**GENERAL TOPIC TO KEEP CHECKING**.

AS YOU ARE UPSKILLING, ALSO STUDY THE BUSINESS SIDE TOO.

YOU ARE A TEST DEVELOPER (A software tester and A software Developer at the same time).

Explore what and how to use software testing tools.

Computer Science, Software Engineering and Development.

Knowledge(theory) and Skill (practical hands-on)

**Frontend tools (client tools) – Middleware tools (Web Services, API, RestAPI, SOAPUI, Servers, Services) – Backend tools (database like MongoDB, MySql, Microsoft server sql, Postgre, SQLlite)**

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

**Review, Revisit, Re-learn, Re-read, Re-Practice. (LOGICAL guilds the PHYSICAL).**

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

**7 days of studying and practicing a program.**

SDET – Test Automation Engineer (Software Quality Engineer).

Client - Server Architecture. (Where is the software application program installed or located – local or remote?)

Local and Remote computer (system) machines.

Types of clients and servers. (Determine if a software application program is either a client or server depending on the current context).

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

Back to Basics (WHAT and HOW).

Lifecycle of software PROGRAM.

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.

What is a System?

What is a software

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

**USE REAL WORLD OR LIFE EXAMPLES.**

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

**Familiarize yourself, try, study and learn it, do it now, so that next time, it does not look new to you. You continue and gain more the next time you encounter it.**

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, Scripting Languages**. E.g Specflow, TestNg, Nunit, Java, CSharp, RestAssured, Karate API, Selenium, Python and many more.

Packages, namespace, module contains API for programmers to solve a need programmatically.

**Computer Files / Document 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.**

* Every program has a documentation or API user guide. Look for it and read it.
* 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, Jenkin, 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.
* **Database – JDBC, MYSQL, ORACLE, MICROSOFT**
* RemoteWebdriver, desiredcapabilities.
* Unit Testing Frameworks - Junit, TestNG, NUnit
* **Logical Reasoning and Analysis.**
* **Understand what a software and types of software is.**
* **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, watch videos, visual images or diagrams, explain and apply to solve a specific problem**.
* Base64 mechanism.

================================================================

* **Task/Need/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 is a sequence or series of operations or processes through which a task or piece of work goes through from the start (initialization) to completion, breaking each piece into (Analytical Skills) unit by completing each step for the task.**
* **PLANNING helps to breakdown a workflow**.
* Encoded and Decoded meanings.

**Syntax construction and rules.**

* **Program** – input process output storage (variables, indexers, index, key/value pair concept).
* **Object** – state and behaviour.
* **It can be called Program, Type, Element, Object, Resource, Service, Entity, Record, NODE.**
* **It can be called property, field, attribute, input, key.**
* **Data – set of value (singular or plural)**
* **Data-Type - the data determines what type can hold that value.**
* **Grouping Related Required logical objects as attributes or properties or fields or columns or inputs or keys that REPRESENT an OBJECT.**
* **Operations, Mechanism, Actions, Process, Function (statements), Method, Implementation of HOW.**
* Defects and bug.
* Maven build-tool.
* HTML, CSS, JavaScript, XPATH, CSS – Selector, DOM, NODEE.
* Debugging Tools.
* Download COMPLETE REFERNCE BOOKS.
* **Study with textbooks, Videos, explain and apply to a specific problem**.
* Nth tier layer (phase) architecture. Loosely couple architecture.
* Presentation – Business Logic layer – Implementation Layer.
* Design Patterns – Creational, Structural and Behavioural (Flowchart, pseudocode), Data Structures – linear and non-linear.
* UML AND Entity Relationship diagram.
* **Problem Solving Skills – Algorithm – Implementation - maintenance.**
* **Analytical Skills and Soft skills.**
* Lambda.
* Program -> Object -> Type -> Value and Reference Types.
* Follow instructions, Rule, Constraints, Requirements, SPECIFICATIONS.
* 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.
* Scripting Languages.
* Required relation properties that REPRESENT an object, e.g. Student.
* PUSH and always PULL mindset.
* **Annotations and Attributes (metadata) (A programmer gives a set of logical instructions to the compiler to process a program).**
* **A programmer instructs the COMPILER to execute the source code in a sequentially orderly manner, HE direct the execution workflow.**
* **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.
* Software testing manual and automation tools.
* Verification, Validation and Assert.
* Think Positive and Negative testing, Think right, left, up and down.

**Object Oriented Programming Languages**

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

**Functional Programming Styles**

**Study programming and BUILD, BUILD, BUILD**

* Understand the specification/requirements in Problem Solving statement (Skill - identify the **object** (nouns) (apply **Object Oriented Principles**) and their state and behaviour, input process output.
* Create **loosely (creational and structural)** coupled (**Behavioural**) layered architecture (implementation - apply OOP principles), then Algorithm step by step workflow.
* Presentation (WHAT layer) – Business Logic Layer (implementation of HOW) – Database layer.
* **Design Pattern**: Creational – Structural – Behavioural.
* **Study other developers code to find out how they applied logical solutions to solve a problem programmatically**.
* **Workflow, statements**.
* **You need to know the RULE and CONSTRAINTS of the problem, know what is mandatory and optional.**
* **THINK LOGICAL.**
* **Study technical prep interview contents (videos and texts, blogs).**
* **Mathematic and English.**
* **Logical Sequential workflow.**
* **Cucumber, Specflow, Behave.**

**A Function / Method**

* A function performs a specific functionality or feature.
* One data is used more than once, then it must be created inside a function and reused.
* Once a step in a function has sub-steps to get an output, create a separate function and call it into that main function.
* Function names are action **verbs** (action words).
* Function contains a block of statements.
* Step by step execution of statements.
* Data type – e.g (Boolean condition) followed by an identifier, and it must be named according to reason or purpose.
* The data determines the Type used to hold the data.

condition is the data

Boolean is the Type

* Expression – lambda
* Time and Space complexity

**HOW to learn a Software or Program or Plugin/Extensions**

* What programming language is it built on.
* Learn the **inputs (inputs are types, the data determines what kind of data the input can hold, e. string type holds string value or data)**, **processes (process data held in the inputs)**, **output (process returns a value or void), and storage (store return or assigned data)** of the software.
* Learn the **Commands**, **features,** or **functionalities**.
* What other software does it integrates with.
* How to use the software features or functionalities.
* When to apply the software features.
* Why do need to use the software features.
* Some Objects are Built-time objects, Compile-time objects, Run-time objects, they instantiate data before or after execution.