Module - 3 Assignment (Testing on live application)

- **What is RDBMS**

- RDBMS stands for Relational Database Management System.
- It's a software system that's designed to manage relational databases.
- These databases are structured collections of data that are organized in tables, with each table consisting of rows and columns. RDBMS use SQL (Structured Query Language) for managing, manipulating, and querying the data within these databases.
- Some popular examples of RDBMS include MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, and SQLite.
- They are widely used in various applications and industries to store, retrieve, and manage structured data efficiently.

- **What is SQL**

- SQL, or Structured Query Language, is a standard programming language designed for managing relational databases.
- It's used to communicate with and manipulate databases by performing various operations such as querying data, inserting new records, updating existing records, and deleting records.
- SQL provides a standardized way for users to interact with databases, regardless of the specific database management system (DBMS) being used.
- SQL operates through a set of commands, often categorized into groups such as Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL), and Transaction Control Language (TCL).
- SQL is widely used across different industries and is supported by most relational database management systems, including MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, and SQLite, among others.
- Its versatility and standardization make it an essential tool for database administrators, developers, and data analysts.

- **Write SQL Commands**

- Here are some common SQL commands:
  - SELECT: Retrieves data from one or more tables.
  - INSERT: Adds new records into a table.
  - UPDATE: Modifies existing records in a table.
  - DELETE: Removes records from a table.
  - CREATE: Creates new tables, views, or other database objects.
  - ALTER: Modifies the structure of an existing database object.
  - DROP: Deletes tables, views, or other database objects.
  - GRANT: Provides specific privileges to database users.
  - REVOKE: Revokes previously granted privileges.
  - COMMIT: Saves all changes made in the current transaction.
  - ROLLBACK: Reverts the changes made in the current transaction.
  - JOIN: Combines rows from two or more tables based on a related column between them.

- **What is join?**

- In SQL, a JOIN is used to combine rows from two or more tables based on a related column between them.
- It allows you to retrieve data from multiple tables simultaneously by specifying how the tables are related to each other.
- Joins are fundamental for querying data from normalized databases where information is distributed across multiple tables.

- **Write type of joins.**

- There are different types of joins:

- **INNER JOIN:** Retrieves rows from both tables where there is a match between the columns in the specified condition.
  SELECT Orders. Order ID, Customers.Customer Name
  FROM Orders
  INNER JOIN Customers ON Orders.Customer ID = Customers.Customer ID;

- **LEFT JOIN (or LEFT OUTER JOIN):** Retrieves all rows from the left table (the table specified before the JOIN keyword), and the matched rows from the right table. If there is no match, NULL values are returned for the columns from the right table.
  Customers.Customer Name, Orders.Order ID
  FROM Customers SELECT
  LEFT JOIN Orders ON Customers.Customer ID = Orders.Customer ID;

- **RIGHT JOIN (or RIGHT OUTER JOIN):** Retrieves all rows from the right table (the table specified after the JOIN keyword), and the matched rows from the left table. If there is no match, NULL values are returned for the columns from the left table.
  SELECT Orders.Order ID, Customers.Customer Name
  FROM Orders
  RIGHT JOIN Customers ON Orders.Customer ID = Customers.Customer ID;

- **FULL JOIN (or FULL OUTER JOIN):** Retrieves all rows from both tables, matching rows where they exist and returning NULL values for unmatched rows.
  SELECT Customers.Customer Name, Orders.Order ID
  FROM Customers
  FULL JOIN Orders ON Customers.Customer ID = Orders.Customer ID;

- **CROSS JOIN:** Returns the Cartesian product of the two tables, meaning it combines each row from the first table with every row from the second table. Be cautious with this type of join as it can generate large result sets if the tables involved are big.
  SELECT *
  FROM Table1
  CROSS JOIN Table2;

- **How Many constraint and describes it self**

- **T**here are several types of constraints that can be applied to tables to enforce data integrity and business rules. Here are the main types of constraints in SQL and their descriptions:

1. **Primary Key Constraint:** A primary key constraint uniquely identifies each record in a table. It ensures that there are no duplicate values in the specified column(s), and it typically consists of one or more columns. Primary keys are used to enforce entity integrity and are often associated with the table's primary index.

2. **Foreign Key Constraint:** A foreign key constraint establishes a relationship between two tables by enforcing referential integrity. It ensures that values in a column (or columns) of one table match values in a primary key or unique key column of another table. Foreign keys are used to maintain consistency between related tables.

3. **Unique Constraint:** A unique constraint ensures that values in a column (or columns) are unique across all rows in the table. Unlike primary keys, unique constraints allow null values, but they enforce uniqueness for non-null values. Unique constraints are used to enforce business rules that require certain attributes to be unique.

4. **Check Constraint:** A check constraint limits the values that can be inserted or updated in a column based on a Boolean expression. It ensures that data entered into the column meets specific criteria defined by the expression. Check constraints are used to enforce domain integrity and validate data integrity.

5. **Not Null Constraint:** A not null constraint specifies that a column cannot contain null values. It ensures that each row in the table has a valid value for the column, preventing the insertion of incomplete or missing data. Not null constraints are used to enforce data completeness.

