**C# Notes**

*Intro Basis (Q to ChatGPT):*  
With regards to C# Automation API and Back End Testing, if you look at all the relevant topics of C# software development programming, understanding and implementation and all the categories and topics that a person needs to have an understanding of. This will all range from Beginner, to Intermediate to Advanced levels.

Give me a list of all the skills, topics, categories, etc that a person needs and split this into the 3 levels (Beginner, Intermediate and Advanced).

Your list needs to be all encompassing to handle all areas needed to be contributing to a team's code base asap.

*Response from ChatGPT:*

Here’s a **comprehensive roadmap** split by **Beginner**, **Intermediate**, and **Advanced** levels, specifically tailored for a developer aiming to contribute to a **C# Automation API and Backend Testing** project.

This guide covers the **C# language**, **automation**, **API testing**, **back-end fundamentals**, and the **supporting ecosystem/tools** required to be productive in a modern .NET team environment.

Here’s a **comprehensive roadmap** split by **Beginner**, **Intermediate**, and **Advanced** levels, specifically tailored for a developer aiming to contribute to a **C# Automation API and Backend Testing** project.

This guide covers the **C# language**, **automation**, **API testing**, **back-end fundamentals**, and the **supporting ecosystem/tools** required to be productive in a modern .NET team environment.

**✅ Beginner Level**

**Goal**: Understand the fundamentals of C#, .NET, basic testing concepts, and essential tooling.

**🧠 C# Language Basics**

* Variables, data types, and operators
* Control structures (if, switch, loops)
* Functions and parameters
* Error handling (try/catch/finally)
* Classes and objects
* Namespaces and assemblies
* Basic debugging

**🧰 .NET & Project Structure**

* Introduction to .NET Core / .NET 6+
* Project types: Console app, Class Library, ASP.NET Web API
* Using Visual Studio / VS Code
* Project structure & solution organization

**🧪 Unit Testing Basics**

* Introduction to unit testing
* xUnit / NUnit basics
* Writing basic test methods
* Test project structure
* Running tests from IDE/CLI

**🌐 Web & APIs**

* Understanding HTTP methods (GET, POST, PUT, DELETE)
* REST vs SOAP
* Consuming APIs using HttpClient
* JSON serialization/deserialization with System.Text.Json or Newtonsoft.Json

**📦 NuGet and Package Management**

* Installing and managing NuGet packages
* Common packages: xUnit, RestSharp, FluentAssertions

**💻 Basic Version Control (Git)**

* Cloning a repo
* Basic commands: add, commit, push, pull
* Branching and merging

**🚀 Intermediate Level**

**Goal**: Build solid backend test automation using real-world patterns, contribute to a codebase, and understand best practices.

**💻 Intermediate C# Concepts**

* OOP principles (Encapsulation, Inheritance, Polymorphism)
* Interfaces and abstraction
* Generics
* Enums, Structs
* LINQ
* Collections and dictionaries
* File I/O basics

**🔍 Test Automation Skills**

* Writing modular and reusable tests
* Test setup and teardown methods
* Data-driven testing
* Mocking (using Moq or NSubstitute)
* FluentAssertions or similar assertion libraries
* Using test runners in CI pipelines

**🌐 API Testing**

* Automated testing of RESTful APIs
* Using RestSharp / HttpClient with test frameworks
* Authentication & Authorization (Bearer tokens, Basic Auth)
* Postman collections to C# tests migration
* Contract validation (JSON schema)

**🛠 Test Frameworks & Tools**

* xUnit/NUnit: Fixtures, Theory/Data tests
* RestSharp for API interactions
* WireMock.Net or MockHttp for API mocking
* Serilog or similar logging for test output

**🏗 Backend System Concepts**

* RESTful API design principles
* HTTP Status codes & headers
* JSON / XML formats
* Basic understanding of databases (SQL Server, SQLite)
* Using EF Core or Dapper (read-only/test context)

**🧪 Integration Testing**

* API-level integration tests
* Setting up test environments
* Working with in-memory databases (InMemoryProvider, SQLite)
* Test doubles (Fakes, Stubs, Mocks)

**⚙️ Dependency Injection (DI)**

* Understanding DI in .NET Core
* Configuring services for testing
* Overriding services in tests

**📂 Source Control & Collaboration**

* Understanding PR workflows
* Code review basics
* Resolving merge conflicts
* Using GitHub Actions or Azure DevOps for CI

**🧠 Advanced Level**

**Goal**: Take full ownership of test architecture, mentor others, and improve team productivity through tooling and strategy.

