**Module 3: Testing on live application**

**Q.1 What is RDBMS?**

* RDBMS stands for Relational Database Management System. It's a type of software used to store, manage, and retrieve data that is organized into tables (also called relations). Each table contains rows (records) and columns (fields), and relationships can be established between different tables using keys.
* **Key Concepts of RDBMS:**
* **Tables**: Data is stored in structured tables.
* **Rows and Columns**: Each row represents a record; each column represents a data field.
* **Primary Key**: Uniquely identifies each row in a table.
* **Foreign Key**: A field in one table that refers to the primary key in another, used to establish relationships.
* **SQL (Structured Query Language)**: The standard language used to interact with RDBMS.
* **Popular RDBMS Examples:**
* **MySQL**
* **PostgreSQL**
* **Oracle Database**
* **Microsoft SQL Server**
* **SQLite**

**Q.2 What is SQL?**

SQL stands for Structured Query Language. It is the standard language used to communicate with relational databases. You use SQL to create, read, update, and delete data — commonly known as CRUD operations.

* **Main Uses of SQL:**

1. **Create** databases and tables
2. **Insert** data into tables
3. **Retrieve** data using queries
4. **Update** existing data
5. **Delete** unwanted data
6. **Control** access and manage user permissions

**Q.3 Write SQL Commands**

### Bottom of Form

### Data Query Language (DQL)

Used to retrieve data from the database.

* **SELECT** – Retrieves data from one or more tables.
* SELECT \* FROM employees;
* SELECT name, salary FROM employees WHERE department = 'HR';

**Data Definition Language (DDL)**

Used to define and manage database structure.

**CREATE** – Creates a new table, database, index, etc.

CREATE TABLE employees (

id INT PRIMARY KEY,

name VARCHAR(50),

department VARCHAR(50),

salary DECIMAL(10, 2)

);

* **ALTER** – Modifies an existing table.

ALTER TABLE employees ADD hire\_date DATE;

* **DROP** – Deletes tables, views, or databases.

DROP TABLE employees;

* **TRUNCATE** – Removes all rows from a table but keeps its structure.

TRUNCATE TABLE employees;

**Data Manipulation Language (DML)**

Used to manage data inside tables.

* **INSERT** – Adds new data.

INSERT INTO employees (id, name, department, salary) VALUES (1, 'Alice', 'HR', 50000);

* **UPDATE** – Modifies existing data.

UPDATE employees SET salary = 55000 WHERE id = 1;

* **DELETE** – Removes data.

DELETE FROM employees WHERE id = 1;

**Data Control Language (DCL)**

Used to control access to data.

* **GRANT** – Gives permissions.

GRANT SELECT, INSERT ON employees TO user\_name;

* **REVOKE** – Removes permissions
* REVOKE INSERT ON employees FROM user\_name;

**Transaction Control Language (TCL)**

**Manages changes made by DML statements.**

* **COMMIT** – Saves the changes.
* COMMIT;
* **ROLLBACK** – Undoes changes since the last COMMIT.

ROLLBACK;

* **SAVEPOINT** – Sets a point to roll back to.

SAVEPOINT save1;

* **SET TRANSACTION** – Sets properties for the transaction.

SET TRANSACTION READ ONLY;

**Q.4 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's super useful when data is spread across multiple tables and you want to bring it together in a single result.

**Types of JOINS**

1. **INNER JOIN**

Returns **only matching rows** from both tables.

SELECT employees.name, departments.dept\_name

FROM employees

INNER JOIN departments ON employees.dept\_id = departments.id;

1. **LEFT JOIN (or LEFT OUTER JOIN)**

Returns **all rows from the left table**, and matching rows from the right table. If no match, returns NULL.

SELECT employees.name, departments.dept\_name

FROM employees

LEFT JOIN departments ON employees.dept\_id = departments.id;

1. **RIGHT JOIN (or RIGHT OUTER JOIN)**

Returns **all rows from the right table**, and matching rows from the left table. If no match, returns NULL.

