AI-DRIVEN REST API TESTING

API Automation and Testing

Lappeenranta–Lahti University of Technology LUT

Master's Programme in Software Engineering and Digital Transformation

2023

Isuri Navarathna Mudiyanselage

Examiner(s): Associate Professor Jussi Kasurinen

ABSTRACT

Lappeenranta-Lahti University of Technology

School of Engineering Science

Software Engineering

Isuri Navarathna Mudiyanselage

AI-Driven Rest Api Testing

Master's Thesis

2023

67 pages, 19 figures, 5 tables, 1 appendix

Examiners: Associate Professor Jussi Kasurinen

Keywords: Software Testing, API Testing, Software Automation, REST API, Artificial Intelligence, Machine Learning.

Artificial intelligence has become increasingly popular for its capability to automate complex tasks, and software engineering is no exception. This thesis offers a comprehensive overview of the intersection between artificial intelligence and web API testing. It explores how artificial intelligence can enhance API testing and automation, providing valuable insights into the potential benefits and advancements in this field. The primary objective of this study was to review existing research solutions for API testing and gather insights from software professionals regarding the current challenges they face with API testing tools. Additionally, the study aimed to understand the professionals' expectations for AI-powered API tools in the future. Finally, a conceptual framework was proposed to address these challenges. The study specifically focused on four main areas: web API, current AI solutions, existing gaps and limitations, and the proposed conceptual framework to overcome the current issues in API testing.

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

School of Engineering Science

Tietotekniikan koulutusohjelma

Isuri Navarathna Mudiyanselage

Tekoälyohjattu Rest Api -testaus

Pro gradu tutkielma

2023

67 sivua, 19 kuvaa, 5 taulukkoa ja 1 liitettä

Tarkastaja(t): Apulaisprofessori Jussi Kasurinen

Avainsanat: Ohjelmistojen testaus, API-testaus, ohjelmistoautomaatio, REST API, tekoäly, koneoppiminen.

Tekoälystä on tullut yhä suositumpi sen kyvystä automatisoida monimutkaisia tehtäviä, eikä ohjelmistosuunnittelu ole poikkeus. Tämä opinnäytetyö tarjoaa kattavan yleiskatsauksen tekoälyn ja web API -testauksen risteyskohdaksi. Se tutkii, kuinka tekoäly voi tehostaa API-testausta ja automaatiota ja tarjoaa arvokkaita näkemyksiä tämän alan mahdollisista eduista ja edistysaskeleista. Tämän tutkimuksen ensisijaisena tavoitteena oli tarkastella olemassa olevia API-testauksen tutkimusratkaisuja ja kerätä ohjelmistoalan ammattilaisilta näkemyksiä API-testaustyökalujen nykyisistä haasteista. Lisäksi tutkimuksella pyrittiin ymmärtämään ammattilaisten odotuksia tekoälypohjaisille API-työkaluille tulevaisuudessa. Lopuksi ehdotettiin käsitteellistä viitekehystä näihin haasteisiin vastaamiseksi. Tutkimus keskittyi erityisesti neljään pääalueeseen: web API, nykyiset tekoälyratkaisut, olemassa olevat puutteet ja rajoitukset sekä ehdotettu käsitteellinen kehys API-testauksen ajankohtaisten ongelmien ratkaisemiseksi.

ACKNOWLEDGEMENTS

I extend my sincere gratitude to my supervisor, Jussi Kasurinen, for his invaluable guidance and support throughout the entire thesis process. His assistance in selecting an engaging topic and providing a flexible schedule has been immensely appreciated. The journey from the first email to the completion of this thesis has spanned approximately 7 months, during which I have gained a wealth of knowledge and personal growth. I am confident that the insights gained from this experience will serve as a solid foundation for my future career advancements.

I would like to express my heartfelt gratitude to my friends and family for their unwavering support throughout this journey. Your assistance in providing valuable insights and participating in the survey has been truly invaluable. Your encouragement and support have played a vital role in helping me gain the necessary understanding and completing this endeavor. Thank you for always being there for me.

To my dearest husband, Madushan, I want to express my heartfelt gratitude for your unwavering support and infinite patience throughout my academic journey. Without you by my side, I would never have been able to reach this significant milestone in my studies. Words cannot adequately convey how thankful I am for everything you have done. You truly are my rock, and I appreciate you more than words can express.

ABBREVIATIONS

Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
API	Application Programming Interface
REST	Representational State Transfer
CI/CD	Continuous Integration/Continuous Deployment
GUI	Graphical User Interface
DRL	Deep Reinforcement Learning
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol
TCP	Transmission Control Protocol
CSV	Comma-Separated Values
QA	Quality Assurance

Table of contents

Abstract

Acknowledgements

Abbreviations

Table of Contents

Introduction **1**

Research Objectives and Questions 3

Related research **4**

Web APIs 4

Web API Testing and Automation 4

What is AI and bots? 7

AI and Web API 9

Research Methods **17**

Survey 19

Findings of literature review **20**

Survey Results **22**

Discussion **42**

Framework Concept 47

Conclusion **51**

References **52**

Appendices

Appendix 1. Survey

List of Figures

Figure 1: Approach by Ed-douibi et al. (2018) 12
Figure 2: Study process 18
Figure 3: Respondents' API testing level experience 22
Figure 4: Respondents' API automation level experience 22
Figure 5: Respondents' API automation level experience in years 23
Figure 6: Respondents'' ratings about the importance of API automation 24
Figure 7: Tools/Frameworks used for API testing and automation 24
Figure 8: Challengers experienced 25
Figure 9: Respondents'' awareness of the AI concepts 26
Figure 10: How respondents think AI can improve API test automation 27
Figure 11: Are respondents consider to adopt AI concepts 27
Figure 12: Concerns of incorporating AI to API test process 28
Figure 13: Expected features an AI tool can have 29
Figure 14: Usage of AI tools for API testing and automation 30
Figure 15: Experience about limitations or challengers 30
Figure 16: AI tool awareness of the respondents 31
Figure 17: Benefits of using AI-based tools 32
Figure 18: Concept to gain knowledge 47
Figure 19: Concept for a future tool 49

List of Tables

Table 1: An overview of GUI test automation solutions 10
Table 2: Ranking of the challenges faced during API automation 26
Table 3: Future with AI as per respondents 35
Table 4: Categorization of the future of test with AI answers 38
Table 5: Opinions about the quality of the survey 41

1 Introduction

In today's world, software testing is heavily reliant on test automation. Demand for software testing using the fewest resources grows along with software complexity. When it comes to consuming resources, execution speed, and test coverage, manual software testing operations seem to be unproductive. These are the precise issues that test automation can solve, and it does so in the majority of circumstances. Software automation was defined by Dustin et al. as ”*automation of management and performance of test activities containing development of test scripts, their execution, and verification of test requirements with the help of appropriate automated testing tools. Nevertheless, test automation should be considered as a broader term, including not only the automated test scripting and execution but also other activities across the whole software testing process”*. Even though automation has its own drawbacks and issues, it is impossible to imagine quality assurance in the modern world without it. like fragile automation scripts or inadequate defect detection. These, however, are problems that some artificial intelligence techniques and systems have previously addressed. However, there are still some areas in quality assurance that call for additional AI applications and research. One is API testing, which is frequently referred to as integration testing in the test pyramid.

Web API testing is becoming more important than ever because a defect in a business's API could have a significant effect on both the services that depend on that API as well as externally from third-party apps and end consumers. The majority of currently used tools and testing methodologies call for instrumenting the system being tested or creating tests. When creating Web APIs, REpresentation State Transfer (REST) has emerged as the method of choice. REST, the most popular sort of web service at the moment, is used by many businesses, like Google, Amazon, Twitter, Reddit, and LinkedIn, to offer APIs to their products and services. REST explains the architectural tenets, characteristics, and limitations needed to create distributed hypermedia platforms on an Internet scale. The fact that REST is a design approach and does not depend on any form of standards to specify REST APIs makes it challenging to develop, evaluate, and integrate REST APIs (Ed-douibi et al., 2018). There are many difficulties in testing web services, especially RESTful web services. Various methods have been suggested. However, rather than REST, the majority of the work done at this point in the literature has focused on SOAP web service black-box testing (Andrea , 2019). SOAP is a clearly defined protocol that is based on XML format. Even Nevertheless, the majority of businesses are now switching to REST services, which typically use JSON (JavaScript Object Notation) as the content of the message payload data type. Furthermore, while white-box testing is a focus of several articles, these techniques are not always practical because the source code for APIs is not always accessible. Last but not least, as far as we are currently aware, no prior work has made an effort to use artificial intelligence techniques (apart from search-based) in the context of API testing (Martin-Lopez, 2020). Due to complexity and a lack of adequate documentation, software developers typically undertake integration testing more frequently than testers. But as the industry shifts to more flexible designs rather than heavy monolithic architectures, the task is becoming more and more difficult as the number of APIs utilized in a system increases more rapidly. For instance, if a system has more than 50 APIs, how is the tester or developer going to test it? Although there are a lot of automation tools available, scripting and maintenance still need a lot of human labor. This gets tougher and harder as the population increases.

This research project aims to overcome all the aforementioned problems, achieving a level of automation and accuracy for REST APIs that has never been achieved before. The primary goal of this research is to examine prior literature in order to identify any gaps and crucial future improvements for API testing and automation. Then, collect information from 100 industry professionals on the challenges they are currently facing and the kind of technology they expect to be used for API testing. Finally, a testing framework for the automatic development of complicated test cases for APIs is planned, based on the issues mentioned in the literature and survey. The goal of this research is to use AI to automatically detect software flaws. In particular, the goal is to create a concept based on image processing and intelligent software referred to as "bots" that can produce a large number of test inputs—perhaps millions—at a time.

1.1 Research Objectives and Questions

The objective of this research is to discover an approach for enhancing API testing using cutting-edge AI technology concepts. Teams working on software development and organizations of all stripes will find this material useful. By different sorts mean that there are different kinds and sizes of organizations. These organizations come in different sizes, and the software and system architecture is either small-scale or large-scale. The findings from this research will provide answers to the following queries:

1. What testing and automation options are currently available for APIs?
2. What challenges do API testing and automation currently face?
3. How can Artificial Intelligence be used to enhance API testing?

The objectives that follow will be achieved by this research:

1. Find out what API testing solutions are currently known about via research or literature?
2. Learn about the practical difficulties with API testing and automation.
3. Propose an AI-driven API testing concept.

The results associated with the aforementioned findings will provide knowledge to the groups and disciplines related to testing and quality assurance, as well as to those who govern the research on software development and instruct software development organizations on testing and development methodologies.

2 Related research

This chapter gives an overview about web APIs testing, automation and existing AI related and non-AI related solutions.

2.1 Web APIs

Web APIs are a crucial interconnection tool for Internet-based software service access. Applications with a higher degree of interconnection can offer their users greater services for less money. For instance, using the Google Maps API, a business search portal can show users the nearby establishments on a map (Sohan et al., 2015). Web apps and APIs, which seem to be favoured over traditional Web services that utilize WSDL and SOAP, are becoming more and more prevalent in all kinds of services on the Web (Maleshkova et al., 2010). Undoubtedly, web services will keep playing a significant role in the growth of loosely linked component-based systems both within and across businesses (Maleshkova et al., 2010). But the true query is: What is a web API, exactly? A web API (Application Programming Interface) is an application programming interface for the Web. It is an independent program, to put it simply. Despite the advanced languages or technology, there are no restrictions on access across a wide range of applications or platforms. It is something that, in a more generic sense, is used to integrate several applications for greater functions. For instance, utilizing the Google map APIs would suffice if an application wants to add a map or location instead of writing extensive code. According to the Maleshkova et al.(2010), there are three types of APIs: RPC-style, RESTful and Hybrid. But they identified RESTful (REpresentational State Transfer) services as web APIs.

