QA AUTOMATION SYLLABUS 2025

CORE JAVA SYLLABUS

1. Java Basics

- What is Java? Features of Java
- JVM, JRE, JDK
- Data Types and Variables
- Type Casting
- Operators (Arithmetic, Relational, Logical, Bitwise, Assignment, Ternary)
- Input from user (Scanner class)

2. Control Flow Statements

- Conditional Statements (if, if-else, switch)
- Looping Statements (for, while, do-while)
- break, continue, return statements

3. Arrays and Strings

- 1D and 2D Arrays
- Array methods
- String and StringBuffer, StringBuilder
- Important String methods (length(), charAt(), substring(), split(), etc.)

4. Methods (Functions)

- Method declaration and calling
- Method with arguments and return type
- Method Overloading
- main() method

5. Object-Oriented Programming (OOP)

- Class and Object
- Constructor (Default, Parameterized)
- this keyword
- Inheritance
- Method Overriding
- Polymorphism (Compile-time and Runtime)
- Encapsulation
- Abstraction

- super keyword
- Static and non-static members
- final keyword

6. Access Modifiers

- public
- private
- protected
- default (no modifier)

7. Wrapper Classes

- Integer, Float, Double, Character, etc.
- Autoboxing and Unboxing

8. Exception Handling

- try, catch, finally
- throw, throws
- Checked and Unchecked Exceptions
- Custom Exception

9. Java Collections Framework (VERY IMPORTANT)

- List (ArrayList, LinkedList)
- Set (HashSet, LinkedHashSet, TreeSet)
- Map (HashMap, LinkedHashMap, TreeMap)
- Queue (PriorityQueue)
- Iterator and ListIterator
- Differences between ArrayList vs LinkedList, HashSet vs TreeSet, HashMap vs Hashtable

10. Java 8 Features (Frequently asked in QA roles)

- Lambda Expressions
- Functional Interface
- Streams API
- forEach()
- Method Reference

11. File Handling

- Reading and Writing files using File, FileReader, FileWriter, BufferedReader, BufferedWriter
- Reading properties files (used in frameworks)

12. Multithreading (Only Basics Needed)

- Thread class and Runnable interface
- Thread lifecycle
- sleep(), join(), wait(), notify()

13. Java Keywords

- static, final, this, super, return, new, instance of, etc.

14 Java Memory Management (Overview)

- Stack and Heap
- Garbage Collection

15. Object Class and Important Methods

- equals()
- hashCode()
- toString()
- clone()

SELENIUM SYLLABUS

1. Introduction to Selenium

- What is Selenium?
- Components of Selenium (IDE, RC, WebDriver, Grid)
- Advantages and Limitations
- Why WebDriver is used today
- Architecture of Selenium WebDriver
- How Selenium interacts with browsers

2. Setting Up Selenium WebDriver

- Downloading and installing Selenium
- Configuring drivers (ChromeDriver, GeckoDriver, EdgeDriver)
- Launching browsers using Selenium

3. Browser Navigation Commands

- get() vs navigate().to()
- navigate().forward()
- navigate().back()
- navigate().refresh()
- close() vs quit()

4. Locators in Selenium

- ID
- Name
- Class Name
- Tag Name
- LinkText and PartialLinkText
- XPath (Absolute & Relative)
- CSS Selector
- When to use which locator

5. Working with Web Elements

- click()
- sendKeys()
- clear()
- getText()
- getAttribute()
- isDisplayed()
- isEnabled()
- isSelected()

6. Dropdowns and Select Class

- Selecting by visible text, index, value
- Handling multi-select dropdowns
- Get all selected options

7. Checkboxes and Radio Buttons

- Selecting & deselecting
- Checking whether it's selected

8. Alerts and Pop-ups

- Handling JavaScript alerts (accept, dismiss, getText)
- Confirmation pop-ups
- Prompt pop-ups (sendKeys to alert)

9. Frames and iFrames

- Switching to frame by index, name, WebElement
- Switching back to main content
- Nested frames handling

10. Windows and Tabs Handling

- getWindowHandle()
- getWindowHandles()
- Switching between multiple windows/tabs
- Closing specific windows

11. Mouse and Keyboard Actions (Actions Class)

- Mouse Hover
- Double Click
- Right Click (Context Click)
- Drag and Drop
- Move to Element
- Click and Hold

