servers and retrieve the desired information or perform specific actions.