2.2 Web API Testing and Automation

With the quick uptake of REST, numerous software businesses are making their applications available as RESTful web APIs, while client code developers are incorporating these APIs into their programs. Web APIs vary over time, thus developers must adapt their applications to reflect those changes. On the other hand, developers frequently run into difficulties during the process of migration, and API providers are generally unaware of how client developers respond to API changes (Wang et al., 2014). Many popular platforms, including Facebook, Google Maps, and Twitter, have their own procedures to address this issue. For instance, because Facebook does not utilize a formal versioning system, it takes a very different approach to the creation/modification of web APIs. Instead, the release of newer features is accomplished through a process known as "migrations," which entails small modifications to the API that every developer is free to enable or disable at their discretion during the roll-in phase. After this time, all clients will have access to the improvements permanently (Espinha et al., 2014). However, this is still a problem because updating code and delivering a product need a variety of processes, including automation, system testing, and integration testing, which can take a lot of time and go over the allotted time period. The tesing is among these components' most challenging elements because it has numerous sides. The digital revolution of enterprise applications throughout hybrid Cloud and edge settings is being driven by the web API economy. To be succed with such trasofrmarions, end-to-end testing of the application and API constitution is needed (Ackerman et al., 2021). However, it can be difficult to verify if Web APIs are correct. A test engineer often also requires setup the appropriate data into the data sources and maybe mock interconnections with other external services in addition to creating messages through the network (using HTTP over TCP, for example) against the API (Golmohammadi et al., 2022). According to the (BANGARE et al., 2012), there are two factors which demonstrate that testing APIs differs from testing GUIs. One of them is that it necessitates internal knowledge, therefore the tester must be fully aware of the logical reasoning behind each encounter. For instance, in order to make a REST POST request to create a patient using an API, precise knowledge of the payload's construction and the type of data required is required. Another factor is the availability of the source code, which implies that having access to it will enable testers to comprehend and evaluate the implementation mechanism employed, conduct appropriate tests, and narrow the scope of the tests. Due to these factors, API testing was originally handled by developers, but as the number of APIs increased, the task got more difficult. Even while there were competent testers, the large number made it difficult for them as well. In order to reduce this risk, several software companies chose to use API automation. Automation plays a crucial role in API testing, enabling the execution of test cases and scenarios efficiently and accurately (Jones, 2019). Automated API testing boosts test coverage, decreases manual effort needed for repetitive operations, and aids in finding bugs early in the development cycle (Brown, 2018). As a result, findings are consistent and there is a lower likelihood of human error. It enables engineers to conduct tests frequently. Depending on the specific needs of the project, a variety of strategies can be used when it comes to API testing (White, 2020). Testing an API's functioning through individual functions and methods is the main goal of functional API testing (Johnson, 2018). On the other hand, unit API testing is about testing individual units of code to make sure they work properly within the API (Smith, 2019). Stress and load testing In order to assess the performance and stability of the API, it is put through intense loads and stress conditions (Brown, 2018). Finally, security API testing seeks to identify API flaws and guarantee the security of sensitive data (Jones, 2019). There are numerous solutions on the market that can help with API testing and automation (Garcia, 2019). Popular API testing frameworks like Postman, SoapUI, and REST Assured give programmers and testers the tools they need to efficiently plan, carry out, and manage API tests (White, 2020). When choosing an API testing tool, it is important to take into account aspects like usability, compatibility with various programming languages, and support for automation (Johnson, 2018). There are several important steps that must be taken while automating API tests (Smith, 2019). Finding the test scenarios and getting the appropriate test data are the first steps in test case design and planning (Brown, 2018). Afterwards, configuring the necessary infrastructure and dependencies is part of setting up the test environment (Jones, 2019). The ability to execute test cases without human interaction is made possible by writing automated test scripts (White, 2020). The process of executing automated tests and evaluating the results is known as test execution and result analysis (Garcia, 2019). Finally, recording test findings and keeping track of any faults found are part of reporting and defect management (Johnson, 2018). A number of best practices should be followed to guarantee successful API testing and automation. A reliable and stable testing environment is created by properly setting up and maintaining the test environment (Jones, 2019). Maintainability of test code can be enhanced by scripting design patterns such as Page Object Model or Behavior-Driven Development (White, 2020). Managing authentication and authorisation makes sure APIs are securely protected (Garcia, 2019). Continuous Integration and Delivery (CI/CD) pipelines that incorporate API testing enable to quickly identify problems and produce high-quality software (Johnson, 2018). Despite the advantages of API testing and automation, there are some issues that must be resolved (Brown, 2018). The generation of test data can be difficult, needing special tools or methods to produce realistic and varied data (Smith, 2019). It might be difficult to manage issues when handling dependencies and testing intricate scenarios (Jones, 2019). It's imperative to monitor and troubleshoot when running tests to quickly find and fix problems (White, 2020). When testing APIs that depend on external systems or services, therefore the test environment synchronization is crucial (Garcia, 2019). Finally, to address potential vulnerabilities, security testing should be included in API testing activities (Johnson, 2018). In summary, API testing and automation are essential elements of contemporary software development. Developers may produce high-quality software that satisfies user expectations by ensuring the functionality, reliability, efficiency, and security of APIs (Brown, 2018). API testing and automation may speed up the development process and increase the delivery of reliable and secure software with the appropriate tools, best practices, and techniques (Smith, 2019).

2.3 What is AI and bots?

Artificial intelligence or AI is known as AI. The goal of this area of computer science is to build intelligent machines that can carry out tasks that ordinarily call for human intelligence. Speech recognition, decision-making, problem-solving, learning, and other activities may be included in these tasks. AI, to put it simply, is the process of developing computer systems that can mimic human intellect and carry out activities on their own. It's a fascinating topic that is always changing and has the potential to completely transform a number of different sectors. Today's Artificial Intelligence (AI) revolution is flattening the world by enabling machines to carry out cognitive tasks like sensing, thinking, learning, and interacting (Ergen, 2019). artificial intelligence is another, powerful technological wave, but one that is still emerging (Ergen, 2019). Ergen (2019) asserts that AI is comparable to the wave In the 1980s, when computing power became incredibly cheap and accessible. Similar to how it makes predictions cheap and accessible, AI will quickly automate the repetitive and repeatable works through robots. According to (Du-Harpur et al., 2020), Artificial intelligence, or AI, is the ability of an automated system to communicate, reason, and act independently in both known and undiscovered scenarios in a similar way to a human. Major technological corporations like Apple, Google, and Amazon are highlighting artificial intelligence (AI) extensively in their product introductions and acquiring startups with an AI focus (Agrawal et al., 2018). It all stems from the results it produces: cost savings, ease of daily tasks, and the ability to perform some human tasks without intervention of a human. Another well-known and widely used element of artificial intelligence is bots. They are a long-used software engineering idea, namely in the gaming industry, rather than something that just appeared with AI. Applications known as software engineering bots can respond to external stimuli, such as events generated by tools and messages submitted by users, and carry out automated activities in response. They serve as a conduit between consumers and services (Shihab et al., 2022). Chatbots are also one category of the bots and according to Nawaz & Gomes (2020), they are higly used in recruitments. Chatbots are a unique setup that can unquestionably increase the effectiveness of the review step of recruitment industry. The right candidates can be distinguished from the unsuitable by sending a text message to every prospective application that asks a series of brief, pre-written questions using AI powered chatbots. Not only chatbots there are many other bot categories which are help to impove things. One such a bot type is DevBots. DevBots are already widely used in open source software and business, and interest in using bots in software engineering is growing (Erlenhov et al., 2020). Bots are becoming widely adopted in a number of industries, including e-commerce, customer support, and education. There is no exception in software development (Shihab et al., 2022). These AI-powered bots and software are built using various machine learning algorithms. These algorithms can be divided into four types: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning (Pugliese et al., 2021). Supervised learning uses assignments to teach a function that maps an input to an output based on sampled inputs and outputs. As a result, the learning procedure is based on evaluating the calculated output and projected output, that is, learning refers to computing the error and modifying the error to achieve the expected output (Pugliese et al., 2021). This technique includes algorithms like Nave Bayes classification, linear regression, logistic regression, and support vector machines (Pugliese et al., 2021). An example of supervised learning is automatic message responding and face recognition. Unsupervised learning analyses data sets that are unlabeled without the intervention of humans. Without accompanying labels, the algorithm in unsupervised learning efficiently classifies samples into distinct groups based on the properties of the training data itself (Pugliese et al., 2021). K-means clustering, autoencoders, and principal component analysis are examples of unsupervised algorithms (Pugliese et al., 2021). The most popular example of of this technique is the automatic recognition of friends of a user in social networking channels such as Facebook. Reinforcement learning is based on a set of algorithms that typically operated progressively to autonomously assess the best behavior in a given environment in order to increase its efficiency (Pugliese et al., 2021). At every stage, a reinforcement algorithm, also known as an agent acts and anticipates future traits based on past and present attributes, and a reward or penalty is applied based on the prediction made (Pugliese et al., 2021). As a result, it is an effective tool for developing AI models that can improve the operative efficiency of complex systems such as robots, self-driving automobile activities, manufacturing as well as supply chains (Pugliese et al., 2021). Any application that is powered by these machine learning techniques and algorithms can provide the end user a novel experience and some inteligent features.

2.4 AI and Web API

Software testing is not an exception to how AI is transforming numerous facets of software development as technology improves. The software testing process is made more efficient, automated, and subject to intelligent analysis thanks to AI. The creation of automated test cases is one method AI replaces testing. AI algorithms are able to examine the system's code and behaviour, comprehend it, and automatically produce test cases based on numerous scenarios and edge cases. Because less manual labour is needed to create test cases, testers can save time and effort. Enhancing test coverage is another important function of AI. AI algorithms can pinpoint system components that require more thorough testing by evaluating vast volumes of data and looking for patterns. They can also spot any flaws or performance hiccups that conventional testing techniques might miss. This promotes thorough testing and higher general standards of quality. As a result, a large number of software businesses are striving to develop AI-powered test automation tools, and these technologies are generating what some perceive to be much-needed industry acceleration in testing advancements (King et al., 2019).

Various enhancements and suggestions are provided in a few research publications, especially for GUI level test automation employing AI technology. Numerous of them have been tested using real-world settings, and the outcomes are substantially better than expected. The table below provides a summary of the GUI testing solutions each article offers.

Authors	**Solution(s)**	**Results**
Gao et al. (2022)	- GUI element identification and classification using the models YOLOv5 and Efficient-Det0. - Classes were constructed to categorize the relationship types between the GUI elements and nodes (nodes means if the element was AI element - AI node or non-AI element - Non-AI node). - The generation of test scenarios for the mockup diagrams using plotly library and NetworkX library.	- A 75% precise score indicates that all of the GUI elements can be accurately detected. - A 79% of relationships correctly identified by the model. - Ability to properly anticipate the complete test scenarios set for a comprehensive AI application.
Walia (2022)	A framework to generate automation scripts using convolution Neural networks (CNN) with following steps. - Resolution specifications are transformed into a collection of test cases. - Creation of a video recording from the collection of test cases found earlier. - Automated test script creation from the above video recording.	Confirmation/Validation accuracy of 94% with max training accuracy of 95%.
Eskonen et al.(2020)	An image based DRL technique for exploratory GUI testing	The test ran more quickly than an experienced tester would have, and it led to the discovery of numerous bugs that traditional automation cannot pick up.

Table 1: An overview of GUI test automation solutions

The previously mentioned three approaches are designed primarily to improve GUI testing and handle three different problems: test scenario generations, script generation, and autonomous exploratory testing. Most of them have proven to be capable of greatly enhancing the effectiveness of the software testing process. API testing is not an exception as the technology evolves many are thinking to automate API testing with AI. When it comes to APIs, the most commonly used type is REST APIs, and testing is becoming more difficult as the number of APIs used for a system grows with new polylithic architecture patterns. There are various solutions provided by researchers about API testing and automation, but all of them have flaws, and in recent years, many people have begun to consider incorporating AI to accelerate API testing.

Reza and Van Gilst (2010) identified various issues with REST API testing more than a decade ago, but the most significant one is testing it without involving external or third-party consumers. The development of stubs to emulate other web services is a time-consuming and error-prone procedure that can require substantial amounts of developer work (Reza & Van Gilst, 2010). As a result, they decided it would be better to have a framework that can be used to create test stubs for each of the predicted web services used in the bigger system. According to the framework proposed by Reza and Van Gilst (2010), they employ XMLs to specify what needs to be tested and what to do. They named it input specification, and it defines the service type and related parameters. Assume there is a parameter named latitude, and its value should be between 90 and 95, and it can be a float number. As a result, this XML specification defines it as well as regular expression for documenting all test scenarios linked to this parameter. Their next step was to generate test output, which they accomplished using two approaches. The first is the usage of a CSV file to store data (data centric), and the second is the generation of test data using a program logic. Finally, the output is an XML template containing all test cases, which greatly assists both testers and developers in carrying out tests without having to spend a significant amount of effort creating test cases and data. Also, long-running stress tests on a software program are possible without having to define a large number of data points, which can be very useful when testing AJAX applications where the browser and application memory management anomalies occasionally cause the behaviour of long-running applications to be somewhat unforeseeable. Even though this has been developed without the help of AI, the concept is still better input to build a solution with AI algorithms. In 2018 Ed-douibi et al. presented another solution which is totally based on the open API specification (Formerly known as swagger). API testing based on specifications is the validation of an API using the available features provided in a specification document (Ed-douibi et al., 2018). In specifications-based REST API testing, test cases include sending HTTP/S queries and confirming that server replies correspond to the specification, like the OpenAPI one (swagger) (Ed-douibi et al., 2018). The generation of such test cases is part of automated specification-based API testing. Following figure 01 describe of their approach to generate test cases using a specification.

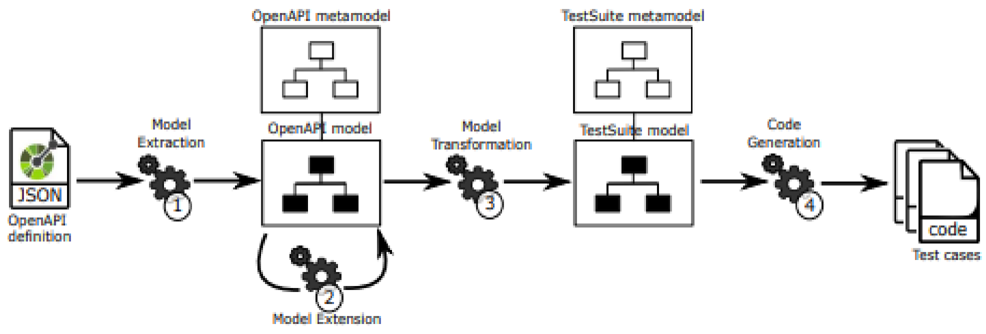

Figure 1: Approach by Ed-douibi et al. (2018)

This solution's ultimate result is a set of scripted test cases based on the specifications. This is a very solid strategy; however, the question is if we can rely just on the specification because additional elements such as network, protocol, and authentication affect API testing.

The following solution is the well-known EvoMaster by Andrea (2019), whose basic concept was automated test case generation. This is a technique/algorythm for automatically generating test cases for RESTful APIs. EvoMaster is an intelligent tool that generates these test cases using evolutionary methods. RESTful APIs are commonly used to enable communication between various systems or applications. It is critical to test these APIs to assure their functioning and dependability. Manually generating test cases for APIs, on the other hand, can be expensive and prone to human mistakes. EvoMaster comes into play here. EvoMaster optimizes the test case-generating process via evolutionary algorithms. It begins with a set of basic test cases and expands them iteratively using techniques such as genetic algorithms. These algorithms use natural evolution principles like as crossovers and mutations to evolve test cases toward greater coverage and effectiveness. EvoMaster saves software developers and software testers time and effort by producing test cases automatically. It contributes to ensuring that the API works well, handles varied inputs accurately, and provides the desired results. EvoMaster additionally improves test coverage by providing a wide range of test cases that cover a variety of scenarios and edge cases. Software testing is able to be viewed as an optimization issue in which the goal is to optimize the code coverage and fault identification of the produced test suites. After defining a fitness function for a given testing issue, a search algorithm can be used to search the space of every potential solution (in terms of test case context) (Andrea, 2019). There are various types of search algorithms, the most well-known of which are Genetic Algorithms. Genetic Algorithms are optimization algorithms that are inspired by natural selection and genetics processes. They are used to tackle complicated problems by replicating the theory of evolution, selection, crossovers, and mutation processes. Every individual represents a potential solution to the problem. The program then assesses every individual's fitness or quality based on a predefined objective function. A genetic algorithm's central concept is to continuously evolve its population over generations. There are several specialized search algorithms for generating test suites, including the Whole Test Suite, MOSA, and MIO. All of which are based on genetic algorithms. The alogrythm used by EvoMaster is MIO (Many Independent Objective) which is a search method designed for the generation of test suites for integration and system testing. This test case generation depend on three elements: problem representation, search operators and fitness function. This tool was implemented by giving relevant information to the three elements listed above using the Swagger schema. They also used Oracle to add assertions to each created test case, which saved a software developer a lot of time. In brief, this is a white box methodology written in Kotlin and tested against a variety of Java projects. The best feature of this tool is that it is completely automated, which means that a developer can select any class and build a full test suite that covers 80% of code lines. Although this is not an AI-powered tool, the algorithm, and techniques provided excellent feedback for any future API products.