SELECT employees.name, departments.dept\_name

FROM employees

RIGHT JOIN departments ON employees.dept\_id = departments.id;

1. **FULL JOIN (or FULL OUTER JOIN)**

Returns **all rows from both tables**, matching when possible, else return NULL.

SELECT employees.name, departments.dept\_name

FROM employees

FULL OUTER JOIN departments ON employees.dept\_id = departments.id;

1. **CROSS JOIN**

Returns the **Cartesian product** – every row from the first table joined with every row from the second table.

SELECT e.name, d.dept\_name

FROM employees e

CROSS JOIN departments d;

**Example Tables:**

**employees**

| **id** | **name** | **dept\_id** |
| --- | --- | --- |
| 1 | Amruta | 101 |
| 2 | Neha | 102 |

**departments**

| **id** | **dept\_name** |
| --- | --- |
| 101 | HR |
| 102 | IT |

Using an **INNER JOIN**, you can get:

SELECT employees.name, departments.dept\_name

FROM employees

INNER JOIN departments ON employees.dept\_id = departments.id;

**Result:**

| **name** | **dept\_name** |
| --- | --- |
| Amruta | HR |
| Neha | IT |
|  |  |

**Q.5 How many constraint and describes it self?**

In SQL, constraints are rules applied to table columns to enforce data integrity. There are 6 main types of constraints. Here's a simple explanation for each:

1. NOT NULL

Ensures that a column cannot have NULL values.

CREATE TABLE students (

id INT NOT NULL,

name VARCHAR(50) NOT NULL

);

2. UNIQUE

Ensures that all values in a column are different (no duplicates).

CREATE TABLE users (

email VARCHAR(100) UNIQUE

);

3. PRIMARY KEY

➡ Uniquely identifies each row in a table. It is a combination of NOT NULL and UNIQUE.

sql

CopyEdit

CREATE TABLE employees (

id INT PRIMARY KEY,

name VARCHAR(50)

);

Every table should ideally have a primary key.

4. FOREIGN KEY

Establishes a relationship between two tables. It refers to the primary key in another table.

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

FOREIGN KEY (customer\_id) REFERENCES customers(id)

);

It helps maintain referential integrity between tables.

5. CHECK

Ensures that values in a column meet a specific condition.

CREATE TABLE products (

id INT,

price DECIMAL(10,2),

CHECK (price > 0)

);

Good for rules like “age must be over 18”, or “salary must be positive”.

6. DEFAULT

Sets a default value for a column if none is provided.

CREATE TABLE accounts (

id INT,

status VARCHAR(10) DEFAULT 'active'

);

Summary Table:

| Constraint | Description |
| --- | --- |
| NOT NULL | Value must be provided |
| UNIQUE | All values must be different |
| PRIMARY KEY | Uniquely identifies rows |
| FOREIGN KEY | Creates relationship with another table |
| CHECK | Validates values against a condition |
| DEFAULT | Sets a default value if none is given |

**Q.7 Difference between RDBMS and DBMS?**

**DBMS (Database Management System)**

A software to **store and manage data** in a structured format.

**RDBMS (Relational Database Management System)**

A **type of DBMS** that stores data in the form of **related tables (rows and columns)**.