6. **Default Constraint:** A default constraint specifies a default value for a column when no value is explicitly provided during an insert operation. If a column with a default constraint is omitted from an insert statement, the default value specified in the constraint is used. Default constraints are used to provide default values for columns.

- These constraints are fundamental to database design and are used to enforce various data integrity rules and business requirements.
- By applying these constraints to tables, database administrators ensure that the data remains consistent, accurate, and reliable throughout its lifecycle.

- **Difference between RDBMS vs DBMS**

Feature	DBMS	RDBMS
Data Representation	Data is represented in a more generic manner, allowing for various data models.	Data is represented in a tabular format with rows and columns, forming tables.
Data Relationships	Does not necessarily enforce relationships between tables.	Enforces relationships between tables using keys (primary keys, foreign keys).
Schema Flexibility	Provides flexibility in defining schemas, allowing for multiple data models.	Follows a structured schema defined by tables, columns, and relationships.
Data Integrity Enforcement	May or may not enforce data integrity constraints.	Enforces data integrity through constraints like primary key, foreign key, etc.
Query Optimization	Query optimization is less sophisticated compared to RDBMS.	Uses advanced optimization techniques to improve query performance.
ACID Compliance	May or may not fully support ACID properties (Atomicity, Consistency, Isolation, Durability).	Fully supports ACID properties to ensure data consistency and reliability.
Scalability	Generally less scalable, especially for large datasets and complex queries.	Designed for scalability, supporting larger datasets and handling complex queries efficiently.
Examples	Examples include SQLite, Microsoft Access, and Berkeley DB.	Examples include MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.

- **What is API Testing**

- API testing, or Application Programming Interface testing, is a type of software testing that focuses on verifying the functionality, reliability, performance, and security of APIS (Application Programming Interfaces).
- APIS serve as the communication channels that allow different software systems or components to interact and exchange data with each other.
- They are commonly used in modern software development for integrating various services, sharing data between applications, and enabling communication between different software modules.
- API testing involves testing the APIS directly, without the need for a user interface.
- API testing is essential for ensuring the quality and robustness of software systems, particularly in the context of Micro services architectures, where multiple services interact through APIS.
- By thoroughly testing APIS, organizations can identify and address issues early in the development life Cycle, improve interoperability between systems, enhance security, and deliver high-quality software products to end-users.
- It typically includes the following activities:

1. **Functional Testing:** Verifying that the API functions as expected by testing its endpoints, request and response formats, HTTP status codes, error handling mechanisms, and data validation rules.

2. **Integration Testing:** Testing the integration between different components or services by sending requests to APIS and verifying that they interact correctly and exchange data accurately.

3. **Security Testing:** Assessing the security vulnerabilities and ensuring the API'S compliance with security standards by testing for authentication mechanisms, authorization controls, input validation, encryption, and protection against common security threats such as SQL injection and cross-site scripting (XSS).

4. **Performance Testing:** Evaluating the performance and scalability of the API by testing its response time, throughput, latency, and concurrency handling capabilities under various load conditions.

5. **Reliability Testing:** Testing the reliability and stability of the API by subjecting it to stress tests, fault tolerance tests, and failure recovery tests to ensure that it can handle failures gracefully and recover without data loss.

6. **Documentation Testing:** Reviewing and testing the API documentation to ensure that it is accurate, comprehensive, and up-to-date, providing developers with the necessary information to use the API effectively.

- **Types of API Testing**

- **Functional Testing:** This type of testing checks whether the API behaves as expected by testing its functionalities. It verifies the inputs, outputs, and the interaction between the API and the application.

- **Unit Testing:** Developers usually perform unit testing to verify the individual units or components of the API. It focuses on testing the smallest parts of the code to ensure they work as intended.

- **Integration Testing:** Integration testing involves testing the interactions between different components or modules of the API. It ensures that these components work together seamlessly.

- **Load Testing:** Load testing evaluates the API'S performance under normal and peak load conditions. It helps identify performance bottlenecks and ensures that the API can handle a large number of requests without crashing or slowing down.

- **Security Testing:** Security testing checks for vulnerabilities and ensures that the API is secure from potential threats such as unauthorized access, SQL injection, cross-site scripting (XSS), etc.

- **Stress Testing:** Stress testing assesses the API'S stability and reliability under extreme conditions. It involves testing the API beyond its normal operational capacity to identify its breaking point.

- **Usability Testing:** Usability testing evaluates how user-friendly the API is. It focuses on aspects such as documentation clarity, ease of use, and error handling.

- **Compatibility Testing:** Compatibility testing ensures that the API works correctly across different platforms, devices, and operating systems.

- **What is Responsive Testing?**

- Responsive testing is a type of testing that ensures a website or web application displays and functions correctly across various devices, screen sizes, resolutions, and orientations.
- The goal of responsive testing is to verify that the user interface adapts appropriately to different devices, providing an optimal viewing and user experience regardless of the device being used.

- Here are some key aspects of responsive testing:

1. **Layout:** Responsive testing checks that the layout of the website or web application adjusts dynamically based on the screen size and orientation. This includes testing for proper placement of elements such as navigation menus, buttons, images, and text blocks.