12. Keyboard Events (Using Keys class)

- Pressing keys (Enter, Tab, Shift, etc.)
- Using sendKeys(Keys.ENTER)

13. Waits in Selenium

- Implicit Wait
- Explicit Wait (WebDriverWait)
- Fluent Wait
- Importance of waits in synchronization

14. JavaScriptExecutor

- Click an element using JS
- Scroll down/up to element

- Highlight element
- Fetch hidden text

15. Web Tables Handling

- Static tables
- Dynamic tables
- Read specific row/column data
- Loop through rows and columns

16. Screenshots

- Taking screenshot of full page
- Taking screenshot of WebElement

17. File Uploads and Downloads

- Uploading file using sendKeys(path)
- AutoIT/Robot Class basics for upload pop-up (just mention)
- Download validations using download folder path (basic idea)

18. Page Scrolling Techniques

- Scroll using JS
- Scroll to bottom/top
- Scroll until element is visible

19. Selenium Grid (Basic Idea)

- What is Selenium Grid?
- Remote execution basics
- Hub & Node concept (just concept, no setup)

20. Best Practices (Without Frameworks)

- Writing reusable browser setup code
- Reusing locators
- Using try-catch blocks
- Keeping code readable and modular

QA AUTOMATION SYLLABUS 2025

CUCUMBER FRAMEWORK SYLLABUS

1. Introduction to TestNG

- What is TestNG?
- Why do we use TestNG in Selenium?
- Difference between JUnit and TestNG
- Installing TestNG in Eclipse/IntelliJ

2. Creating Your First TestNG Test

- Writing a basic TestNG test class
- Using @Test annotation
- Running TestNG class
- Understanding TestNG output

3. TestNG Annotations (Core Part)

- @Test
- @BeforeMethod and @AfterMethod
- @BeforeClass and @AfterClass
- @BeforeTest and @AfterTest
- @BeforeSuite and @AfterSuite
- Execution order of annotations

4. Grouping Test Cases

- What is grouping in TestNG?
- How to create groups using groups attribute
- Executing selected groups from XML
- Include/exclude groups

5. Prioritization and Dependency

- priority attribute
- dependsOnMethods
- enabled=false
- invocationCount
- timeOut attribute

6. TestNG Assertions

- assertEquals()
- assertTrue() and assertFalse()
- assertNotEquals()
- assertNull() and assertNotNull()
- Soft Assertions vs Hard Assertions
- SoftAssert class

7. TestNG XML Configuration File

- Creating testng.xml
- Running tests through XML
- Including/excluding classes, methods
- Running multiple test classes
- Suite and test tags
- Parameters in XML

8. Parameterization

- @Parameters from XML
- @DataProvider annotation
- Difference between @Parameters vs @DataProvider
- Running same test with multiple sets of data

9. Parallel Execution

- Running tests in parallel using XML
- parallel=tests, parallel=methods, thread-count
- Benefits of parallel execution

10. TestNG Listeners (Basic Level)

- What are Listeners?
- Implementing ITestListener
- onTestStart(), onTestSuccess(), onTestFailure()
- Using listeners for logging or screenshots

QA AUTOMATION SYLLABUS 2025

11. TestNG Reports

- Default reports provided by TestNG
- How to read and analyze reports
- Overview of test output folder

12. Retry Failed Tests

- Why we retry tests
- Creating Retry Analyzer
- Attaching retry logic to test methods

13. Suites and Test Tags

- Difference between suite and test in XML
- Organizing suites for smoke, regression, sanity
- Running multiple suites

14. Real-Time Use in QA Projects

- Structure of test classes using TestNG
- Using annotations for setup/teardown
- Combining DataProvider with Selenium
- Executing full regression using testng.xml

TESTNG SYLLABUS

1. Introduction to Cucumber

- What is Cucumber?
- Why use Cucumber in testing?
- Cucumber vs Selenium
- Cucumber in real-time projects
- Advantages of Cucumber for QA

2. Understanding BDD (Behavior Driven Development)

- What is BDD?
- BDD vs TDD
- Why BDD is preferred in automation?
- Real-world example using BDD style

3. Gherkin Language Basics

- What is Gherkin?
- Writing feature files in Gherkin
- Gherkin keywords:
    - Feature:
    - Scenario:
    - Scenario Outline:
    - Given, When, Then, And, But
- Background keyword usage
- Tags (@Smoke, @Regression, etc.)