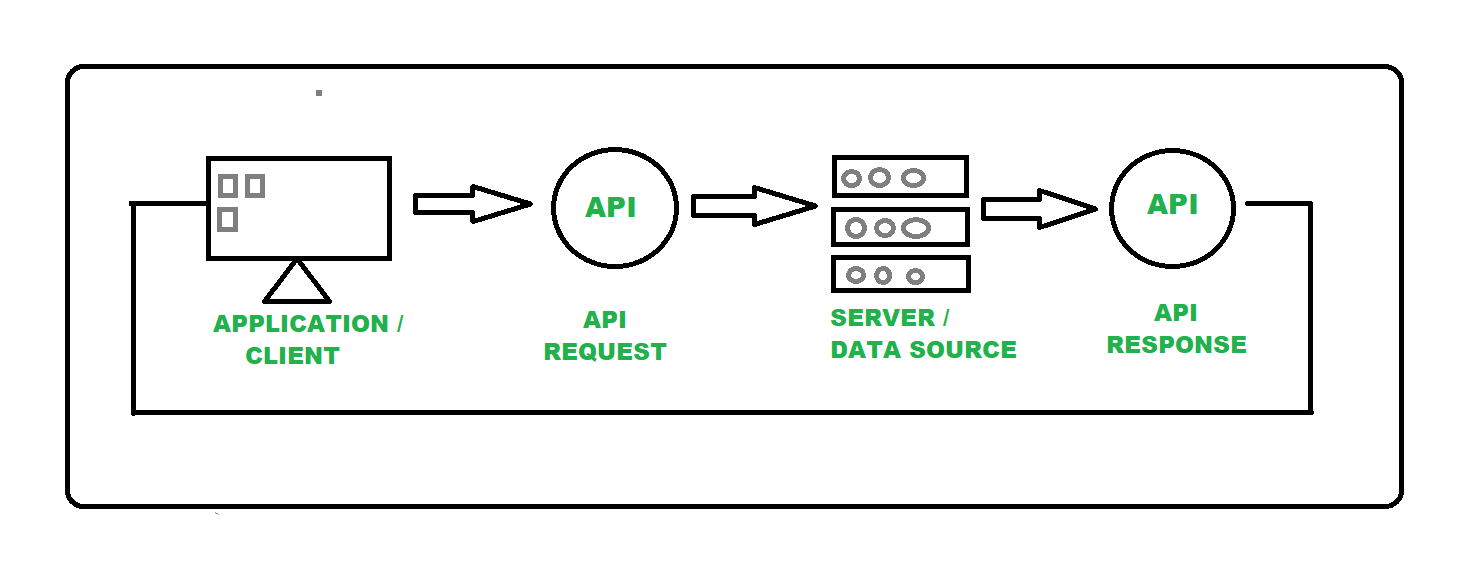
**Key Differences:**

| **Feature** | **DBMS** | **RDBMS** |
| --- | --- | --- |
| **Data Storage** | Data is stored as files or hierarchical form | Data is stored in **tables (rows/columns)** |
| **Relation Support** | Does **not support** relationships | **Supports** relationships using **foreign keys** |
| **Normalization** | Not supported | Fully supports **normalization** |
| **Data Integrity** | Not very strong | Enforces strong **data integrity** rules |
| **Multiple Users** | Usually supports **single user** only | Supports **multiple users** |
| **Examples** | Microsoft Access, File System | MySQL, Oracle, PostgreSQL, SQL Server |
| **Security** | Basic level | Advanced security features |
| **Transactions** | Limited or no support | Supports **transactions** (COMMIT, ROLLBACK) |

**Q.8 What is API Testing?**

**API Testing** (Application Programming Interface Testing) is a type of software testing where **you test the backend (the logic & data)** of an application — **without using the UI (User Interface)**.

APIs allow different software systems to talk to each other. In API testing, you check if that communication is working correctly.



**Fig: Types of API Testing**

**Example:**

Imagine a **weather app**. When you click "Get Weather", the app sends a request to a weather API, and the API sends back the temperature and forecast.  
 In API testing, you check:

* Did the app send the correct request?
* Did the API return the correct weather info?
* Did it return it fast?
* Did it handle errors properly

**Key Things Tested in API Testing:**

| **What You Test** | **Example** |
| --- | --- |
| **Request & Response** | Is the API returning the correct data? |
| **Status Codes** | 200 OK, 404 Not Found, 500 Server Error, etc. |
| **Data Format** | JSON or XML – is it correct and complete? |
| **Performance** | Is the API fast and efficient? |
| **Security** | Is unauthorized access blocked? |
| **Error Handling** | Does it return proper error messages? |