Martin-Lopez et al. (2022) emphasize the importance of thorough API testing in order to assure quality, reliability, and compatibility and online testing relevance to API. Online testing is the practice of testing APIs in a live, changing setting that simulates real-world usage scenarios. The primary benefit of online testing is being able to capture the dynamic nature of API interactions and discover faults that typical offline testing may miss. There are numerous challenges associated with API online testing. One significant difficulty is a lack of control over backend systems, as APIs frequently rely on external services and databases. This makes it difficult to isolate and reproduce individual test scenarios, perhaps resulting in inconsistent test findings. Furthermore, APIs frequently evolve over time, producing changes that can affect their behavior and compatibility. To accurately capture these changes, APIs must be tested in real-time. To solve these issues, the authors (Martin-Lopez et al., 2022) propose a number of testing techniques. One method is to generate test cases automatically. This entails automatically constructing test cases based on specifications such as API documentation or OpenAPI descriptions. Automated generation ensures that a wide range of scenarios are evaluated by providing extensive coverage of various API endpoints and their parameters. API fuzzing is another technique that has been discussed. Fuzzing is the process of submitting random or faulty inputs to APIs in order to discover vulnerabilities or unforeseen behavior. Fuzzing can be an effective method for detecting security problems such as injection attacks or buffer overflows. The authors emphasize the need of using fuzzing techniques to evaluate the robustness and security of RESTful APIs. Martin-Lopez et al. (2022) also discusses property-based testing as a useful technique for testing RESTful APIs. Property-based testing entails establishing properties or invariants for the API that should be true under multiple scenarios. The testing framework then produces random inputs and determines if these properties are met. Property-based testing can provide a complete and systematic approach to evaluating RESTful APIs by defining characteristics that encapsulate the desired behavior of an API. In addition to above-mentioned techniques, the authors emphasize the significance of considering performance, security, and scalability during online testing. It is difficult since API behavior can be influenced by factors such as network latency, load balancing, and caching.

When compared to the aforementioned challengers, the previously described solutions have numerous issues that must be addressed in order to be competitive in today's market. Then we need a solution that is both dynamic and logical. The answer could be AI, but the next big question is how it will support APIs testing. Mirabella et al. (2021) explore the fascinating field of deep learning and how it can be used to predict the validity of test inputs for RESTful APIs. The researchers begin by emphasizing the necessity of guaranteeing the quality of test inputs, as this has a direct impact on the dependability and efficiency of RESTful APIs. Identifying and confirming acceptable inputs for testing has traditionally been an exhausting and time-consuming task. However, Mirabella and the others present a revolutionary strategy that uses deep learning to automate this process. Their method entails training a deep learning model with massive amounts of historical data. This data contains information on prior testing inputs and their validity outcomes.

The researchers enabled their model to learn patterns and correlations by feeding it this data, thereby making it an expert in judging the validity of novel test inputs. Their experiments yielded genuinely astounding outcomes. The deep learning model predicted the legitimacy of test inputs for RESTful APIs with outstanding accuracy. This discovery has the potential to change the way that developers approach testing and validation, preserving their time and effort. However, the researchers did not stop there. They also studied the effect of several factors on the proposed model's performance. They got useful insights into refining the model's accuracy and efficiency by rigorously assessing variables such as input complexity, data size, and noise levels. These findings provide developers with realistic suggestions for implementing deep learning-based prediction systems for RESTful API testing. This solution is far better than previous solutions but yet it does not address all parts of API testing.

Martin-Lopez (2020) presented new proposal about AI-driven web API testing. This researcher discovered that creating test cases alone based on the API specification is typically infeasible, as it lacks essential information such as authorization data such as API keys. As a result, when it comes to completely covered tests, specification-based approaches are out. Therfore what best is model based testing techniques for APIs. Martin-Lopez and his team used model based approach for test case generation and verify them with oracle and then they proposed how AI comes in to play with API testing. According to the proposal, API requests and responses are being monitored with the goal of identifying two major patterns: invariants and metamorphic relations. Invariants in API request-response interactions are consistent patterns, such as "if parameter X is used in a specific operation, the response will always contain property Y." These patterns are consistent over several API calls. Metamorphic relations, on the other hand, involve relationships between two or more API requests and their associated responses. "If we invoke a specific operation and then invoke the same operation with parameter X, the result of the second call must be a subset of the first," for example. Metamorphic relationships aid in the identification and validation of expected changes in API behavior. The recognition of these patterns should aid in the detection of previously unknown bugs. They advocated using software bots to monitor the scenarios. The bots' knowledge is used to automate manual operations, notably fine-tuning the test configuration file and automating certain portions of test case generation. One example is the ability of bots to automatically deduce inter-parameter relationships by analyzing real-world requests to the service. Bots can deduce these dependencies without manual intervention by examining successful and erroneous calls and their underlying causes. Furthermore, meaningful and realistic values for variables/parameters could be derived by parsing their natural language descriptions (derived from the API documentation) or by modifying original values from prior executions. Martin-Lopez's (2020) proposal is outstanding however the prototype is still in the works, but it covers many areas of API testing.

In commercial level ParaSoft is one of pioneer to talk about AI-driven API testing tools at 2018. On October 2nd, 2018, an exciting announcement about the latest edition of Parasoft SOAtest was issued. This release includes revolutionary machine learning techniques that improve the Smart API Test Generator. Testers may now easily construct meaningful API test scenarios with the help of AI and ML, making the testing process more efficient and effective with this tool. SOAtest's release elevates the test creation process to a whole new level. Users can now apply machine learning to train SOAtest's artificial intelligence engine using their organization's existing test inventories. This means that the AI engine can develop useful API test templates automatically, saving time and effort. Organizations may use this functionality to quickly develop a solid foundation of tests, extend API coverage, and scale their API testing process. This is seems a good solution but still did not attract many customers.

3 Research Methods

The purpose of this research is to examine reality and develop an API testing concept. As a result, the core nature is inductive and does not aim for hypothesis testing. As a consequence, the study can be summarized as an induction study that analyzes current literature and collects data from specialists in order to develop a concept. Induction is the process of going from the specific to the general, such as when constructing empirical observations concerning a phenomenon of interest and developing conceptions and theories on the basis of them (Locke, 2007). Induction is typically a qualitative approach, but there are no rules or definitions governing its application, and it can also be quantitative. The primary goal of this research is to determine what existing solutions exist and what type of requirements real professionals expect from AI-driven testing and automation. Finally, depending on the data and litreature review, outline a concept. As a result, the majority of the data in this study are qualitative.

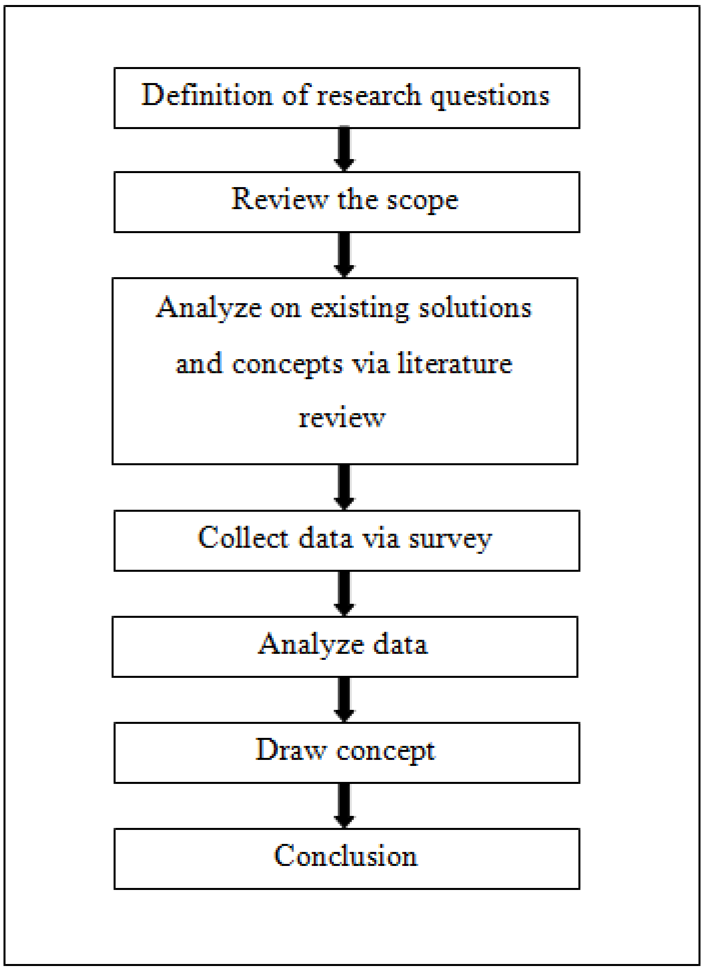
Definition of research questions
Review the scope
Analyze on existing solutions and concepts via literature review
Collect data via survey
Analyze data
Draw concept
Conclusion

Figure 2: Study process

Above figure 2 gives the overview of this research conducting process. Each stage is described in greater detail below.

- Define research questions: outline the questions that this research study will answer.
- Review scope: Evaluate the study questions and scope in relation to the time frame and feasibility. Redefine the scope.
- Analyze on existing solutions and concepts via literature review:
- Collect data via survey: A structured questionnaire will use to gather required information and target is 100 software professionals.
- Analyze data: This phase will begin once the data has been collected. This comprises data categorization and summarization. Both open and closed ended questions will be used to explain and analyze what AI-driven solutions are already available for the QA community and what is lacking.

- Draw concept: Draw or design AI-based solution based on the analyzed data and articles.
- Conclusion: This is the end of the research when it is defined what the specific outcomes of this study are and what improvements and research may be done using the discovered results.

3.1 Survey

The survey asked 18 questions to learn about respondents' diversifications and other related characteristics. There are nine open-ended questions and nine closed-ended questions. Appendix 1 has further information.

4 Findings of literature review

AI-powered API testing solutions have gotten a lot of interest recently because of their promise to improve the efficiency and efficacy of software testing processes. Several major results in this sector have emerged from the study of the literature. To begin, one of the primary benefits of AI-powered API testing solutions is their ability to automate the development of test cases. Traditional testing approaches frequently rely on manual test case creation, which can be time-consuming and error-prone. AI-powered solutions, on the other hand, use machine learning algorithms to autonomously review API specifications and build relevant test cases, saving substantial amounts of time and effort. Secondly, research shows that AI-driven API testing solutions can efficiently handle complicated scenarios and edge cases. These systems can intelligently discover potential flaws and vulnerabilities that manual testing may miss. This contributes to the API's robustness and dependability under varying settings. Furthermore, research underlines the importance of AI-driven solutions in lowering the maintenance effort necessary for API testing. Traditional testing methodologies necessitate regular manual modifications to test cases as APIs expand and change. AI-driven solutions, on the other hand, can react to API changes by exploiting their learning skills to automatically alter test cases, lowering the cost of maintenance. Another important discovery in the literature is the ability of AI-driven solutions to improve test coverage. These technologies can reveal sections of the API that manual testing may have overlooked by analyzing vast volumes of data and trends. This increased coverage aids in the detection of potential bugs and performance concerns, resulting in more reliable and resilient APIs. There aren't many literatures to support AI-driven API testing or automation solutions, according to the papers examined in this thesis. However, certain solutions mainly based on specifications produce good results.

In recent times, we've witnessed notable advancements in AI-driven solutions for GUI-based testing and automation. However, there seems to be a lack of similar solutions for APIs. While a few options exist, many of them rely heavily on specifications, making it challenging to capture edge cases effectively. One standout solution come across during the study is EvoMaster, which is a highly regarded non-AI solution. However, it operates as a code-based (whitebox) solution and is therefore not suitable for web APIs. In a recent study by Mirabella et al. (2021), they presented an intriguing solution focused on utilizing deep learning for the input validation of APIs. However, it's important to note that while their approach is highly effective for input validation, it may not cover all testing areas comprehensively. The test scope covered by this solution is primarily aligned with the scope covered by specification-based solutions, which are widely used in API testing. In the context of API testing, there are crucial aspects that go beyond just sending a request and validating the response. One of the key areas where we may be missing out is the identification of patterns and different authorization handling. API testing encompasses more than just verifying the accuracy of responses. It involves delving into various other factors such as security and performance. Ensuring that APIs are secure and properly handle authorization is of utmost importance to protect sensitive data and prevent unauthorized access. Identifying patterns within API behavior can greatly enhance testing efforts. By recognizing and understanding recurring patterns, we can create more robust test cases that cover a wider range of scenarios, potentially uncovering hidden bugs or vulnerabilities. Additionally, performance testing plays a vital role in determining how APIs handle varying loads and stress. Assessing response times, scalability, and overall performance under different conditions gives us valuable insights into the API's capabilities and helps identify potential bottlenecks or performance issues. Martin-Lopez (2020) put forth an impressive solution that addresses several of the factors mentioned earlier in API testing. His approach introduces a novel component to the test framework by utilizing bots to learn patterns. This innovative concept shows promise in enhancing the effectiveness and efficiency of API testing. Although Martin-Lopez's solution is commendable, it's worth noting that the results of its implementation are currently unknown. Further research and experimentation are required to evaluate its performance and reliability in real-world scenarios. When it comes to commercial-level solutions in this domain, the landscape is somewhat limited. However, Parasoft stands out as a provider that offers a solution in the API testing space. Nevertheless, the extent to which Parasoft's solution addresses the aforementioned factors and its overall capabilities remain uncertain at this point. As the field of API testing continues to evolve, it is essential to keep exploring innovative approaches and assessing their effectiveness

5 Survey Results

This study employed a structured questionnaire and distributed it across online platforms to collect data on what the QA community truly expects from an AI-driven API test tool. A survey was distributed to 120 professionals, and 48 responses were obtained. The primary demographic for this study was QA professionals with API testing experience. The questionnaire has 19 questions, including open-ended and closed-ended questions. The first three questions were intended to determine each professional's level of familiarity with API testing and automation.