4. Feature File

- What is a feature file?
- How to structure a feature file
- Naming conventions
- Writing multiple scenarios in one feature file
- Scenario vs Scenario Outline with Examples table

5. Step Definitions

- What are step definition files?
- Mapping Gherkin steps to Java methods
- Using regular expressions in steps
- @Given, @When, @Then annotations
- Reusing step definitions across scenarios

6. Glue Code

- What is glue in Cucumber?
- glue = {"stepDefinitions"} explanation
- Folder structure for glue code

7. Data-Driven Testing in Cucumber

- Using Scenario Outline and Examples
- Using Data Tables in feature files
- Handling key-value pairs in step definitions
- Using Lists and Maps in Java for test data

8. Cucumber Hooks

- What are hooks in Cucumber?
- @Before and @After hooks
- Order of hooks execution
- @BeforeStep and @AfterStep (optional)
- Using hooks for setup and cleanup

9. Tags in Cucumber

- Adding tags to scenarios or features
- Running specific tagged tests
- Tag expressions: @Smoke and @Regression, not @Sanity

10. Cucumber Runner Class

- Creating a runner class using @CucumberOptions
- Parameters in @CucumberOptions:
    - features
    - glue
    - tags
    - dryRun
    - monochrome
    - plugin
- Running feature files using the runner

11. Cucumber Reports

- Generating reports using:
    - pretty
    - html:target/htmlreport
    - json:target/jsonreport
- Reading and analyzing reports

12. Error Handling and Debugging

- How to fix step definition mismatches
- Duplicate steps
- Undefined steps
- Skipped steps due to previous failures

13. Cucumber Expressions (New Syntax)

- Difference between Regular Expression and Cucumber Expression
- {string}, {int}, {word} placeholders
- Writing clean and readable step definitions

14. Reusable Step Definitions

- Reusing common steps across scenarios
- Keeping steps modular and readable
- Common folder structure for easy reuse

15. Best Practices

- Keep feature files simple and readable
- Avoid too many steps in one scenario
- Group scenarios logically using tags
- Follow naming conventions
- Always dryRun before actual execution

REST ASSURED SYLLABUS

1. What is API and API Testing?

- API meaning (Application Programming Interface)
- Why API testing is important
- How API differs from UI
- Where API fits in SDLC
- Real-time example: Login, Fund Transfer in Banking

2. REST vs SOAP

- What is REST API?
- What is SOAP API?
- Difference between REST and SOAP
- Why REST is preferred for web services

3. Basics of REST API

- HTTP Methods:
    - GET
    - POST
    - PUT
    - DELETE
- HTTP Status Codes
- Headers and Body
- Path Parameters vs Query Parameters
- JSON & XML in API

4. Introduction to REST Assured

- What is REST Assured?
- Why REST Assured for API automation?
- Features of REST Assured
- REST Assured vs Postman
- Real-world use in banking: validate transactions, balances, login, etc.

5. Setting Up REST Assured in Java

- Add REST Assured dependency in project (using Maven or manual)
- Import required packages
- Setup basic structure for GET/POST testing

6. REST Assured Skeleton/Architecture

- Given() – Set pre-conditions
- When() – Call the API
- Then() – Validate response
- Understanding BDD style syntax

7. GET Request in REST Assured

- Writing simple GET API tests
- Validating status code
- Validating headers
- Validating response body (using JSONPath)

8. POST Request in REST Assured

- Creating JSON body
- Posting JSON payload
- Validating POST response
- Extracting ID/token from response

9. PUT and DELETE Requests

- Updating existing records with PUT
- Deleting a record using DELETE
- Validating updated/deleted status

10. Path Parameters and Query Parameters

- How to pass path parameter (e.g. /users/{id})
- How to pass query parameter (e.g. ?status=active)
- Examples from banking: Get transaction by ID, Filter accounts by type

11. Request and Response Specifications

- Creating reusable request spec
- Creating reusable response spec
- Helps reduce repeated code

12. Headers and Cookies

- Adding single/multiple headers
- Validating response headers
- Working with session cookies

13. Validating JSON Response

- Use of body() and hasItem(), equalTo()
- JSONPath validation
- Validating nested JSON fields

14. Extracting Response Data

- Extract specific fields from JSON
- Store response values for next API call
- Use extract().path() or extract().response()

