**Plan your recipe.**

**Take 5-minute breaks when you study (Time Management)**

**Review, Re-read, remind yourself, Re-Learn nth number of times.**

**SDET – TEST AUTOMATION ARCHITECT. TEST DEVELOPER.**

**PRACTICE THE TOOLS Features and Functions.**

**Google search -> Full Stack test Automation engineer**

**Test Automation RoadMap**

**Do & Build Projects.**

**Mastery Techniques – Logically doing something at least 5 times repeatedly to perfect the skills or habit.**

* What is learning and how to learn BACK TO BASICS.
* Learn the rudiments of programming.
* Learning graph and types of learning.
* Software Programming, Software Engineering, Software DEVELOPMENT.
* Computer Science and Information systems.
* Software Testing.
* Types of API’s – Web-Services, API, Software Libraries, COM, Restfull API.
* Abstraction (BLUEPRINT - a prototype object) concept in OOP.
* REPRESENTATIONAL state transfer.
* Data and Datatypes.
* File types and file systems.
* Code and Data.
* GUI and UI. – Technical and Non-Technical users.
* How to handle COMPLEX INFORMATION – ANALYTICAL SKILLS.
* Soft Skills,
* End User.
* Memory build – remembering techniques.
* Operations, Methods, Actions, Mechanism, Process.
* Mathematics and English.
* Sequential Workflow (A complete End 2 End transaction (events) between nth objects, Start – End. Open - Close)
* E e.g the flow of work in a YAML script.
* Collections and lists – Singular / Plural.
* Database software application programs and SQL – result set, entity records.
* Page Object Model.
* Object transactions.
* Accessing data from an object.
* Client Server relationships.
* Local and Remote.
* Front-end, Middleware, Backend
* Business Rules and needs, Specifications, Requirements.
* Computer system.
* Software **Systems** (OS, NOS)
* Software **applications** and Software applications Programs.
* APPROACH – ways to logical break or decompose the problem into logical units.
* Divide and conquer, loosely coupled concept.
* Understanding OOP principles and patterns.
* Procedural programming, OOP principles, Functional Programming.
* Modular Programming.
* Automation Interview QUESTIONS.
* Test Automation Architecture.
* Nth tier layer architecture.
* Design Patterns AND Object relationships.
* Behaviour – interaction & interfaces.
* Git and GitHub, Bitbucket.
* EXECUTION – compile-time – Build-time – Runtime.
* Clue, Guess and what that object is related to.
* RERESENTATION meaning.
* Problem Solving
* Algorithm (recipe) – pseudocode and flowchart.
* Modelling UML, ER diagrams.
* Mobile, Web, Desktop apps.
* Interfaces meaning e.g (API).
* Base (Supertype) and Derived (Subtype).
* Unit Testing Frameworks are used for Before and After test executions. Junit, NUnit, TestNG.
* Annotation and Attributes – An end user or a programmer gives a set of instructions to an engine (interpreter or compiler) to sequentially order the execution of an application script.

**Note: if you are trying to do something and it not working, drop it, take a walk, exercise, see a movie, or do anything not related to that problem you trying to solve. However, while you are busy with something else, think about what you are trying to solve, An idea or solution with drop in your logical mind (physical brain).**

**Soft Skills -> Pay attention detail information and to words and their definitions and meaning.**

**Focus on the GOAL and Objective.**