**Tools Used for API Testing:**

* **Postman** – Most popular and user-friendly for manual testing
* **SoapUI** – For SOAP and REST APIs
* **Rest Assured** – Java library for automated API testing
* **JMeter** – For performance testing
* **Swagger** – For API documentation and testing

**Basic Example using Postman:**

If you're testing a **login API**, you might send a POST request like this:

{

"username": "amruta123",

"password": "password123"

}

You check:

* If the response status is 200 OK
* If it returns a token or message like "Login successful"
* If it gives an error when wrong credentials are sent

**Q.9 Types of API Testing?**

API testing involves evaluating the functionality, performance, and security of an API. There are several different types of API testing, each focused on specific aspects of the API's behavior and performance. Here are the most common types:

**1. Functional Testing**

* **Purpose:** Validates that the API functions as expected and meets the requirements.
* **Focus:** Ensures the API correctly performs its intended actions, such as sending requests and receiving proper responses. It checks for correct data return, error codes, and business logic.

**2. Performance Testing**

* **Purpose:** Evaluates how the API performs under varying loads.
* **Focus:** Measures response times, latency, and throughput. Common types of performance testing include:
  + **Load Testing:** Testing the API under normal and peak loads.
  + **Stress Testing:** Testing the API under extreme conditions to determine its breaking point.
  + **Spike Testing:** Testing how the API behaves under sudden, sharp increases in load.
  + **Scalability Testing:** Testing the API’s ability to scale with increasing load.

**3. Security Testing**

* **Purpose:** Identifies vulnerabilities in the API.
* **Focus:** Ensures that the API is secure against attacks and data breaches. Common security testing practices include:
  + **Authentication Testing:** Verifying that only authorized users can access the API.
  + **Authorization Testing:** Ensuring users have proper access rights.
  + **Data Encryption Testing:** Ensuring data transmitted is encrypted and protected.
  + **Input Validation Testing:** Ensuring that the API does not accept malicious input (e.g., SQL injections, XSS attacks).

**4. Regression Testing**

* **Purpose:** Ensures that new changes (e.g., new features, bug fixes) don’t break existing functionality.
* **Focus:** Tests the API's previously functioning features to make sure they still work after updates or changes to the system.

**5. Load Testing**

* **Purpose:** Determines how the API handles high volumes of traffic.
* **Focus:** Simulates a large number of API calls to assess its behavior under load and to ensure it can handle the required amount of concurrent requests without crashing or slowing down significantly.

**6. Boundary Testing**

* **Purpose:** Checks how the API behaves when it reaches its boundaries or limits (e.g., the maximum data size or the maximum number of requests).
* **Focus:** Testing edge cases to ensure that the API behaves as expected when it reaches its limits.

**7. Compatibility Testing**

* **Purpose:** Ensures the API functions correctly across various platforms, environments, or systems.
* **Focus:** Verifies that the API works across different operating systems, browsers, and devices (especially for web-based APIs). Also checks for compatibility with different versions of databases or external systems it might interact with.

**8. Error Handling Testing**

* **Purpose:** Ensures the API responds properly to errors.
* **Focus:** Tests how the API handles invalid inputs, unexpected situations, and failure conditions. The focus is on whether the API returns the correct error messages, error codes, and status codes.

**9. Documentation Testing**

* **Purpose:** Verifies that the API documentation is accurate and aligns with the actual functionality of the API.
* **Focus:** Ensures that all endpoints, parameters, and response types described in the documentation are correct and that the API behaves as documented.

**10. Integration Testing**

* **Purpose:** Checks how the API interacts with other systems, services, or components.
* **Focus:** Ensures that the API correctly interacts with external dependencies, such as databases, third-party APIs, or microservices. This is crucial to verify that the entire ecosystem works smoothly.