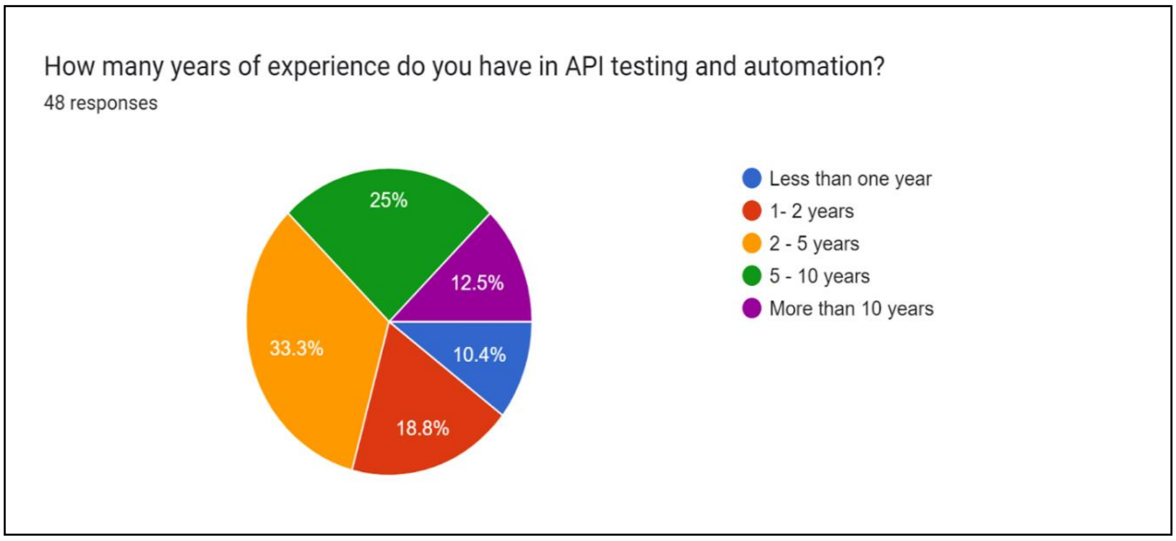

Figure 3: Respondents' API testing level experience

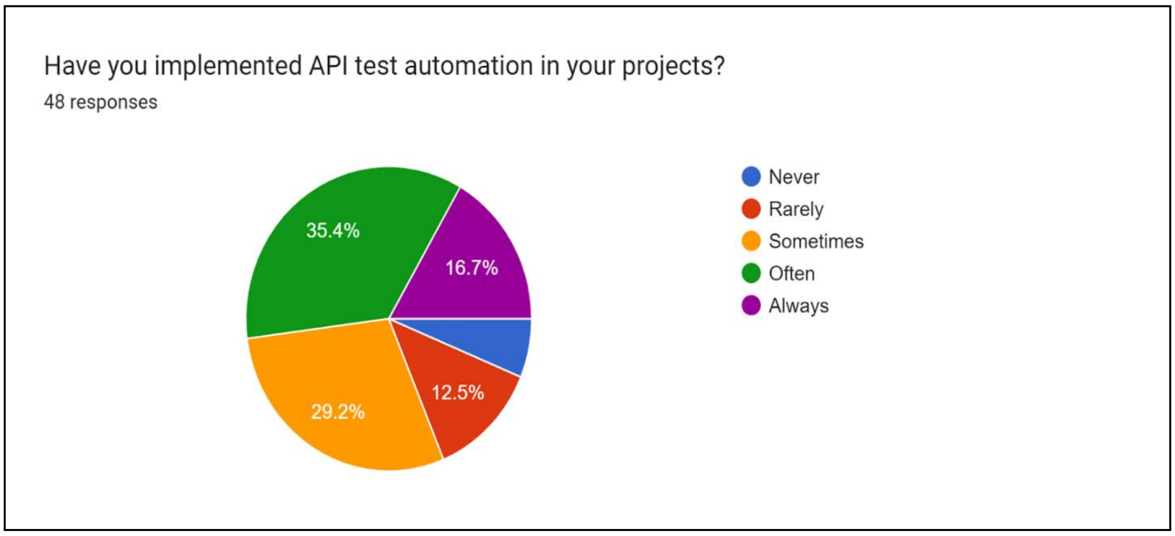

Figure 4: Respondents' API automation level experience

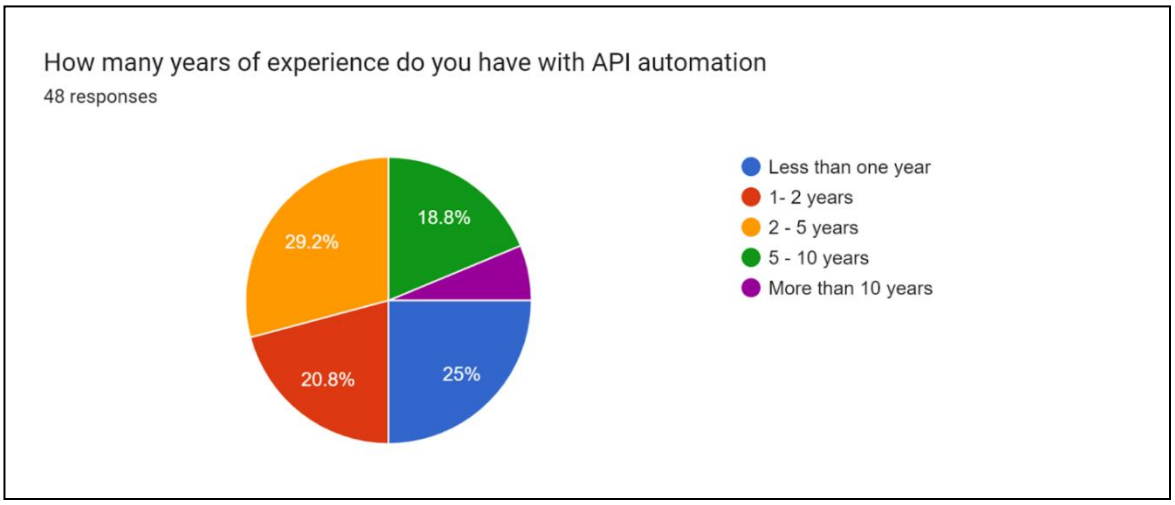

Figure 5: Respondents' API automation level experience in years

According to the responses gathered, the majority of the group has 2-5 years of expertise in API testing and automation (33.3%), while 25% claim to have experience for 5 - 10 years. Overall, respondents are dispersed throughout all groups, with respondents with less than one year of experience having the least amount of experience (10.4%). In terms of API automation experience, 35.4% said they use it frequently, while 29.2% said they only use it occasionally. Except for 6.3% (3 respondents), all respondents have automation experience. People with automation experience range from 2 to 5 years, with 25% having less than one year of experience and 20.8% having 1-2 years of experience. Only 6.3% of respondents claimed to have more than 5 years of experience. It appears that automation experience is less than overall API testing experience. In summary, all respondents have experience with API automation, with the majority having more than 2 years of experience, but when it comes to automation, only 45 have experience, with the majority having less than 5 years of experience. Overall, the respondents have extensive experience with API testing and automation knowledge.

Following that, this survey focuses on how respondents see the importance of API automation to a project or product.

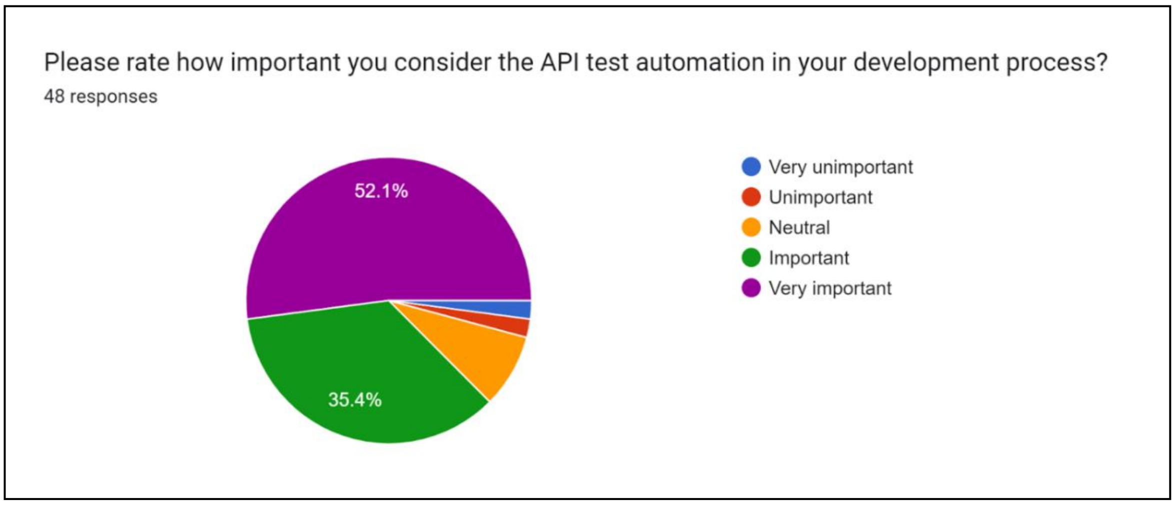

Figure 6: Respondents'' ratings about the importance of API automation

As anticipated, 52.1% said it is extremely important, while 35.4% said it is important. Only two respondents said it was unimportant, and the vast majority feels that API test automation is a critical responsibility for any project.

The questionnaire was then arranged to learn what tools these respondents utilized for testing and automation.

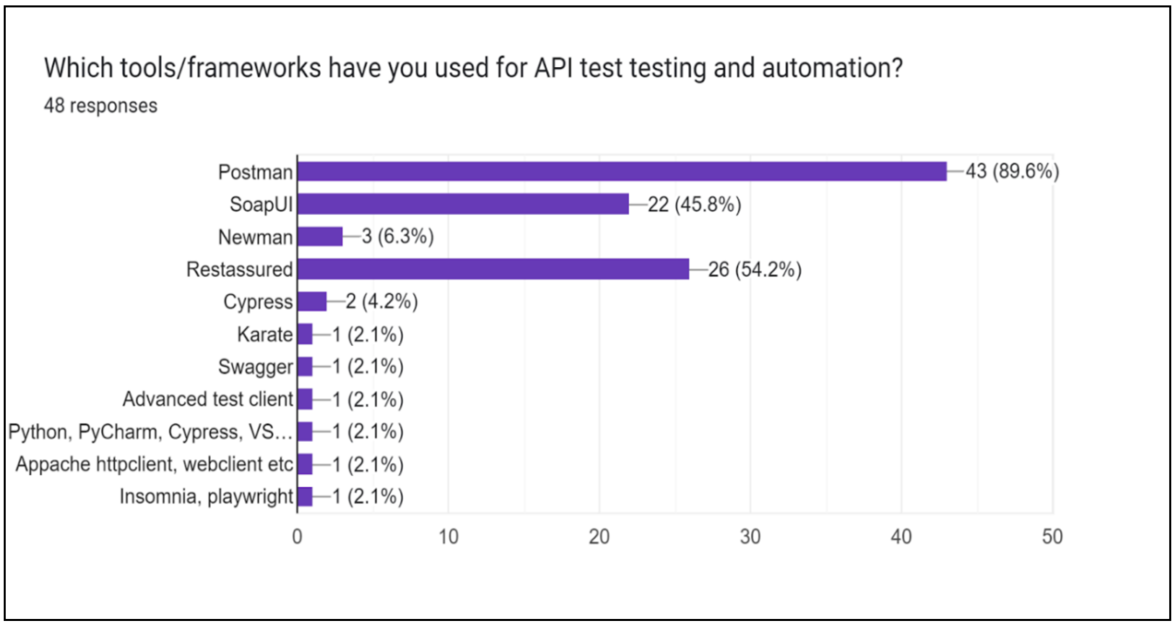

Figure 7: Tools/Frameworks used for API testing and automation

According to the data, the most popular tool is Postman, which was mentioned by 89.6% of respondents, while 54.2% stated that Rest assured is used for automation. SoapUI came in third place with 45.8% of the voting. Respondents also highlighted Newman, Cypress, Karate, Swagger, Advanced test client, Python, Pycharm, Visual studio, Appache httpclient, webclient, Insomnia, and playwright in addition to these tools.

The following question concerned the kind of challenges that these respondents encountered while working with these tools and frameworks.

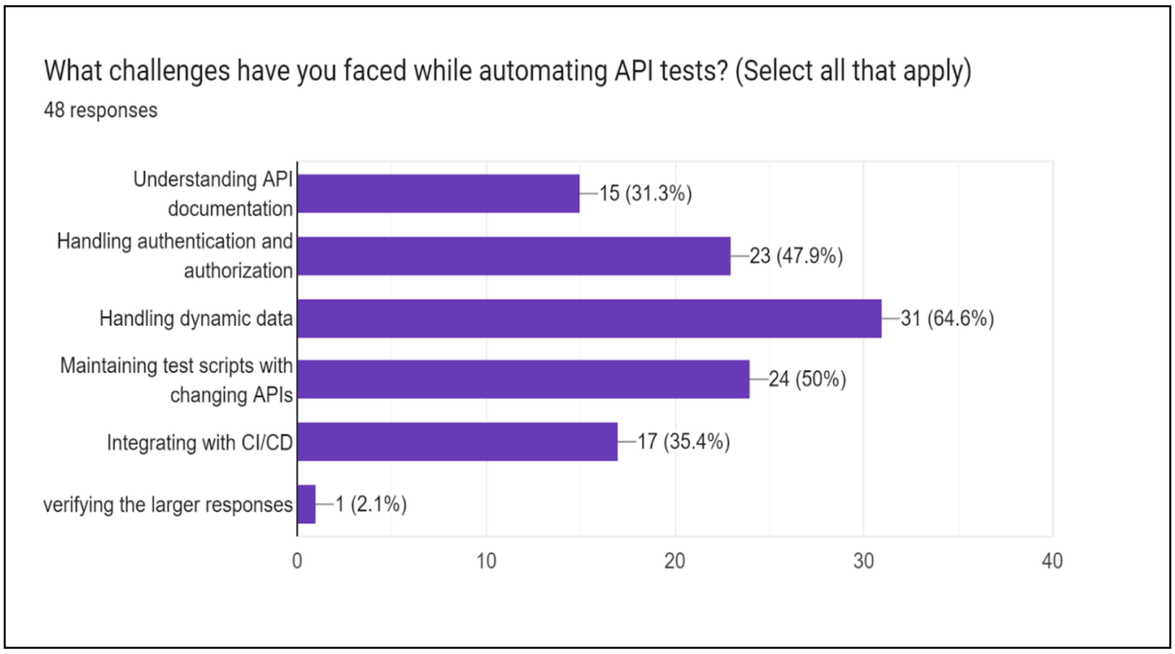

Figure 8: Challengers experienced

This is one of the most crucial questions raised in order to determine what challenges the QA community is currently experiencing with API testing. 64.6% The most pressing issue, according to the replies, is handling dynamic data. This is also supported by the findings of the literature. Because APIs typically deal with dynamic data, which is sometimes only available during runtime. In such circumstances, designing or automating test cases requires a significant amount of work and raises concerns about dependability, particularly if it is dependent on dynamic data such as IDs generated during runtime. The table below despites the rankings of each of the community's challenges.

