**GENERAL TOPIC TO KEEP CHECKING**.

**BACK TO BASICS – WHAT and HOW, WHY.**

**STUDY and PRACTICE EVEN IF IT IS FOR 10 minutes.**

SDET – Test Automation Engineer (Software Quality Engineer)

Client Server Architecture

Types of clients and servers.

Computer Science (Hardware and Software systems), Software Engineering and Development (SDLC).

Back to Basics.

Lifecycle of software.

Always have a design or development or construction plan.

Review, Re-Study everything – Lifelong Learning.

ALWAYS HAVE A PLAN, A PLAN IS YOUR MAP BEFORE YOU START THE JOURNEY.

WHAT AND HOW.

**BUILD ALL KINDS OF PROJECTS – LEARN OTHER DEVELOPERS PROBLEM SOLVING LOGIC.**

**USE REAL WORLD OR LIFE EXAMPLES.**

**THINK LOGICAL LEFT, RIGHT, UP AND DOWN**

Understand WHAT the theoretical logical knowledge is, HOW to Implement the practical hands-on exercises, Apply to a specific problem.

UNDERSTAND **Front-end – Middleware – Backend technologies** used for developing software application, **WHAT** software testing tools are built for testing these applications and **HOW** to use them to solve testing needs.

STUDY and familiarize and play around with **COMPUTER SOFTWARE, PROGRAMS and Programming languages**. E.g Specflow, TestNg, Nunit, Java, CSharp, RestAssured, Karate API, Selenium, Python and many more.

**Computer Files and Data –** file types are defined by the file extensions and **WHAT** kind of data can be stored in file and **HOW** to construct or create the data syntax vocabulary rules in the file and how to access data in the file**.**

**Computer Programs allows a user to complete a specific task.**

* SDET – Test Developer, Test Automation Engineer
* Test Automation RoadMap
* Software Engineering, Computer Science and Software development.
* **Front-end, Middleware and Backend software.**
* Study order developers code, study how they logically solve a programming problem.
* Computer Networking, Computer programming, Computer software application, computer system.
* CICD, Selenium GRID
* ROI – Return on investment.
* UML diagram and Entity relationship modelling.
* Model an object.
* All Objects communicate to each other using a common interface.
* **SOLID** principles.
* desiredcapabilities
* **Logical Reasoning and Analysis.**
* **Computer hardware and Computer software or computer program.**
* **Computer Software (Program) System software, Operating Systems (Linux, windows, Android, IOS), Network Operating Systems.**
* **Application software (Command-Line, Software libraries – API, COM), GUI web-based, desktop, mobile, virtual software application.**
* **Mathematics and Physics.**
* **Understand singular and plural English.**
* Characters for data – data forms information – information forms knowledge – knowledge forms wisdom.
* Forms of learning – read text, videos, explain and apply to solve a specific problem.
* Base64 mechanism.
* **Problem Solving Skills (identify the objects input, process and output) - Algorithm (logical steps flow-chat, pseudocode) – Implementation – Maintenance.**
* **PLAN LOGICAL the UML and ER diagram WORKFLOW (precondition/pre-requisites – now condition – post condition) model when BUILDING Test Automation Frameworks to give you a MAP to guild you as you build.**
* **A LOGICAL WORKFLOW it is a sequence or series of operations or processes through a task or piece of work goes through form the start to completion, breaking each piece in (Analytical Skills) unit step by step to complete a task.**
* **PLANNING helps to breakdown a workflow**.
* **Program** – input process output storage.
* **Object** – state and behaviour.
* **It can be called Program, Element, Object, Resource, Service, Entity, Record.**
* **It can be called property, field, attribute, input, key.**
* **Data – set of value (singular or plural)**
* **Related objects as attributes or properties that REPRESENT an OBJECT.**
* **Operations, Mechanism, Actions, Process, Function (statements), Method.**
* Defects and bug.
* Debugging.
* Download COMPLETE REFERNCE BOOKS.
* **Study with textbooks, Videos, explain and apply to a specific problem**.
* Nth tier layer (phase) architecture.
* Presentation – Business Logic layer – Implementation Layer.
* Design Patterns – Creational, Structural and Behavioural (Flowchart, pseudocode), Data Structures.
* UML AND ER diagram.
* **Problem Solving Skills – Algorithm – Implementation - maintenance.**
* **Analytical Skills and Soft skills.**
* Lambda.
* Follow instructions, Rule, Constraints.
* Version Control Systems GIT and GITHUB, BITBUCKET, Software application programs.
* Java, JavaScript, C#, Python.
* Clues, ideas, guess the keywords (define it and look for object relationship)
* Object REPRESENTATION.
* Client Server architecture.
* Required relation properties that REPRESENT an object, e.g. Student.
* PUSH and always PULL mindset.
* Annotations and Attributes (A programmer gives a set of logical instructions to the **compiler to process a program**).
* Unit Testing Frameworks.
* Serialization and Deserialization.
* TestNG, Specflow, Cucumber, Nunit, Pytest.
* Features or Functionality on a program, API, Software application.
* IDE – **Compiler.**
* UNDERSTAND **Compile time, Build-time, Runtime Objects or Types.**
* Explicit and Implicit meaning.
* **WHAT and HOW.**
* **Local and Remote machines.**
* **Client Server architecture.**
* **There are many approaches or ways to apply a solution to a problem.**

**Software Testing**

* Specifications, Requirements (needs)
* Defect
* Types and Kinds of testing.
* Testing books.

**Object Oriented Programming Languages**

* Java – access (getters) and mutators (setters).
* C#
* Python
* JavaScript