**💡 Advanced C# Topics**

* Asynchronous programming (async/await, Task)
* Delegates, Events, Func, Action
* Expression Trees and Reflection
* Extension Methods
* Attributes and Annotations
* Memory management and performance tuning

**🧪 Advanced Testing Practices**

* Test architecture (layered testing: Unit, Integration, E2E)
* Custom assertions
* Building internal test frameworks/libraries
* Contract testing (e.g. using PactNet)
* Test parallelization
* Snapshot testing
* Load and performance testing with k6 or NBomber

**🔐 Security & Auth Testing**

* OAuth 2.0 / OpenID Connect flows
* Testing APIs with JWT tokens
* Role-based access testing

**🛠 Advanced Tools & Libraries**

* AutoFixture for test data generation
* FluentValidation for validation testing
* SpecFlow or BDD frameworks
* Dockerized testing environments
* Using Selenium or Playwright for API + UI flows

**🧬 CI/CD Integration**

* Writing test stages in CI/CD pipelines (YAML)
* Managing test reports (HTML, Allure, TRX)
* Test result publishing in Azure DevOps/GitHub Actions
* Environment configuration (test/staging/prod)

**🧱 Architecture & Design Patterns**

* Clean Architecture
* SOLID principles
* Repository pattern
* Builder, Factory, Strategy patterns
* Testability in code design

**📊 Observability & Metrics**

* Logging strategies in tests
* Monitoring test health/trends
* Integration with tools like Application Insights or Elastic Stack

**👥 Team Leadership & Contributions**

* Enforcing code standards in tests
* Driving test automation strategy
* Mentoring juniors
* Performing code reviews for test quality

**🧩 Optional / Cross-cutting Topics (All Levels)**

These can appear at any level depending on the project:

* YAML / JSON configurations
* Secrets management in test runs
* Using environment variables for test config
* REST API documentation tools (Swagger/OpenAPI)
* Message queues (RabbitMQ, Azure Service Bus – for back-end integration testing)
* Feature flags and testing around them

*Resources Encountered:*

Mocks:

<https://thecodebuzz.com/unit-test-mock-httpclientfactory-moq-net-core/>

<https://nsubstitute.github.io/>

<https://github.com/devlooped/moq/wiki/Quickstart>

GitHub CodeBase examples:  
<https://github.com/restsharp/RestSharp/>

Framework Documentation:  
<https://docs.nunit.org/articles/nunit/writing-tests/attributes/testcase.html>

YouTube C# Course (not sure of viability):  
<https://www.youtube.com/watch?v=GhQdlIFylQ8>

Assertion Library (Fluent):  
<https://fluentassertions.com/introduction>

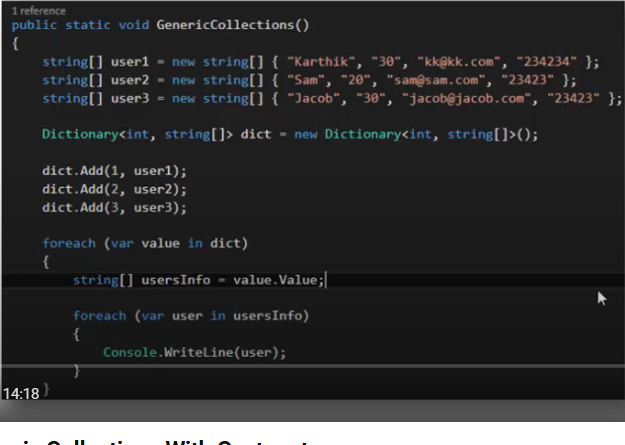
RestSharp Documentation:  
<https://restsharp.dev/docs/advanced/serialization>

Serialization and Deserialization Documentation:

<https://www.newtonsoft.com/json/help/html/SerializingJSON.htm>

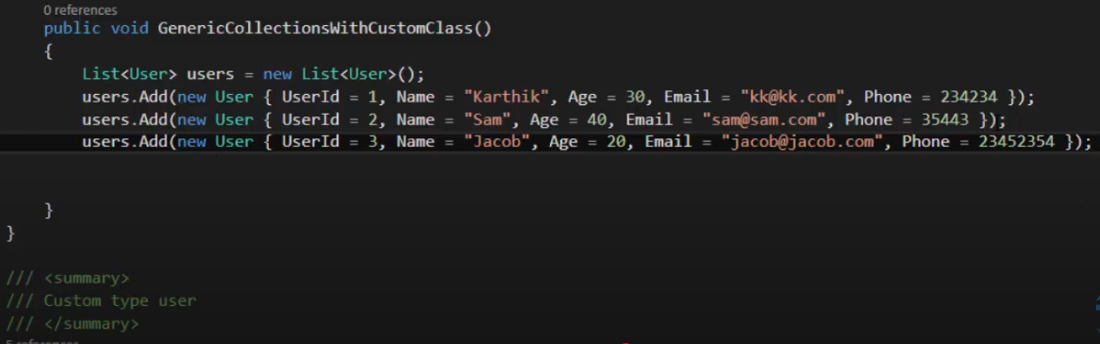
(need to find for System.Text.Json docs) <https://learn.microsoft.com/en-us/dotnet/api/system.text.json?view=net-9.0>

TODO:

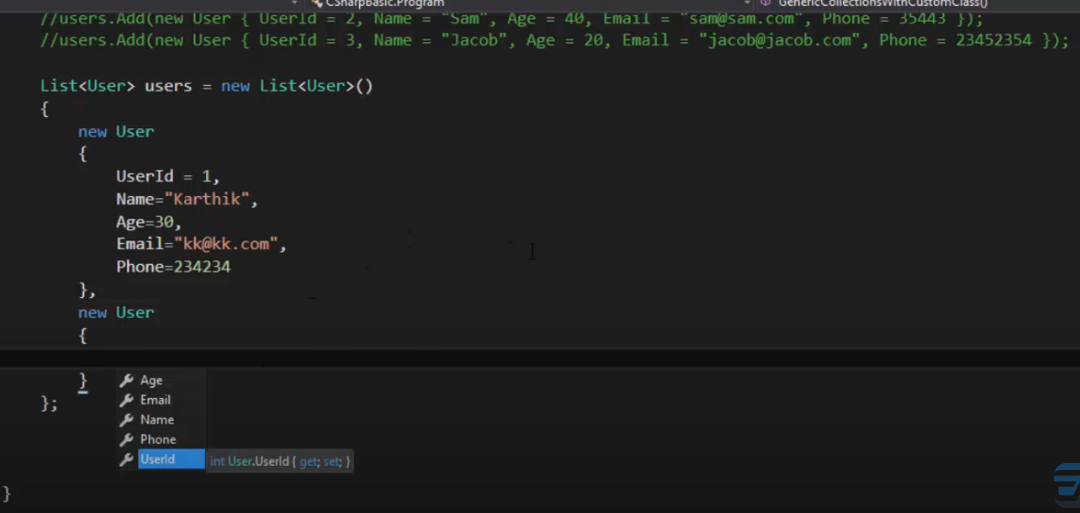
* + Non generic collection
  + Types of generic<> collection:
    - List<>
    - Dictionary<Tkey,Tvalue>
    - Stack<T>
    - E.g. 
  + How to use Generic Collections with Custom Types? Converting the above example to use a Custom Type = User.
    - E.g.

A screen shot of a computer program

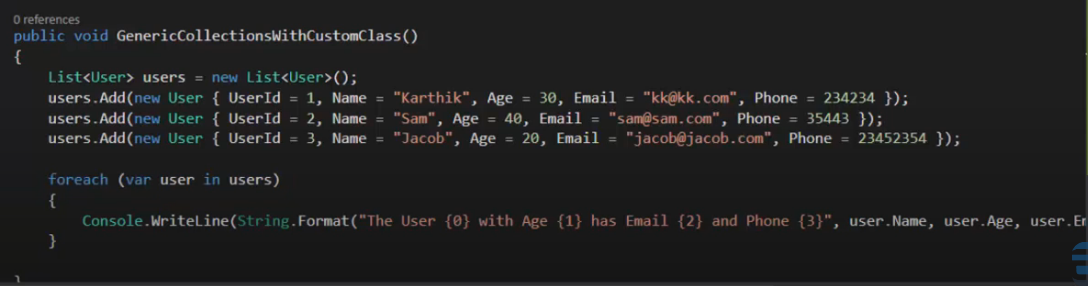
AI-generated content may be incorrect.



Or alternatively…



And, if you needed to iterate through the dictionary with the custom types…



* + LINQ Query