Challenge	**Ratings**
Handling dynamic data	64.6%
Maintaining test scripts withchanging APIs	50%
Handling authentication andauthorization	47.9%
Integrating with CI/CD	35.4%
Understanding APIdocumentation	31.3%

verifying the larger responses	2.1%

Table 2: Ranking of the challenges faced during API automation

The final attribute in table 2 differs from the others since it was mentioned by the respondents, whilst the others were predefined in the survey. As a result, it cannot be calculated as the lowest rating challenge, but rather as another crucial component.

The study then used the remaining questions to determine the respondents' AI awareness.

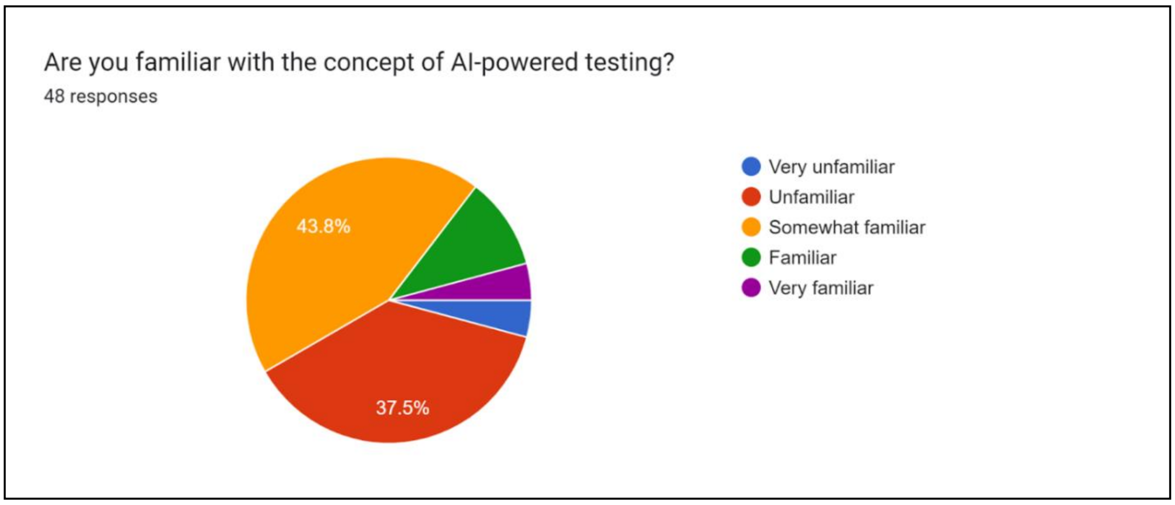

Figure 9: Respondents'' awareness of the AI concepts

Figure 9 depicts respondents' awareness of AI concepts, with 43.8% saying they are somewhat familiar and 37.5% saying they are unfamiliar. Only 14.6% of those surveyed are familiar with these concepts, and the others will take some time to get used to them. The majority of the community appears to be unaware of how AI can be utilized for API testing or automation.

The following question was designed to learn how respondents believe AI can improve API test automation.

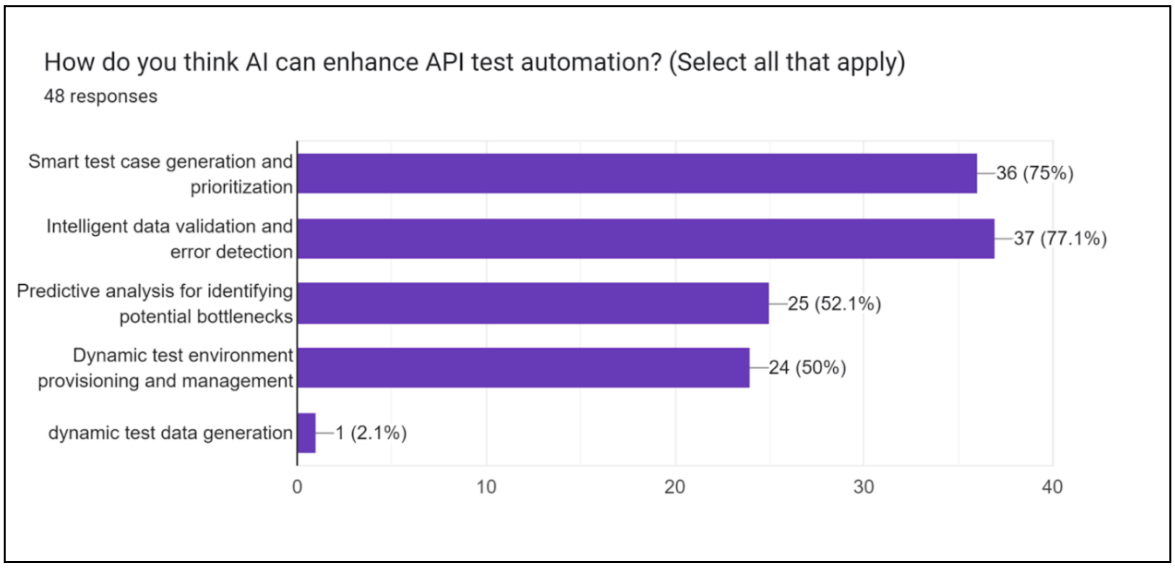

Figure 10: How respondents think AI can improve API test automation

According to the replies gathered, more than 75% responded that Intelligent data validation and error detection, as well as Smart test case generation and prioritization, are the most essential factors that AI concepts may improve. Furthermore, more than half of respondents stated that predictive analysis for identifying possible bottlenecks and dynamic test environment provisioning and management are critical. According to the results, dynamic test data generation is something that falls to the bottom as an importance.

Then we asked respondents if they were thinking about using AI concepts for testing and automation.

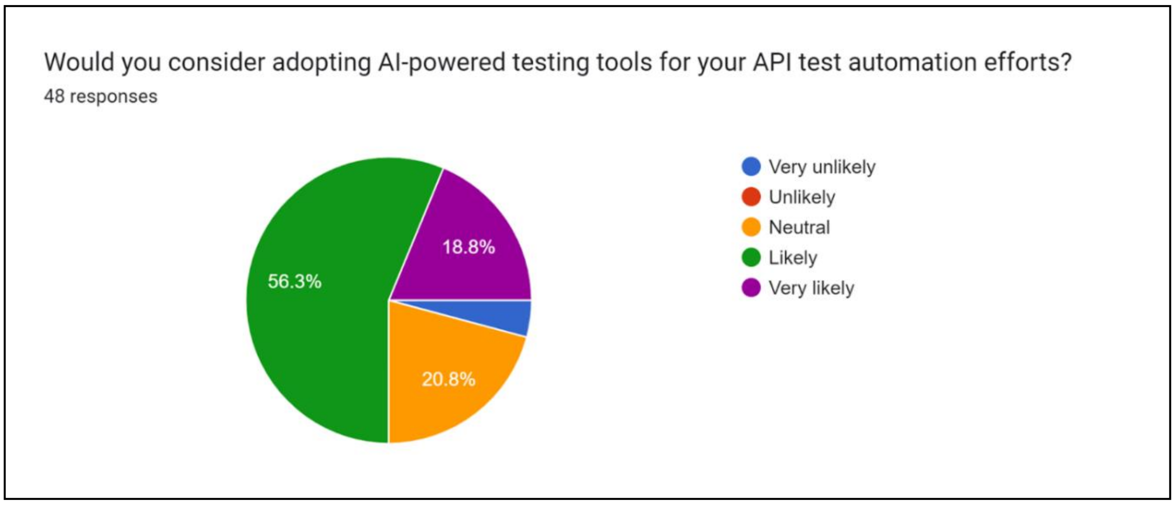

Figure 11: Are respondents consider to adopt AI concepts

While 56.3% of respondents said they are likely to use AI principles for API test automation, 4.2% said they are not. All others stated that they are likely to adopt spontaneously or at a high level. It appears that the majority of the community is likely to accept AI-based products if they are suitable for their needs.

Because AI-based API testing and automation is not widely used, the next challenge is to determine what the QA community's concerns are when using these tools.

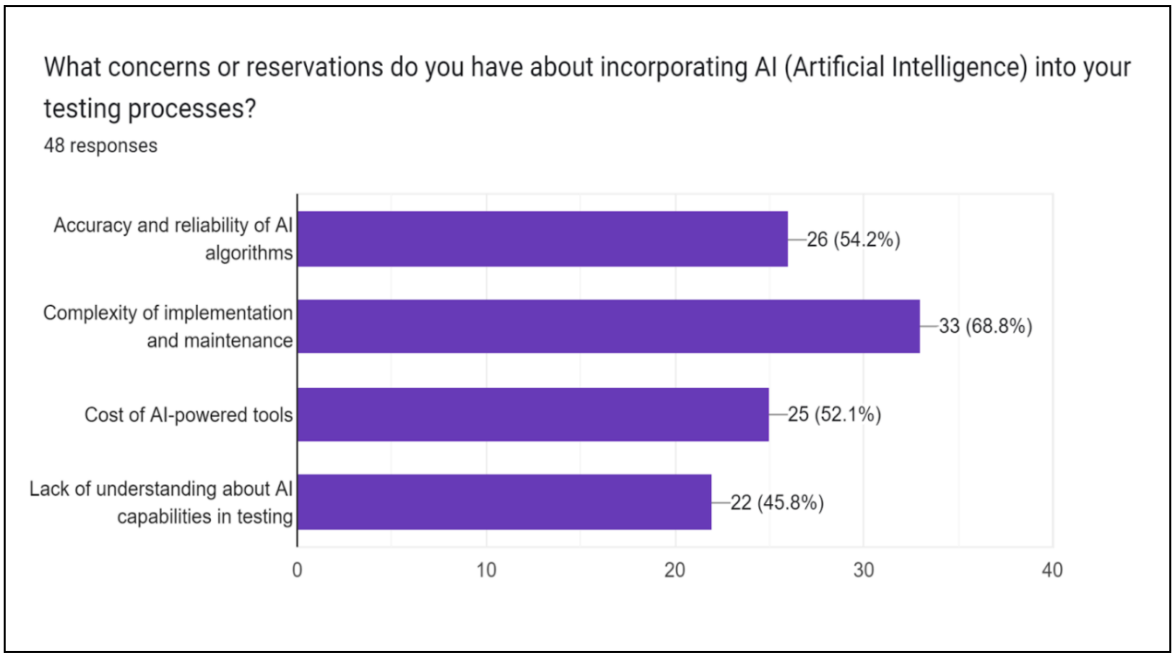

Figure 12: Concerns of incorporating AI to API test process

Almost all predefined objections drew respondents' attention as valid concerns (more than 45% agreed on them). The complexity of implementation and maintenance is the factor that has gotten the most attention. These answers are quite true and realist since respondents express them with their experience and same have identified during the literature review as well.

The following question is intended to define the precise features that a QA can expect from an AI tool for API testing.

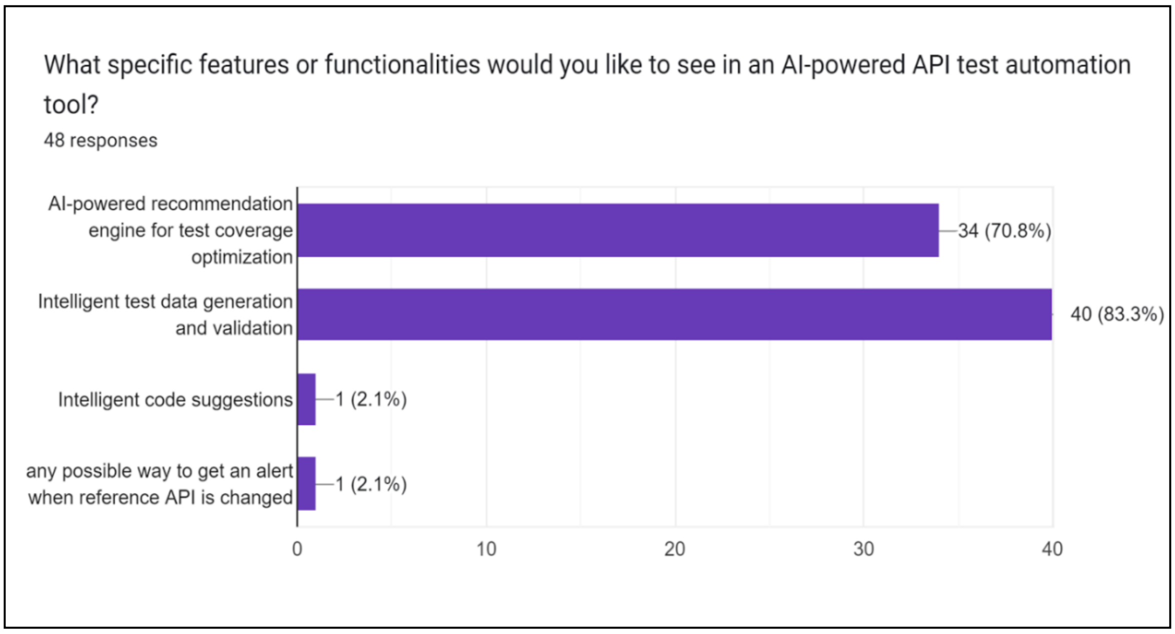

Figure 13: Expected features an AI tool can have

There were two predetermined alternatives for responders (AI-powered recommendation engine for test coverage optimization and Intelligent test data generation and validation), and nearly all of them (more than 70%) marked them as anticipating features. Other than these established possibilities, respondents were given the opportunity to express their own expectations. As a result, two more traits emerge during the survey data collection: Intelligent code suggestions, as well as any conceivable method of receiving an alert when the reference API is updated. All of these are valid and significant considerations to consider while developing a tool.