**11. Uptime and Monitoring Testing**

* **Purpose:** Monitors the availability and uptime of the API over time.
* **Focus:** Ensures that the API is always available and responds within acceptable time frames.

Q.10 What is Responsive Testing?

**Responsive Testing** refers to testing the responsiveness of a website or web application, ensuring that it functions properly and provides an optimal user experience across various devices, screen sizes, and resolutions. This type of testing is crucial in today’s multi-device world, where users access websites from desktop computers, laptops, tablets, smartphones, and even smart TVs.

The goal of **Responsive Testing** is to verify that the web application adjusts its layout, content, and functionality according to the device it is being viewed on, without breaking the user experience.

**Key Aspects of Responsive Testing:**

1. **Screen Size Adaptability:**
   * Ensures that the website or application adapts to different screen sizes and orientations (e.g., portrait or landscape mode).
   * For example, checking how the layout looks on devices with small screens (smartphones) and large screens (desktops, laptops).
2. **Fluid Grid Layouts:**
   * Ensures that the grid system (used for layout structure) of the website adjusts fluidly to different screen sizes, so that elements like images, text, and buttons resize appropriately without overlapping or looking out of place.
3. **Font Size and Spacing:**
   * Ensures that font sizes, line spacing, and overall typography are readable and appropriately scaled across devices.
   * Testing is necessary to make sure text is neither too small nor too large for different devices.
4. **Touchscreen Compatibility:**
   * Verifies that interactive elements (buttons, links, menus) are easy to navigate on touchscreen devices, such as tablets and smartphones.
   * Ensures that touch targets are appropriately sized and spaced to prevent accidental taps.
5. **Viewport Testing:**
   * Ensures that the content fits within the viewport of the device and that users don't have to scroll horizontally or zoom in to view the content properly.
   * Verifies that all elements are visible without requiring extra effort from the user.
6. **Testing on Various Devices and Browsers:**
   * It’s essential to test the application on multiple physical devices or emulators to ensure that the responsiveness holds across a wide range of devices (iPhone, Android, tablets, etc.).
   * It should also be checked across different browsers (Chrome, Firefox, Safari, Edge) and operating systems (iOS, Android, Windows, macOS) to ensure consistent behavior.
7. **Media Queries and Breakpoints:**
   * Verifies that the CSS media queries and breakpoints work as intended and cause the layout to adapt at appropriate screen widths (e.g., for tablet, mobile, desktop views).
8. **Content Alignment and Visibility:**
   * Ensures that content such as images, videos, forms, and tables resize and align correctly across different devices.
   * Also tests whether any content is hidden, cropped, or misaligned when switching between devices or screen sizes.
9. **Navigation Testing:**
   * Checks that the navigation menus, such as dropdowns, hamburger menus (on mobile), and sidebars, work well on different devices.
   * Ensures that menus are easy to use and don't interfere with other content on smaller screens.
10. **Cross-Device Functionality:**

* Ensures that interactive elements such as buttons, sliders, forms, and links function properly across devices. It also ensures that features like scrolling, zooming, and swiping work as expected.

**Types of Devices/Platforms Used for Responsive Testing:**

* **Desktop and Laptop Computers** (Windows, macOS, Linux)
* **Mobile Devices** (Smartphones, both Android and iOS)
* **Tablets**
* **Smart TVs**
* **Various Browsers** (Chrome, Safari, Firefox, etc.)

**Tools for Responsive Testing:**

* **Browser Developer Tools:** Most modern browsers have built-in tools to simulate different screen sizes and devices, such as the "Responsive Design Mode" in Firefox and Chrome.
* **BrowserStack or Sauce Labs:** Cloud-based platforms that allow testing on real devices.
* **Selenium:** Can be used to automate and test the responsiveness of a website.
* **Screenfly, Responsinator:** Websites that show how a page appears on different devices.