15. Handling Authentication

- Basic Auth
- Bearer Token / OAuth2
- Passing token in header
- Example: Token-based login in a banking app

16. Schema Validation

- What is JSON schema validation?
- Validating response structure
- Writing and reusing JSON schema files

17. Logging and Debugging

- Logging full request/response
- Debugging failed requests
- .log().all() for complete visibility

18. Data-Driven Testing (Optional)

- Reading test data from Excel/CSV
- Looping test cases using Java logic

19. Assertions in REST Assured

- Validate status codes
- Validate response fields
- Validate response time
- Chained assertions for clean reporting

20. Real-Time Use Cases in Banking

- Login API → Token retrieval
- Balance check API
- Fund transfer API
- Transaction history API
- Mini-statement download API
- Validate error codes for invalid users

QA AUTOMATION SYLLABUS 2025

POSTMAN SYLLABUS

1. Introduction to Postman

- What is Postman?
- Why do we use Postman for API testing?
- Postman vs REST Assured
- Advantages of Postman for manual API testing
- Real-time usage in banking apps (Login, Balance Check, Fund Transfer)

2. Installing and Setting Up Postman

- How to install Postman (native app or browser version)
- Exploring Postman interface: tabs, history, sidebar, and settings

3. Understanding HTTP Requests

- GET, POST, PUT, DELETE, PATCH methods
- Structure of a request: URL, method, headers, body, parameters
- Real-time example: Banking Login API using POST

4. Sending Requests in Postman

- Entering endpoint URL
- Selecting method (GET, POST, etc.)
- Adding headers (Content-Type, Auth token)
- Sending request and viewing response

5. Working with Parameters

- Path parameters (e.g., /user/{id})
- Query parameters (e.g., ?type=savings)
- Adding parameters using Postman Params tab

6. Working with Headers and Cookies

- Common headers: Content-Type, Authorization
- Setting custom headers
- Viewing response headers
- Sending cookies and understanding session behavior

7. Authentication in Postman

- No Auth (Public APIs)
- Basic Auth (username & password)
- Bearer Token (Token in header)
- API Key Auth (key in header/query)
- OAuth 2.0 (overview and usage)

8. Request Body and Payloads

- Sending data in body using:
    - raw JSON
    - x-www-form-urlencoded
    - form-data
- Sending file upload (e.g., KYC document)

9. Validating Responses

- Status code check
- Viewing response time, size, body, and headers
- JSON response validation (success message, user ID, token)

10. Writing Tests in Postman (Basic JavaScript)

- What is the Tests tab?
- Writing test cases using pm.test()
- Example: Validate status code, response time, response body
- Common Postman test snippets (auto-suggestions)

11. Collection and Requests

- What is a collection?
- Creating a new collection
- Saving requests into collections
- Creating folders inside collections

12. Environment and Global Variables

- What are variables in Postman?
- Creating environment variables (e.g., baseURL, token)
- Using variables in requests ({{baseUrl}})
- Difference between global, environment, and local variables

13. Pre-request Scripts

- What is a pre-request script?
- Writing JavaScript before request execution
- Example: Generating timestamp or dynamic token

14. Collection Runner

- Running multiple requests at once
- Running tests from a collection
- Viewing test result summary
- Iterating with CSV or JSON data

15. Data-Driven Testing

- Importing CSV/JSON files
- Parameterizing requests with variable values
- Example: Test login with 5 different user credentials

16. Chaining Requests (Passing Data Between Requests)

- Extracting data from one request
- Storing values in variables using pm.environment.set()
- Using stored values in subsequent requests

17. Postman Console and Debugging

- Opening and using Postman Console
- Logging using console.log()
- Debugging request issues

18. Mock Servers (Optional)

- What is a mock server?
- Creating a mock server
- Using mock responses for testing

19. Real-Time Banking Scenarios

- Login API (POST) – get token and save to variable
- Fetch Account Balance (GET) – use token in header
- Fund Transfer (POST) – send amount, account numbers in body
- Transaction History (GET) – add date filter as query parameter
- Download Mini Statement (GET) – PDF file handling (optional)

20. Postman Best Practices

- Always use environments for base URL
- Use collections to organize APIs
- Use tests to validate every response
- Maintain tokens and dynamic data through variables

APPIUM SYLLABUS

1. Basics of Mobile Testing

- What is mobile application testing?
- Types of mobile apps: Native, Hybrid, Web
- Importance of mobile testing in QA