2. **Viewport:** It verifies that the viewport meta tag is correctly configured to ensure that the content scales properly to fit the screen size of different devices.

3. **Media Queries:** Responsive testing validates that CSS media queries are implemented correctly to apply different styles based on the device's characteristics, such as screen width, height, and resolution.

4. **Fluidity:** It ensures that elements within the webpage resize and reflow smoothly as the screen size changes, without causing overlapping or clipping of content.

5. **Navigation and Interactions:** Responsive testing checks that navigation menus, dropdowns, and interactive elements are accessible and usable across devices, including touch-enabled devices.

6. **Performance:** It evaluates the performance of the website or web application on different devices and network conditions to ensure fast loading times and smooth interactions.

7. **Cross-Browser Compatibility:** Responsive testing also considers cross-browser compatibility to ensure consistent behavior across various web browsers on different devices.

- Responsive testing is crucial in today's multi-device landscape, where users access websites and web applications from a wide range of devices, including smartphones, tablets, laptops, and desktop computers. By performing responsive testing, developers and QA testers can ensure that the user experience remains consistent and satisfactory across all devices, leading to higher user satisfaction and engagement.

- **Which types of tools are available for Responsive Testing**

1. **Browser Developer Tools:** Most modern web browsers come with built-in developer tools that include features for responsive testing. These tools allow developers and testers to simulate different devices, view the webpage at various screen sizes, and debug responsive issues directly within the browser. Examples include Chrome DevTools, Firefox Developer Tools, Safari Web Inspector, and Edge DevTools.

2. **Responsive Design Testing Tools:** These tools provide dedicated environments for testing responsive designs across multiple devices and screen sizes. They often offer predefined device presets, allowing users to quickly switch between different device profiles and view how the webpage renders on each device. Examples include BrowserStack, CrossBrowserTesting, LambdaTest, and Responsinator.

3. **Viewport Resizer Extensions:** These browser extensions allow users to resize the browser window to simulate different screen sizes and resolutions. They are useful for quick and lightweight responsive testing directly within the browser. Examples include Window Resizer for Chrome and Firefox, and Viewport Resizer for Chrome.

4. **Emulators and Simulators:** Emulators and simulators replicate the behavior of various devices and operating systems on a computer. They are useful for testing responsive designs across different platforms without needing access to physical devices. Examples include Android Emulator, iOS Simulator (available with Xcode), Genymotion (Android emulator), and Xamarin Test Cloud.

5. **Automated Testing Tools:** Automated testing tools can be used to automate responsive testing processes, including testing across multiple devices and screen sizes. These tools often provide capabilities for running tests in parallel on different devices and browsers to ensure consistent behavior. Examples include Selenium WebDriver, Appium (for mobile apps), TestComplete, and Cypress

6. **Performance Testing Tools:** While not solely dedicated to responsive testing, performance testing tools can help evaluate how responsive designs perform under different load conditions and network speeds. They provide insights into factors such as page load times, rendering performance, and resource utilization. Examples include Apache JMeter, LoadRunner, WebPageTest, and GTmetrix.

- By leveraging these tools, developers and QA testers can effectively test responsive designs and ensure that websites and web applications deliver a consistent and optimal user experience across various devices and screen sizes.

- **What is the full form of .ipa, .apk**

- ipa stands for "**iOS App Store Package**." It is the file format used for distributing and installing applications on iOS devices such as iPhones and iPads. IPA files contain the binary executable code, resources, and metadata required to install and run an iOS application on a compatible device.
- apk stands for "**Android Package Kit**." It is the file format used for distributing and installing applications on Android devices. APK files contain the compiled code, resources, and manifest file required to install and run an Android application on a compatible device.

- **How to create step for to open the developer option mode ON?**

1. **Unlock your Android device:** If your device is locked, unlock it using your PIN, pattern, password, or biometric authentication method (fingerprint or face unlock).
2. **Open Settings:** Find and tap the "Settings" app on your Android device. This app usually has a gear icon and is commonly found on the home screen or in the app drawer.
3. **Navigate to About Phone:** Scroll down in the Settings menu and find the "About phone" or "About device" option. Tap on it to access additional information about your device.
4. **Locate Build number:** In the "About phone" or "About device" section, scroll down until you find the "Build number" entry. It may be located under "Software information" or a similar heading.

5. **Tap Build number multiple times:** Tap on the "Build number" entry multiple times quickly. You will typically need to tap it seven times in rapid succession. As you tap, you should see a message indicating how many taps are remaining before you unlock the developer options.
6. **Enter your device's PIN, pattern, or password if prompted:** Depending on your device's settings, you may be asked to enter your PIN, pattern, or password to confirm your action.
7. **Developer options are now enabled:** After tapping the "Build number" multiple times and entering your device's security credentials if prompted, you should see sa message confirming that you've enabled Developer options.
8. **Access Developer options:** Return to the main Settings menu, and you should now see "Developer options" listed as one of the available settings. Tap on it to access a wide range of developer-oriented settings and features.

- By following these steps, you can easily enable Developer options on your Android device, allowing you to access advanced settings and features for development and debugging purposes.