All of the questions so far have been designed to determine the respondent's API testing and automation expertise, followed by their familiarity with AI principles and knowledge of API testing. The question was then raised as to whether they had employed AI-based tools in practice.

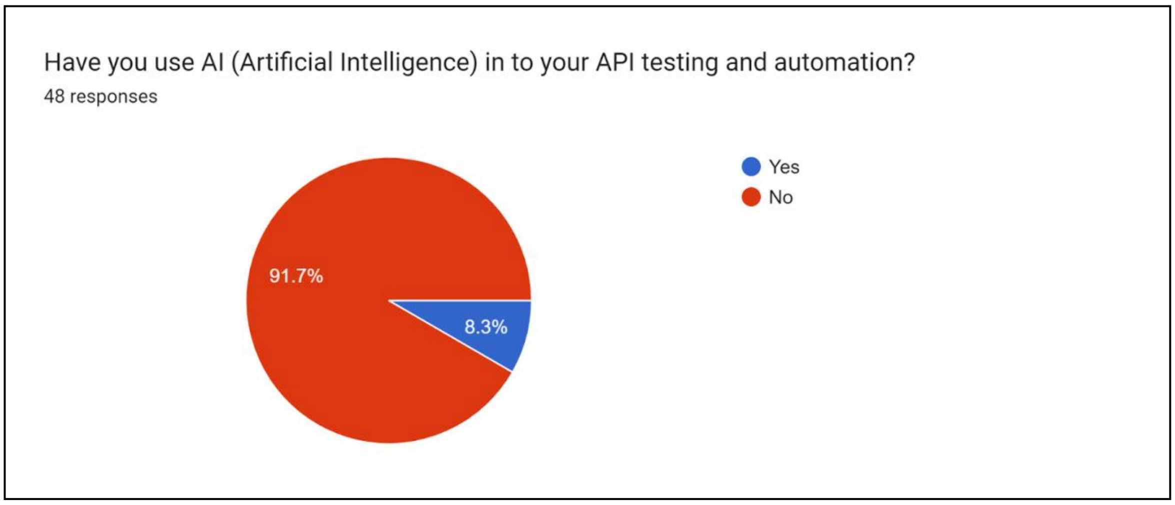

Figure 14: Usage of AI tools for API testing and automation

The majority of respondents (91.7%) stated that they have not utilized any AI technologies for API testing and automation, while 8.3% stated that they have some experience. Many people do not appear to have used them much in practice for a variety of reasons.

The following two questions are intended to elicit information from individuals who have used AI technologies in the context of their work.

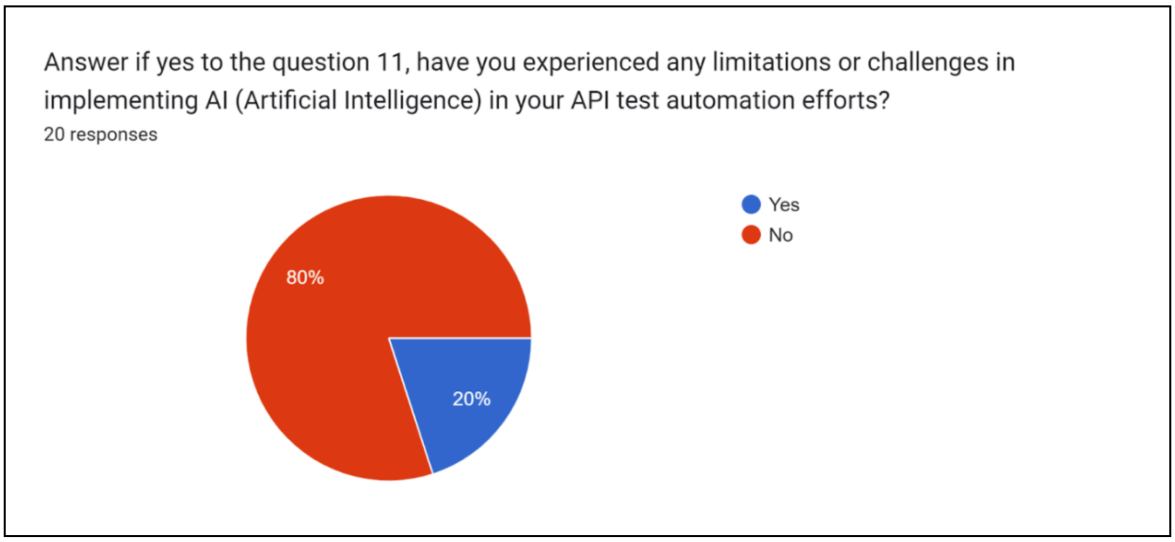

Figure 15: Experience about limitations or challengers

It appears that 80% of experts encountered numerous constraints and experiences while implementing AI-based technologies for API testing and automation. Only 20% people claimed it was straightforward to implement. When queried about the constraints and challenges they experienced, two factors emerged. Some noted that complexity was one of the most difficult issues they had to deal with while implementing, while others claimed that it is frequently near to authentic test cases but not completely correct, necessitating extensive review.

The next question was presented for the entire audience to determine their awareness of AI-based tools, despite having no prior experience with them.

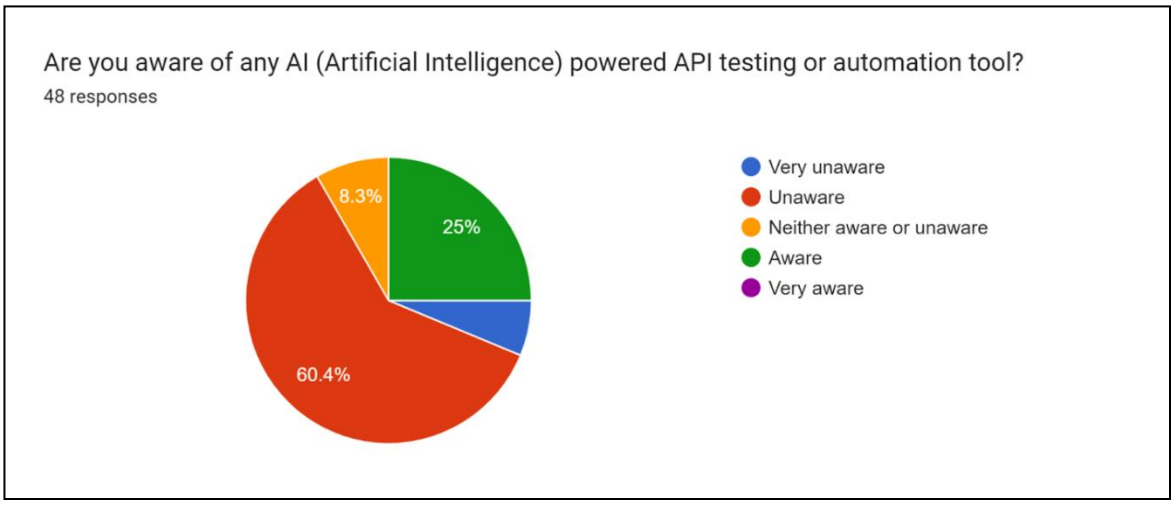

Figure 16: AI tool awareness of the respondents

60.4% of respondents said they are unaware of AI solutions that can be used for API testing and automation, while 33.3% said they are aware or somewhat aware. The majority of people appear to be unaware of these technologies. Then the question rose to the audience with awareness to know what the tools they aware. The following are the tools that arise during data collection.

- Loadmill
- ACCELQ
- Testsigma
- CHAT GPT
- Applitools
- Katalon and SmartBear

We can exclude CHAT GPT from the list because it is a text generation tool that cannot be utilized for API testing or automation.

The following question was regarding the advantages of introducing AI into API testing and automation. The respondents were given four predefined options and the opportunity to state any further benefits they anticipated.

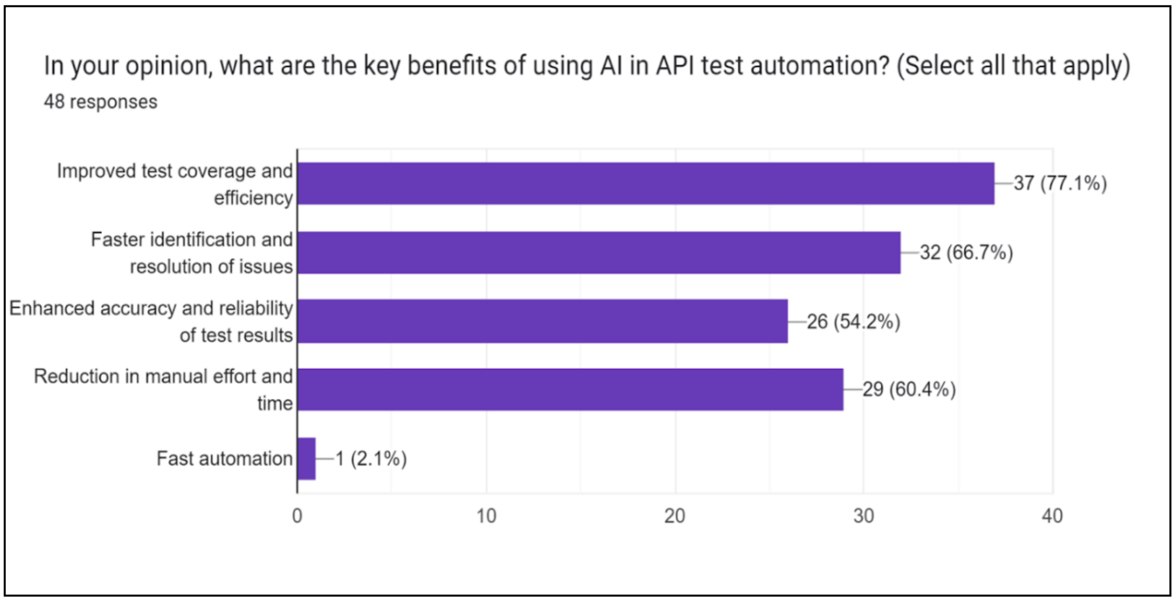

Figure 17: Benefits of using AI-based tools

Other than predetermined options, the audience highlighted Fast automation as an advantage they expect during data collection. Among the preset options, the majority (77.1%) expect increased test coverage and efficiency, while the least (54.2%) expect improved accuracy and reliability of test results. Overall, all received more than 50% of respondents' ticks, indicating that all pre-defined benefits are also community expectations.

The following is an open-ended question about how these professionals see the future of testing and automation with AI technologies. The responses from each respondent are shown in the table below.

Answers
I think it will take over testing in the future
It will reduced the manual work and improve the efficiency of automation work
If this is integrated with CICD pipeline, most of the time ,QA engineers work will not be used. AI can be used to identify testing scope according to implimentation.
It can save in test designing and testing effort of the testers with the better ROI for the projects. Also good point of issue detection in a higher reliability.
It's much more effective and efficient than manual testing.
Very important
Will improve the productivity and will be able tio get good test coverage
I believe that it will be more accurate, efficient and accountable for the API testing

It will be good
Improves efficiency and reduces manual effort
It helps to improve the test quality
Ai will improve the effectiveness and efficiency for example, test data generation, performance testing and predictive analytics using historical test data.
AI will replace interns
It will be more popular
making it more efficient and effective.
Can reduce time
It can reduce human testing effort drastically
AI will highly affect in future not only in testing, but also in every other disciplines of computing. So it will help to reduce the time as well as effort. But it may cause to reduction of jobs as well.
Reduce scenario discovery time, identify deep testing opportunities, early integration
High Unemployment rate in IT industry
It will take over the testing
It will improve efficiency
AI will play a major role in the aspects of bug/defect/code sniffing tools like Bugsnag, Sonar qube & Nagios, Test automation tools/technologies like Selenium as well as API testing tools/technologies like Postman.
And these tools/technologies/products will hopefully start integrating AI into their products.
Once this happens, the testing work would be much easier and the product quality will be much higher.
Similarly, professionals who know how to work with AI & who posses AI related testing experience will be high in demand.
User story creation to everything moving in to AI
Achieve fast automation, high accuracy and better maintainance
AI is expected to have a significant impact on testing work in the future. Several ways in which AI is likely to affect software testing are: Test Automation and Scripting Test Data Generation

Predictive Analytics Chatbots and Virtual Testers Test Reporting and Documentation
Very much. But until machine algorithm gets matured, the reliability would be questionable
Most of the automated tests should be integrated with AIs, and it will make it easier to develop the tests and maintenance easily
Some missing scenarios of the scope
Would love to learn and adopt in future projects
Good
it will speed up the testing work
Ai will be heavily used.
Most efficient and accurate way to testing with less time
AI can make testing efficient
Testing would be easier
AI is Taking IT industry by storm i guess it will be more helpful and will benefit developers in terms of speed and efficiency
It will make things efficient
It will imrove coverage
Test Case Creation and Maintenance: Streamlining Efficiency with AI, Test Execution Optimisation: AI-Powered Prioritisation for Faster Results, Defect Analysis and Root Cause Identification: AI for Enhanced Debugging
Basic tests are written for you, maybe even some E2E types of tests, leaving the more complex, edge cases to QE
Will help humans verify their work and take care of the more menial tasks freeing them up for more valuable work
It is the future
It will improve the test coverage
It will replace manual effort
It will TaKe over manual testing
AI will heavily use for automations
It will improve test coverage

Table 3: Future with AI as per respondents

The answers above can be divided into four categories: test coverage, test efficiency, Software quality, and general.