2. Real Devices vs Emulator vs Simulator

- What is an Emulator?
- What is a Simulator?
- What is a Real Device?
- Differences and when to use which
- Real-time use case: Testing fund transfer in banking app

3. What is Appium?

- Appium introduction
- Why Appium is used for mobile automation
- Appium vs Selenium
- Appium features and advantages

4. Appium Architecture

- Appium Client and Server
- JSON Wire Protocol and REST API
- How Appium interacts with device
- Appium Inspector overview

5. Prerequisites and Setup

- Install Java JDK
- Install Android Studio and SDK Tools
- Install Node.js and Appium Desktop
- Install Appium Inspector
- Set environment variables (JAVA_HOME, ANDROID_HOME)
- Enable USB Debugging on device
- Install Appium Java client dependency in Maven project

6. Device Setup

- Setup and connect Real Android Device
- Setup Emulator using Android Studio
- Get device ID using adb devices
- Check device connection using adb shell

7. App Under Test

- What is an APK file?
- How to get APK or app package name and activity
- Extract appPackage and appActivity using adb or tools

8. Desired Capabilities

- What are desired capabilities?
- Required capabilities:
    - platformName
    - deviceName
    - udid
    - appPackage
    - appActivity
    - automationName

- Optional capabilities:
    - noReset
    - fullReset
    - app
    - newCommandTimeout
- Real-time example: Setting up banking app login

9. Locating Elements in Mobile App

- Different locator strategies:
    - id
    - className
    - xpath
    - accessibilityId
    - androidUIAutomator
- Inspect elements using Appium Inspector and UIAutomatorViewer

10. Writing First Appium Script

- Setup project in Eclipse/IntelliJ
- Launch app and perform login
- Validate elements and print status
- Close app after test

11. Handling Mobile UI Elements

- Click, SendKeys, Clear
- GetText, isDisplayed, isEnabled
- Handle popups, dropdowns, notifications

12. Touch Actions and Gestures

- Tap and Long Press
- Swipe and Scroll
- Drag and Drop
- Multi-touch and zoom (advanced)
- Real-time example: Scroll to latest transactions in banking app

13. Wait Mechanism

- Implicit Wait
- Explicit Wait using WebDriverWait
- Fluent Wait (optional)

14. Handling Device Events

- Hide Keyboard
- Back button press
- Rotate screen
- Lock/Unlock device

15. Working with WebView (Hybrid Apps)

- Native context vs WebView context
- Switching context using getContextHandles()
- Interact with web elements inside hybrid apps
- Real-time example: Banking app payment page inside WebView

16. Handling Alerts and Toast Messages

- Accept and dismiss alerts
- Read toast messages using UIAutomator2
- Handle OTP alerts and system popups

17. Screenshots and Logs

- Capture screenshot on failure
- Save screenshots to folder
- Print logs to console
- Real-time example: Capture screenshot after failed login

18. Real-Time Banking Scenarios

- Login to banking app (enter phone and password)
- Validate OTP entry screen
- Check balance and mini-statement
- Fund transfer between accounts
- Download statement as PDF
- Logout from app

19. Debugging and Troubleshooting

- Handle Session Not Created Exception
- Handle Element Not Found
- Server/Port issues in Appium
- Common adb issues and how to fix them

20. Appium Best Practices

- Use stable locators (avoid dynamic xpath)
- Create reusable methods
- Use Page Object Model structure
- Keep app and device versions consistent
- Clear app data before/after test

Optional Topics (if needed)

- Appium with Page Object Model (POM)
- Parallel execution with multiple devices
- Appium with Grid (Advanced)

JMETER SYLLABUS

1. Introduction to Performance Testing

- What is performance testing?
- Why performance testing is important
- Types of performance testing:
    - Load Testing
    - Stress Testing
    - Spike Testing
    - Endurance Testing
    - Volume Testing
- Example: Test login performance of a banking app under 500 users

2. What is Apache JMeter?

- Introduction to JMeter
- Why JMeter is used for performance testing
- Features of JMeter
- Advantages of JMeter over other tools

3. JMeter Installation and Setup

- Download and install JDK
- Download and extract JMeter
- Launch JMeter GUI using jmeter.bat
- JMeter folder structure