**Why is Responsive Testing Important?**

* **User Experience:** A poor responsive design leads to a frustrating user experience, where content is hard to read, and navigation becomes difficult.
* **SEO:** Search engines, particularly Google, favor mobile-friendly websites. Websites with poor responsiveness may suffer in search rankings.
* **Increased Mobile Usage:** With a growing number of users browsing from mobile devices, ensuring a seamless experience is crucial for reaching a wide audience.
* **Brand Perception:** A well-designed, responsive website helps enhance the brand's credibility, while a poorly designed one can damage its reputation.

**Q.11 which types of tools are available for responsive testing?**

There are several tools available for **responsive testing**, helping developers and testers ensure that websites and web applications adapt correctly across various devices, screen sizes, and browsers. These tools vary in terms of features, such as device emulation, testing on real devices, and automating responsiveness checks. Here are some popular tools for responsive testing:

**1. Browser Developer Tools**

* **Chrome DevTools**: Built into the Chrome browser, DevTools has a "Device Mode" that allows you to simulate various screen sizes and resolutions, enabling responsive testing. You can also test the mobile and tablet versions of your site, and emulate different devices.
* **Firefox Developer Tools**: Similar to Chrome DevTools, Firefox provides a "Responsive Design Mode" for simulating screen sizes of various devices.
* **Edge DevTools**: The Microsoft Edge browser also includes responsive design tools to test how websites look on different screen sizes and resolutions.
* **Safari Developer Tools**: Safari offers "Responsive Design Mode" that lets you test the design for multiple devices and resolutions directly within the browser.

**2. BrowserStack**

* **What It Does:** BrowserStack is a cloud-based testing platform that allows you to test websites on real mobile devices, tablets, and desktops. It supports multiple browsers and operating systems, and you can test the responsiveness of your website across a wide variety of devices.
* **Features:** Real device testing, cross-browser testing, integration with CI/CD pipelines, live testing, and automated testing.
* **Pros:** Offers real devices for testing, various browsers and OS combinations, and supports both manual and automated testing.

**3. Sauce Labs**

* **What It Does:** Sauce Labs is another cloud-based platform for cross-browser and mobile testing. It allows you to test responsive design on real devices and simulators/emulators in different environments (browsers, OS, and devices).
* **Features:** Cross-browser testing, real device testing, automated tests with Selenium, Appium, etc.
* **Pros:** Supports both manual and automated testing, offers mobile-specific features, and integrates with CI/CD tools.

**4. Responsive Design Checker**

* **What It Does:** This is a simple online tool to check how a website appears on various devices and screen sizes. It offers emulation of multiple devices such as smartphones, tablets, and desktops.
* **Features:** Provides multiple device configurations, simulates different screen resolutions, and quickly checks responsiveness without needing a real device.
* **Pros:** Free to use, no installation needed, quick testing.

**5. Screenfly by QuirkTools**

* **What It Does:** Screenfly is a free tool that allows you to test your website on different screen sizes and resolutions. It can simulate various devices like desktops, tablets, and smartphones.
* **Features:** Test on multiple device types, supports custom screen resolutions, and provides options for screen orientations.
* **Pros:** Simple to use, free, and no installation required.

**6. Responsinator**

* **What It Does:** Responsinator is another online tool that allows you to check how your website looks on different devices (smartphones and tablets). It provides quick previews on popular mobile devices.
* **Features:** Displays how a site appears on multiple popular devices and screen sizes.
* **Pros:** Easy-to-use interface, fast, and focuses on popular devices.

**7. LambdaTest**

* **What It Does:** LambdaTest is a cloud-based testing tool that allows users to perform cross-browser testing and responsive testing on real devices and emulators.
* **Features:** Real-time browser and device testing, supports desktop and mobile views, automated Selenium testing, integrations with Jenkins and other CI tools.
* **Pros:** Provides access to a wide range of devices, operating systems, and browsers, both manual and automated testing.