Category	**Answers**	**Answers count**
Test coverage	• If this is integrated with CICD pipeline, most of the time ,QA engineers work will not be used. AI can be used to identify testing scope according to implimentation. • Will be able tio get good test coverage. • AI is expected to have a significant impact on testing work in the future. Several ways in which AI is likely to affect software testing are Test Automation and Scripting Test Data Generation Predictive Analytics Chatbots and Virtual Testers Test Reporting and Documentation. • Some missing scenarios of the scope. • It will imrove coverage. • Basic tests are written for you, maybe even some E2E types of tests, leaving the more complex, edge cases to QE. • It will improve the test coverage. • It will improve test coverage.	8
Test efficiency	• It will reduced the manual work and improve the efficiency of automation work • It can save in test designing and testing effort of the testers with the better ROI for the projects. Also good point of issue detection in a higher reliability. • It's much more effective and efficient than manual testing.	20

	• Will improve the productivity • Improves efficiency and reduces manual effort. • Ai will improve the effectiveness and efficiency for example, test data generation, performance testing and predictive analytics using historical test data. • making it more efficient and effective. • Can reduce time • It can reduce human testing effort drastically • AI will highly affect in future not only in testing, but also in every other disciplines of computing. So it will help to reduce the time as well as effort. But it may cause to reduction of jobs as well. • Reduce scenario discovery time, early integration. • It will improve efficiency • Achieve fast automation. • Most of the automated tests should be integrated with AIs, and it will make it easier to develop the tests and maintenance easily. • it will speed up the testing work. • Most efficient and accurate way to testing with less time. • AI can make testing efficient. • AI is Taking IT industry by storm i guess it will be more helpful and will benefit developers in terms of speed and efficiency. • It will make things efficient. • Test Case Creation and Maintenance: Streamlining Efficiency with AI, Test Execution Optimisation: AI-Powered Prioritisation for Faster Results, Defect Analysis and Root Cause Identification: AI for Enhanced Debugging.	

	• Will help humans verify their work and take care of the more menial tasks freeing them up for more valuable work.	
Software Quality	• It helps to improve the test quality • I believe that it will be more accurate, efficient and accountable for the API testing • identify deep testing opportunities. • Once this happens, the testing work would be much easier and the product quality will be much higher. • Achieve high accuracy and better maintainance.	5
General	• I think it will take over testing in the future • Very important • It will be good • AI will replace interns • It will be more popular • AI will highly affect in future not only in testing, but also in every other disciplines of computing. So it will help to reduce the time as well as effort. But it may cause to reduction of jobs as well. • AI will play a major role in the aspects of bug/defect/code sniffing tools like Bugsnag, Sonar qube & Nagios, Test automation tools/technologies like Selenium as well as API testing tools/technologies like Postman. • And these tools/technologies/products will hopefully start integrating AI into their products. • High Unemployment rate in IT industry • It will take over the testing • Similarly, professionals who know how to work with AI & who posses AI related testing experience will be high in demand.	20

	• User story creation to everything moving in to AI. • Very much. But until machine algorithm gets matured, the reliability would be questionable. • Would love to learn and adopt in future projects. • Good. • Ai will be heavily used. • Testing would be easier. • It is the future. • AI will heavily use for automations. • It will replace manual effort. • It will TaKe over manual testing.	

Table 4: categorization of the future of test with AI answers

According to the categories, many answers suggest that it will speed up the Quality Assurance or testing process, while one states that it will boost the efficiency by lowering human manual labor and shortening the time required to complete a task. One of the key benefits of AI in testing is its ability to speed up the Quality Assurance process. By leveraging AI techniques, such as machine learning and natural language processing, testing teams can automate various aspects of the testing lifecycle. Moreover, AI can play a significant role in analyzing defects and identifying their root causes. By analyzing vast amounts of data and patterns, AI algorithms can help pinpoint the underlying issues that lead to defects. This enables testers to focus their efforts on fixing the root causes instead of just addressing the symptoms. As a result, the overall efficiency of defect analysis and debugging is enhanced, leading to more effective and reliable software. Additionally, AI can revolutionize the testing process by enabling enhanced debugging capabilities. With AI-powered tools, testers can gain insights into complex codebases and identify potential bugs or performance bottlenecks. By utilizing advanced algorithms, these tools can assist in identifying problematic areas and suggesting possible solutions, thus streamlining the debugging process and reducing the time required resolving issues. In summary, AI will offer tremendous potential in improving test efficiency in the future. By automating manual tasks, accelerating QA processes, enhancing defect analysis, and facilitating debugging, AI can empower testing teams to deliver high-quality software in a more efficient and effective manner.

The next category that has received the greatest attention is test coverage. Many experts believe that AI will assist in generating basic tests, and in some cases, even end-to-end tests. This would allow quality assurance engineers to focus more on handling complex edge cases and ensuring comprehensive coverage. AI can play a crucial role in identifying the testing scope based on the implementation of the software. By analyzing the codebase and understanding the functionalities, AI algorithms can determine which parts of the software need to be tested. This intelligent assessment helps in optimizing testing efforts and resources by prioritizing the most critical areas. Furthermore, AI can enhance test coverage by continuously learning and adapting to modifications in the software. As the codebase evolves, AI algorithms can automatically update test cases to accommodate changes, ensuring that the testing remains up to date and comprehensive. By leveraging AI's capabilities in test generation, scope identification, and adaptive coverage, testing teams can achieve higher levels of coverage and more effectively validate the software's functionality across different scenarios. In conclusion, AI's potential to automate test generation and adapt to software modifications is expected to contribute significantly to improving test coverage. This, in turn, will enable quality assurance engineers to focus on handling complex edge cases and ensure a more robust and comprehensive testing process.

Software quality is another critical category that experts have highlighted when discussing the impact of AI. AI can greatly contribute to improving test quality by identifying deep testing opportunities. By analyzing code and data, AI algorithms can uncover potential areas for extensive testing, ensuring that thorough examination is conducted. Moreover, AI's ability to analyze patterns and data can lead to higher accuracy in test results. With advanced algorithms, AI can identify anomalies and deviations that may go unnoticed by human testers, thus improving the overall quality of the testing process. Additionally, AI can assist in better maintenance of test suites. By automatically adapting test cases to changes in the software, AI can ensure that tests remain relevant and effective over time. This helps in maintaining high-quality testing practices and reducing the risk of outdated or inefficient test cases. In summary, AI's will contribution to software quality in significant way in the future. By identifying deep testing opportunities, achieving higher accuracy, and enabling better maintenance of test suites, AI will play a vital role in enhancing the overall quality of the testing process.

The general category encompasses various perspectives on the overall impact of AI on software quality assurance (QA). One common concern raised by many is the potential job displacement of test engineers due to the automation capabilities of AI. It is true that AI can take over manual work and perform tasks more efficiently. However, it is important to note that the human touch and expertise of test engineers are still highly valued. While AI can automate repetitive and mundane tasks, the reliability and maturity of machine algorithms are crucial factors. Some experts emphasized that until AI algorithms become more mature, human test engineers will continue to play a vital role in ensuring the reliability of AI-assisted testing processes. Test engineers bring critical thinking, domain knowledge, and the ability to handle complex edge cases that may be challenging for AI algorithms to handle on their own. Instead of replacing test engineers, AI is more likely to augment their capabilities. By automating routine tasks, AI frees up test engineers to focus on higher-value activities such as test strategy, exploratory testing, and analyzing complex scenarios. This collaboration between AI and test engineers can lead to more comprehensive and efficient software quality assurance processes. In summary, while concerns about job displacement exist, the reliability of AI and the importance of human expertise in testing cannot be overlooked. The collaboration between AI and test engineers is likely to enhance the overall software quality assurance process rather than replace human involvement altogether.

The final question of this survey was designed to learn about the respondents' views about the survey's quality. This was an optional question, with only 18 responses received.

Answers
Thai is good survey and will give away of AI automation
It also helps me to understand and increase my awareness on AI driven test automation on APIs in much more extent. Good survey to check the awareness too.
Very impressive and useful for the future of test automation.
Interesting and make my mind to investigate more on this area :)
Interesting topic
Needy survey at this time
great

great
It was a useful and positive thing is topic selected while everyone talk about AI tools, better add some AI tools available for the survey so if someone uses any of those, they will give some insights
It's very informative one. Good luck
Decently structured.
good one
great topic which will definitely shape up the api test decipline in near future All the best!
Conducting a survey on AI-driven API testing for future implementations is an excellent idea. This survey can provide valuable insights into the current landscape, challenges, and trends in AI-driven API testing.
Good survey
From the testing perspective there is very limited knowledge in the AI aspects, I think this will be a great survey and looking forward to see some insights from this
Really good survey
Overall good coverage but I think it needs more specifics to capture the AI adoption as I think.

Table 5: Opinions about the quality of the survey

Almost all responders remarked that this was a much-needed survey that was well-structured but brief. Another suggestion was to enhance this by concentrating more on AI adoption through testing. It appeared respondents have positive view about the survey.

6 Discussion

The purpose of this discussion is to delve into the survey results alongside the solution outlined in the literature review. In the first part, this will explore these aspects together. In the second part, this will focus on brainstorming potential solutions and methods to overcome the current challenges. Based on the survey results, it appears that most of the respondents have experience in API testing and automation, but they have yet to explore AI-related tools or frameworks. Interestingly, despite this lack of experience, they are enthusiastic about delving into this domain. It's great to see such enthusiasm for learning and embracing new technologies.

According to the survey results, respondents have highlighted several challenges they currently face with API automation. Let's take a closer look at each of these challenges and explore possible solutions decribed in the litreature. Handling dynamic data: Many respondents noted the difficulty in dealing with dynamic data within their API tests. As APIs often return varying data, it can be challenging to create stable and reliable test scripts. The solution developed by Mirabella et al. (2021) using deep learning has proven effective in addressing the aforementioned challenge to some extent. As discussed in the literature review, the researchers trained their model using a significant amount of historical API testing data, enabling it to learn patterns and correlations. The outcome of their model has shown promising results. However, one concern raised by respondents, as identified in unstructured interviews, is the model's ability to identify data requirements from various data sources. This raises the question of whether the model can effectively derive the data from different data bases or structures. To further enhance the solution, researchers could consider incorporating techniques and methodologies that allow the model to handle diverse data sources. This may involve preprocessing techniques to normalize and standardize data from various formats, as well as techniques to identify and extract relevant information. Furthermore, conducting additional research and gathering more feedback from users would be beneficial in understanding specific data source challenges and refining the model accordingly. By actively addressing these concerns, the solution can be optimized to better handle data requirements from different sources, ultimately improving its overall performance and usability.

Maintaining test scripts with changing APIs is another challnge voted by the respondets. APIs are prone to change over time, which can lead to script maintenance issues. Respondents expressed concerns about keeping their automation scripts up to date. The EvoMaster solution stands out as a powerful option for addressing this concern. It offers the ability to analyze code and automatically generate test cases, as well as update them when new changes occur. This feature is particularly advantageous for APIs that are developed internally or have known code bases. However, it's worth noting that the EvoMaster solution's applicability is limited to APIs with access to the codebase. It requires the codebase as input to perform its analysis and generate test cases effectively. This means that for APIs that are not internally developed or lack a known codebase, alternative solutions may need to be considered. While other solutions may focus on aspects such as test case generation, data handling, and validations, they may not explicitly address the challenge of handling different API versions. This highlights a potential gap in the current solutions available. To overcome this limitation, further research and development could be conducted to explore solutions that specifically address version handling for APIs. This could involve techniques such as version control systems, API versioning frameworks, or intelligent algorithms that can adapt test cases to different API versions. By focusing on this critical aspect, developers and researchers can bridge the gap in existing solutions and provide a comprehensive approach to API testing and version management.

Handling authentication and authorization: Authentication and authorization mechanisms can add complexity to API automation. Respondents mentioned the need to handle various authentication methods and ensure secure access to the APIs. This has not been specifically mentioned in any solutions, although it is readily addressed unless it involves any session-related data. Integrating authentication libraries or frameworks and implementing secure token management solutions can help address this challenge. Integrating with CI/CD was another concern that many respondents expressed, they desire to seamlessly integrate their API automation with Continuous Integration/Continuous Deployment (CI/CD) pipelines. This integration enables automated tests to run as part of the software development lifecycle. Leveraging tools like Jenkins, GitLab, or TeamCity can facilitate the integration process. This is another interesting subject for research, however this study considers it as out of scope as the main motivation for doing this is API testing automation. Some respondents mentioned the struggle of comprehending complex API documentation. Clear and comprehensive documentation is crucial for successful API automation. Utilizing tools like Swagger or Postman, which provide interactive documentation and testing capabilities, can greatly assist in understanding and utilizing APIs effectively. However, according to unstructured conversations with several specialists, many projects now employ swagger as the REST API documentation, which is easily understood by testers and business people alike. As a result, it appears that AI is not required to tackle this issue, and if AI is used to generate test cases, understanding documentation may be unnecessary. Even the detailed solution in the literature study does not necessitate a prior in-depth comprehension of the sepcifications.

Given the current challenges in API testing and automation, there are high expectations for AI to come to the rescue. Two particular areas that have caught the attention of many respondents are intelligent data validation and error detection, as well as smart test case generation and prioritization. It's exciting to see how AI can address these challenges, and current solutions are already making progress in these areas to some extent. The solutions presented by Martin-Lopez et al. (2022), Mirabella et al. (2021), and Martin-Lopez (2020) have shown promise in addressing the challenges of API testing and automation. However, further refinement is needed through extensive training and addressing diverse data sources and versioning issues. It is particularly important to focus on error detection from multiple angles, such as security and performance, to ensure comprehensive testing. Indeed, Martin-Lopez et al. (2022) have taken a step forward by incorporating API fuzzing techniques into their solution to address security concerns and uncover vulnerabilities. In addition to these expectations, respondents are also eager for new solutions to prioritize predictive analysis for identifying potential bottlenecks and dynamic test environment provisioning and management. These advancements hold great potential in enhancing the effectiveness and efficiency of API testing. Both predictive analysis and dynamic test environment provisioning and management are areas that have not been explored extensively in the existing solutions. They present exciting research opportunities. Interestingly, during unstructured interviews, some professionals mentioned that incorporating the use of Docker images and executing tests on them would greatly enhance usability and effectiveness while reducing manual work. This suggests that leveraging Docker can be a valuable addition to future AI solutions in API testing and automation.