4. JMeter Test Plan Structure

- What is a Test Plan?
- Add and configure:
    - Thread Group
    - Samplers
    - Listeners
    - Config Elements
    - Assertions
    - Timers
    - Pre/Post Processors
- Real-time analogy: Test plan = Blueprint of load testing

5. Thread Group

- What is a thread group?
- Key properties:
    - Number of Threads (Users)
    - Ramp-Up Period
    - Loop Count
- Example: Simulate 100 users login into a banking portal in 10 seconds

6. Samplers

- What are samplers?
- Common samplers:
    - HTTP Request
    - FTP Request
    - JDBC Request
    - SOAP/REST API Request
- How to send API requests using HTTP Sampler

7. HTTP Request in JMeter

- Setup API URL
- Choose method: GET, POST, PUT, DELETE
- Add request headers
- Add request body for POST methods
- Example: POST login API in banking app

8. Listeners

- What are listeners?
- Common listeners:
    - View Results Tree
    - Summary Report
    - Aggregate Report
    - Graph Results
    - Response Time Graph
- How to read graphs and analyze performance metrics

9. Timers

- What are timers in JMeter?
- Common timers:
    - Constant Timer
    - Uniform Random Timer
    - Gaussian Random Timer
- Add realistic wait time between requests

10. Assertions

- What are assertions?
- Types of assertions:
    - Response Assertion
    - Duration Assertion
    - Size Assertion
    - JSON Assertion
- Validate API responses (status code, text, time)

11. Configuration Elements

- What are config elements?
- Common config elements:
    - HTTP Request Defaults
    - HTTP Header Manager
    - CSV Data Set Config
    - User Defined Variables
- Setup base URL and headers at one place

12. Pre-Processor and Post-Processor

- What is a Pre-Processor?
    - Runs before request
    - Example: Add timestamp or token
- What is a Post-Processor?
    - Runs after request
    - Example: Extract data from response using JSON Extractor

13. Parameterization in JMeter

- What is parameterization?
- Using CSV Data Set Config
- Store test data in CSV file (username, password)
- Use variables in HTTP request
- Example: Login with 100 different users

14. Correlation in JMeter

- What is correlation?
- Extract dynamic values from response
- Use JSON Extractor or Regular Expression Extractor
- Example: Extract access token after login and pass to next request

15. Assertions and Validations

- Validate:
    - Status code
    - Response message
    - JSON value
    - Text match
- Example: Verify "Login successful" message for banking login API

16. Running Load Tests

- How to run the test
- Increase number of users
- Observe graph and response time
- Save and export reports

17. Analyzing Test Results

- Read response time, error %, throughput
- Understand bottlenecks
- Example: API becomes slow if users > 500

18. HTML Report Generation

- Enable reporting using non-GUI mode
- Use command line:
    - Run .jmx file
    - Generate HTML report
- Example: jmeter-n-t test.jmx-l result.jtl-e-o report_folder

19. Real-Time Banking Scenarios

- Load test login API with 1000 users
- Fund transfer API under 500 users
- Balance check API every 10 seconds
- View statement (looped request)
- Check how response time changes with load

20. JMeter Best Practices

- Use realistic test data
- Monitor CPU, memory, network while testing
- Use timers to simulate real user behavior
- Start with low load, then increase gradually
- Run tests in non-GUI mode for better performance

MYSQL SYLLABUS

1. Introduction to Databases and MySQL

- What is a Database?
- What is MySQL?
- Use of MySQL in QA & automation testing
- Difference between MySQL and other RDBMS (like Oracle, SQL Server, PostgreSQL)
- Real-world example: Banking application database

2. MySQL Installation and Setup

- Install MySQL on Windows/Linux/Mac
- MySQL Workbench overview
- Connect to local and remote databases
- Create and use databases

3. Basic SQL Queries

- SELECT, FROM, WHERE
- ORDER BY, LIMIT, DISTINCT
- Using IN, BETWEEN, LIKE
- Filtering rows with conditions
- Example: Get all customers whose balance > 10,000

4. SQL Functions

- Aggregate functions: COUNT, SUM, AVG, MAX, MIN
- String functions: CONCAT, LENGTH, SUBSTRING
- Date functions: NOW(), CURDATE(), DATEDIFF()
- Math functions: ROUND, CEIL, FLOOR

5. Data Manipulation (DML)

- INSERT INTO — Add new rows
- UPDATE — Modify existing rows
- DELETE — Remove rows
- Example: Update password for a specific banking user