**8. Am I Responsive?**

* **What It Does:** A quick and simple tool that lets you check how your website will look on different devices. It gives a snapshot of how your page appears on desktop, tablet, and mobile devices.
* **Features:** Simple, easy-to-use tool that shows how a website will look on different screen sizes with the device's frames.
* **Pros:** Free, fast, and easy to use.

**9. Pinegrow**

* **What It Does:** Pinegrow is a desktop app that allows you to build and test responsive websites. It offers a visual editor and the ability to check how a website appears across various screen sizes.
* **Features:** Responsive design editor, multi-device previews, integration with Bootstrap, and automatic CSS generation.
* **Pros:** Visual interface, ability to build and test responsive websites simultaneously, local editing of code.

**10. TestComplete**

* **What It Does:** TestComplete is an automated testing tool that can be used for testing the responsiveness of web applications across different devices and browsers.
* **Features:** Automated cross-browser testing, responsive web testing, integration with Selenium, and supports both desktop and mobile testing.
* **Pros:** Powerful automation capabilities, integrates with CI/CD tools, tests across browsers, and devices.

**11. Emulator/Simulator Tools**

* **Xcode (iOS Simulator)**: For macOS users, Xcode provides a simulator to test how websites will appear on iPhones, iPads, and other Apple devices.
* **Android Studio (Android Emulator)**: For Android devices, Android Studio offers an emulator to test how websites will look on various Android phones and tablets.

**12. WebPageTest**

* **What It Does:** Although WebPageTest primarily focuses on performance testing, it also offers tools to check how a website looks on different screen sizes and devices, as well as detailed reports on responsiveness.
* **Features:** Device simulation, screen size checks, performance analysis, and waterfall charts.
* **Pros:** Free, detailed reports, performance and responsiveness insights.

**13. Real Device Cloud Testing Platforms**

* **What It Does:** Platforms like **Real Device Cloud** or **Device Farms** allow users to test websites on real mobile devices to ensure a more accurate representation of user experiences.
* **Features:** Provides access to hundreds of real devices for testing.
* **Pros:** Accurate results, more reliable than simulators or emulators.

**Conclusion:**

Responsive testing tools vary in features, ranging from simple emulation tools (like Screenfly and Responsinator) to more advanced platforms (like BrowserStack and LambdaTest) that provide access to real devices and cross-browser testing capabilities. The choice of tool depends on the level of detail and automation required for testing, as well as the need for device-specific testing. For most comprehensive testing, it’s often recommended to use a mix of both emulation tools and real-device testing platforms.

**Q.12 What is the full form of .ipa,.apk ?**

The full forms of the file extensions are:

* .ipa: iOS App Store Package. This is the file format used for applications on Apple's iOS operating system. It is used for installing apps on iPhones, iPads, and iPods.
* .apk: Android Package. This is the file format used for installing applications on Android devices. It contains all of an app's code, resources, assets, certificates, and manifest file.

Both file types are used to distribute and install apps on their respective platforms.

**Q.13 How to create step for to open the developer option mode ON?**

To enable **Developer Options** on an Android device, follow these steps:

1. **Open Settings**:
   * Go to the **Settings** app on your Android phone.
2. **Find About Phone**:
   * Scroll down and tap on **About phone** (it may vary slightly based on the device).
3. **Tap on Build Number**:
   * Scroll down in the "About phone" section and locate the **Build number**.
   * Tap on the **Build number** **7 times** quickly.
   * You may be prompted to enter your device's PIN or password to confirm.
4. **Developer Options Unlocked**:
   * After tapping the **Build number** 7 times, you will see a message that says, "**You are now a developer!**"
5. **Access Developer Options**:
   * Go back to the **Settings** menu.
   * Now, you should see a new option called **Developer options** (usually located under **System** or **Additional settings** depending on the device).
6. **Turn On Developer Mode**:
   * Tap on **Developer options**.
   * Toggle the switch at the top of the page to enable it.
   * You can now access various developer settings, like USB debugging, animations, etc.

**Bottom of Form**