6. Data Definition (DDL)

- CREATE TABLE, ALTER TABLE, DROP TABLE
- Data types in MySQL (VARCHAR, INT, DATE, etc.)
- PRIMARY KEY, AUTO_INCREMENT
- NOT NULL, DEFAULT constraints
- Example: Create a "transactions" table for a bank

7. Joins in MySQL

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL OUTER JOIN (workaround using UNION in MySQL)
- Self Join and Cross Join
- Real-time example: Get user and their transaction details from two tables

8. Grouping and Aggregation

- GROUP BY with aggregate functions
- HAVING clause for grouped conditions
- Example: Total number of accounts by account type

9. Subqueries and Nested Queries

- Scalar subqueries
- Subqueries in WHERE, FROM, SELECT
- EXISTS and NOT EXISTS
- Example: Get customers who made no transactions

10. Views and Indexes

- What is a View?
- Creating, updating, and deleting views
- Index types: single-column, multi-column
- How indexes speed up SELECT queries
- Example: Create a view for daily banking summaries

11. MySQL Constraints and Keys

- PRIMARY KEY, FOREIGN KEY
- UNIQUE, CHECK, DEFAULT
- Adding/removing constraints with ALTER

12. Transactions and Locks

- What is a transaction?
- BEGIN, COMMIT, ROLLBACK
- ACID properties
- Locking: implicit vs explicit
- Example: Transaction in fund transfer scenario

13. Users and Permissions

- Create database users
- Grant and revoke permissions
- Roles and security best practices
- Example: Give read-only access to QA team

14. Import and Export Data

- Import from .csv, .sql files
- Export tables or full DB to files
- Tools: mysqldump, LOAD DATA INFILE

15. Stored Procedures and Functions (Advanced)

- What is a stored procedure?
- Create, call, and drop procedures
- Input/output parameters
- User-defined functions
- Real-time example: Procedure to fetch daily transaction summary

16. Triggers (Advanced)

- What is a trigger?
- Create triggers for BEFORE or AFTER events
- Use cases: audit logs, automatic updates
- Example: Automatically insert into logs table when funds are transferred

17. Performance Optimization

- Query optimization tips
- Explain plan (EXPLAIN)
- Use of indexes
- Avoiding N+1 queries
- Optimize slow queries for regression automation

18. MySQL for Test Automation

- JDBC connection from Selenium/Java
- Read/Write data from MySQL in automation framework
- Use MySQL for test data validation
- Example: Validate balance after transaction via automation test

19. MySQL Best Practices

- Naming conventions
- Normalization (1NF, 2NF, 3NF basics)
- Avoid redundant data
- Backup and restore practices
- Version control for DB scripts

20. Real-Time Banking Scenarios

- Validate login user from DB
- Verify mini statement from DB vs UI
- Validate fund transfer using transaction table
- OTP expiry validation using datetime column
- Reconcile backend vs frontend balance post-transaction

MANUAL TESTING SYLLABUS

1. Introduction to Software Testing

- What is software testing?
- Why testing is important?
- Difference between Manual and Automation Testing
- Roles and responsibilities of a QA

2. SDLC (Software Development Life Cycle)

- What is SDLC?
- Phases of SDLC: Requirement, Design, Development, Testing, Deployment, Maintenance
- Models of SDLC: Waterfall, Agile, V-Model, Spiral, Iterative

3. STLC (Software Testing Life Cycle)

- What is STLC?
- Phases of STLC: Requirement Analysis, Test Planning, Test Case Design, Test Environment Setup, Test Execution, Test Closure

4. Types of Testing

- Functional Testing
- Non-functional Testing
- Static Testing
- Dynamic Testing
- White Box, Black Box, Grey Box Testing

5. Levels of Testing

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing (UAT)

6. Test Case Design Techniques

- Boundary Value Analysis (BVA)
- Equivalence Partitioning (EP)
- Decision Table Testing
- State Transition Testing
- Error Guessing

7. Test Planning and Documentation

- Test Plan
- Test Strategy
- Test Scenario
- Test Case
- Test Data
- Test Summary Report

8. Defect Life Cycle / Bug Life Cycle

- What is a defect?
- Defect status: New, Open, Assigned, Fixed, Retest, Closed, Reopen
- Severity vs Priority

9. Test Execution and Reporting

- Test Execution Process
- Logging results
- Reporting bugs
- Daily status reports
- Test summary report

10. Requirement Traceability Matrix (RTM)

- What is RTM?
- Purpose and usage
- Mapping requirements to test cases

11. Agile Testing

- What is Agile?
- Scrum framework basics: Sprint, Scrum roles, Ceremonies
- Agile Testing Process
- User Stories, Acceptance Criteria
- Definition of Ready (DoR) & Definition of Done (DoD)

12. Exploratory Testing

- What is it?
- When to use it?
- Advantages and limitations

13. Ad-Hoc Testing

- What is ad-hoc testing?
- Types: Monkey Testing, Buddy Testing, Pair Testing

14. Smoke and Sanity Testing

- Difference between Smoke and Sanity Testing
- When and where they are performed
- Real-time example in banking apps

15. Regression Testing

- What is regression testing?
- How to select test cases for regression?
- Importance in frequent release cycles

16. Compatibility Testing

- Browser compatibility testing
- Mobile compatibility
- OS compatibility

17. Usability Testing

- What is usability testing?
- Testing from end-user point of view
- Real-time example: Netbanking app user journey

18. Performance Testing Basics

- What is performance testing?
- Difference from functional testing
- Introduction to Load, Stress, Spike testing (JMeter used for automation)

19. Database Testing Basics

- What is DB testing?
- Validating data in UI vs DB
- SQL queries for backend validation

20. Web Application vs Mobile Application Testing

- Differences in approach
- Real-time use cases
- Challenges in mobile app testing

21. Test Management Tools Overview

- JIRA: Bug tracking & test case management
- TestLink / Zephyr (optional but useful)
- How to log a bug properly

QA AUTOMATION SYLLABUS 2025

AGILE SYLLABUS

1. Introduction to Agile

- What is Agile?
- Why Agile? (Need for flexibility and faster delivery)
- Difference between Agile and Traditional models (like Waterfall)

2. Agile Manifesto and Principles

- 4 values of Agile
- 12 principles of Agile development
- Customer collaboration over contract negotiation
- Responding to change vs following a plan

3. Agile Methodologies

- Scrum
- Kanban
- XP (Extreme Programming)
- Lean
- Crystal

4. Scrum Framework

- What is Scrum?
- Scrum roles: Product Owner, Scrum Master, Development Team
- Scrum events: Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective
- Scrum artifacts: Product Backlog, Sprint Backlog, Increment
- Sprint goal and commitment

5. Scrum Roles in Detail

- Product Owner: Responsibilities, backlog management
- Scrum Master: Responsibilities, removing blockers
- Development Team: Self-organizing and cross-functional

6. Scrum Ceremonies

- Sprint Planning: Inputs, outputs, estimating user stories
- Daily Stand-up (15-minute rule)
- Sprint Review: Demo of work done
- Sprint Retrospective: Inspect & improve

7. Product and Sprint Backlog

- Difference between Product Backlog and Sprint Backlog
- Grooming/Refinement sessions
- Adding new user stories, prioritization

8. User Stories and Acceptance Criteria

- What is a User Story?
- Format: As a [user], I want [feature], so that [benefit]
- INVEST principle
- Writing clear Acceptance Criteria
- Definition of Ready (DoR)
- Definition of Done (DoD)

9. Agile Estimation Techniques

- Story Points
- T-shirt sizing (S, M, L, XL)
- Planning Poker
- Velocity: How it's calculated and used

10. Burndown and Burnup Charts

- What is a Burndown Chart?
- What is a Burnup Chart?
- Tracking progress during a sprint
- Velocity tracking and forecasting

11. Agile Tools

- JIRA (backlog, board, burndown chart)
- Trello (basic kanban)
- Rally (for large-scale Agile)

12. Agile Testing

- Role of QA in Agile
- Continuous Testing and Feedback
- Test early, test often
- Automation in Agile (quick regression cycles)

13. Agile vs Waterfall (Real-Time Comparison)

- Flexibility vs fixed scope
- Iterative vs sequential delivery
- Real example from a banking project: change in requirements mid-sprint

14. Agile Challenges and Best Practices

- Changing requirements
- Team communication
- Handling incomplete stories
- Handling dependencies between teams

15. Scaled Agile (SAFe) – Basics

- What is SAFe?
- PI Planning
- ART (Agile Release Train)
- Large team coordination
- Optional for interviews unless applying to senior/lead roles

---