While there is certainly a strong desire to utilize AI for testing and automation purposes, it's important to acknowledge that there are some challenges and concerns associated with the currently available commercial tools. According to the survey, approximately 8.3% of respondents have experience with AI-driven API testing and automation, and they have identified two major obstacles. The first challenge is the complexity involved, while the second is the issue of accuracy, as the results generated by AI tools often require extensive review to ensure full accuracy. That's absolutely true! Many AI-driven tools are still in the early stages of their journey, and we can expect them to become more user-friendly and reliable as they evolve. Even respondents without prior experience with AI-driven tools expressed reservations about their usage. Some of the main concerns mentioned were the complexity of implementation and maintenance, the accuracy and reliability of AI algorithms, and the cost of using AI-powered tools. However, let's explore how current tools can address these concerns and help overcome these challenges. Among the tools mentioned by respondents, Loadmill stands out as a good option. Loadmill has the impressive capability of generating API tests within minutes, offering two options for test generation (Loadmill, no date). The first option is contract testing, where tests are generated based on the JSON schema. The second option is to record and generate tests using the Test Composer Chrome extension. This is a better solution because it eliminates the need for prior knowledge about the specification or the API in order to generate automated tests, making it highly flexible and dynamic. Indeed, while Loadmill does address some challenges such as complexity and cost to some extent, there are still a few gaps to consider. One of the main concerns is the maintenance aspect, as the generated test scripts will still require knowledgeable experts to maintain them for updates. Additionally, Loadmill being Chrome-specific may limit its compatibility with other browsers or platforms. Another gap is the issue of versioning, as the tool may generate new scripts each time, requiring the entire test suite to be replaced, which can be costly and time-consuming, especially when dealing with multiple APIs and test cases. Furthermore, there are additional challenges when it comes to replacing the test suite generated by Loadmill. Each time a replacement occurs, someone will need to be involved in updating the test data within the scripts. This poses a problem as Loadmill does not provide a solution for versioning, platform independence (regarding the extension), and ongoing maintenance. These factors continue to remain unresolved by the tool. Another tool mentioned by respondents is Testsigma, which is an AI-driven tool specifically designed for GUI testing

GUI (Testsigma, No date). While it does have API support, it is similar to Postman and may require further optimizations for API testing. Similarly, ACCELQ follows a similar pattern, but it stands out due to its seamless integration with various platforms and its ease of adoption (Andrades, 2023). The other tools mentioned also fall into similar categories, and overall, there are still gaps that need to be filled in the AI-driven testing landscape.

Based on the results and previous discussion, there are a few key gaps that still need to be addressed by future solutions. These include:

- Comprehensive Training: Future solutions should focus on further refining their training process to improve accuracy and reliability in API testing and automation.
- Handling Diverse Data Sources: It is essential for future solutions to be able to handle different data sources effectively, ensuring compatibility and reliable testing across various scenarios.
- Versioning Challenges: Future solutions need to address the complexities that arise from versioning in APIs, enabling seamless testing and automation across different versions and update existings.
- Error Detection in Multiple Dimensions: While current solutions touch upon error detection, future solutions should strive to cover various dimensions such as security and performance to ensure comprehensive testing.
- Predictive Analysis: Incorporating predictive analysis into solutions can enable the identification of potential bottlenecks, allowing for proactive measures to optimize API performance.
- Dynamic Test Environment Provisioning and Management: Future solutions should emphasize the ability to dynamically provision and manage test environments, enabling efficient and flexible testing processes.
- Less manul labor involvement: No need frequent extensive review and updates by experts..
- Test case prioritation: Priortize tests based on identified main flows and edge cases. This will further enhance the testing process and reduce the manul work need from experts.
- Platform indepedent solution.

By addressing these gaps, future solutions can advance the field of API testing and automation, providing more effective and reliable solutions for professionals.

6.1 Framework Concept

This study effectively created a concept for a future implementation tool by taking into consideration the survey results and findings from the literature review. Building on the findings, the approach seeks to overcome the existing gaps and obstacles in API testing and automation. This establishes the groundwork for the creation of an innovative solution that can expedite and improve the efficiency of API testing processes.

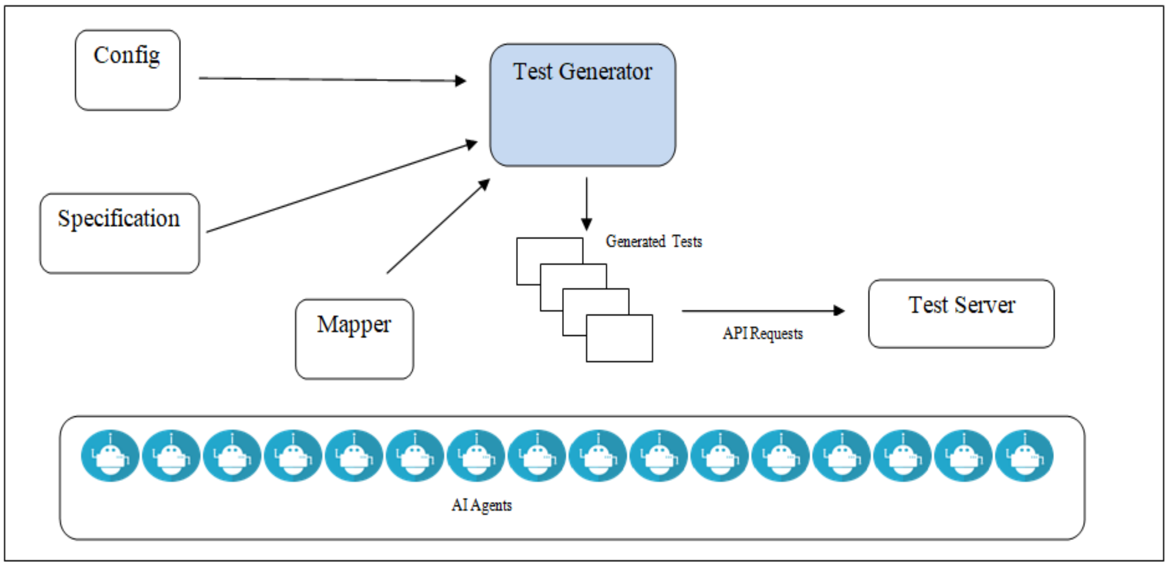

Figure 18: Concept to gain knowledge

Based on the solutions analyzed in the literature review, Figure 18 represents a concept for gaining knowledge about different API testing and automation. The flow begins by obtaining API specifications, such as Swagger documents, which are used to educate AI agents. These specifications should encompass various aspects, including different versions of the same API, diverse structures, various HTTP methods, varying authorization mechanisms, and API examples from different domain areas. This approach aims to provide comprehensive knowledge to the AI agents for effective API testing and automation.

- Config: This file contains the main items required to successfully perform a test (such as authorization details and environment details) as well as parameters that are more important to test.
- Mapper: This will appear right before generating the test cases to include whether the system needs to obtain test data from a different data source. This might be an Excel file or a database. This should define how data should be mapped.

The process will begin after the specification and config are complete. These two will be the test generator's principal input. The test generator begins to build tests based on the specifications, employing various test approaches such as boundary value analysis, random sampling, decision trees, and equivalent portioning. If the tests require data from outside sources, the test generator will include those in the created tests depending on the mapper specification. We have now produced tests that will run against the chosen environment. This entire method is not novel, but rather an existing one devoid of AI notions. Figure 18 depicts a layer of AI bots. These AI agents have been installed in locations to monitor and learn what is happening. These AI agents are AI-powered bots that are built using unsupervised learning and k-means clustering. They have mostly split into two factions. One group will monitor and understand the patterns around the test generator, while the other will be positioned around the API requests area, where the created tests are executed. The second group will utilize reasoning techniques to monitor and learn relationships and patterns. This will also allow agents to learn erroneous situation and success situations. Once these agents have gained sufficient knowledge, they will begin to alter the config file and redefine the key parameters. Then, in addition to the specification-based tests, these agents will assist the generator module in creating tests based on patterns and relationships, as well as error-prone conditions learned by both groups. We can improve this system further by including reinforcement learning. In this scenario, the action will generate test cases, and if the cases are valid, the agents will be rewarded. This procedure will reduce the number of invalid test cases. To train the AI agents, this approach necessitated a massive number of specifications. Once these agents have gained sufficient knowledge (when the test case validity rate exceeds 90%), this knowledge can be utilized to develop a module core engine to generate test cases.

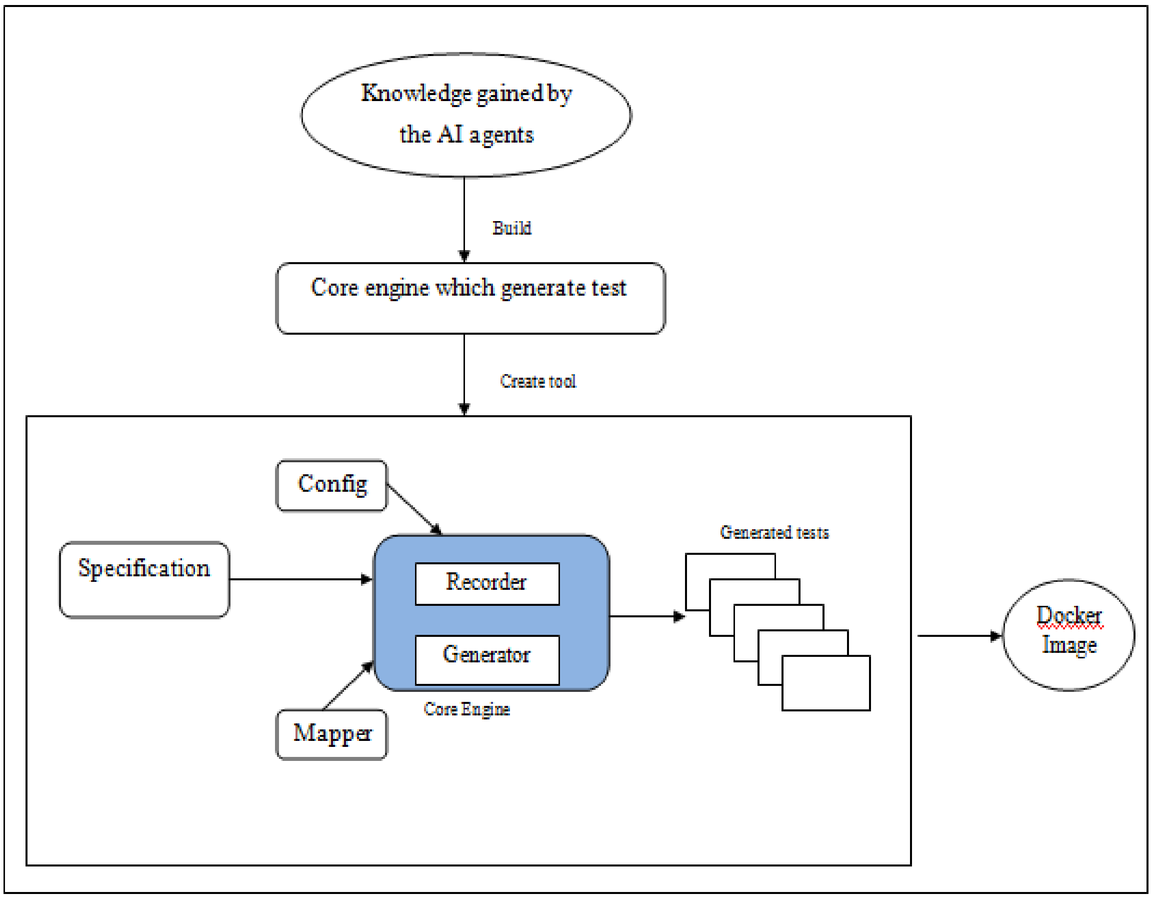

Figure 19: Concept for a future tool

Figure 19 depicts the conceptual framework for an AI-based test generation tool. The core engine, which has knowledge of the AI agents, is the essential component of this framework. This system still requires the config file and the mapper. The configuration file is now different since it only contains authorization-specific information and target environment-specific information. The core engine will identify the key parameters based on the knowledge obtained. The mapper remains unchanged since the module cannot determine where to obtain the necessary test data for generating tests and maps. If the mapper is empty, the generator will populate it with test data based on the specifications. To generate tests, this tool primarily has two options available. The first is to provide a specification as input, and the second is that the core engine contains a recorder that will record and identify API requests when the target system is executing. This tool's capabilities do not stop with test case generation. As the process concludes, a docker image of the created test case repository will be generated, which can be used to install into any dynamic environment. Users may input the target environment (the endpoint for API calls) and specified authorization details as args key-value pairs during the docker image installation process, which will update the config file. Otherwise, it will use the installation environment as the end point and the authorization as no auth or the details specified in the config file. As a result, this system can now produce tests with or without specifications and assist in the execution of these tests in various environments. The following problem to consider is updating these test cases when the API version changes. This capability can be implemented without the use of artificial intelligence. The docker image can keep the API version as metadata, and if an update is required, the docker image can provide as an input to the system and the new specification as well. Then the core engine will recognize the delta and add, update, or remove test cases.

This process can improve by using deep learning but need more research to specify a formal concept. The concept presented here mainly trying to fill the gaps identified in this study but this can be further improve by adding non-functional testing support based on generated tests.

7 Conclusion

Today, APIs play an important role in many system architectures as they transition from monolithic to more flexible microservice architectures. Given that API testing and automation are important contributors to software quality, they necessitate a thorough understanding of the API structure as well as some technical knowledge. With the growing number of open and internal APIs, this has started to put more and more weight on the shoulders of the developers and testers. As a result, the purpose of this research was to determine what testing and automation alternatives are currently available for APIs. What are the current issues of API testing and automation? And how may artificial intelligence be utilized to improve API testing? According to the findings of this study, the current tools used by professionals have significant shortcomings, and there is not enough study on AI-driven API testing. Based on the findings, the present QA community does not have enough experience with AI-powered API tools, and there are strong expectations that AI will tackle some of today's most pressing challenges. Furthermore, taking into account the data and the findings of the literature review, conceptual framework has been created to address some of the identified gaps. The provided concept is only a starting point for this topic; further research is required to assess its feasibility, how well it fits with expectations, and to improve it with deep learning.

References

DUSTIN, Elfriede, RASHKA, Jeff, and PAUL, John. Automated software testing: introduction, management, and performance. Addison-Wesley, 1999. p. 4. ISBN 978-0201432879.

Ed-douibi, H., Izquierdo, J.L.C. and Cabot, J. (2018) ‘Automatic Generation of Test Cases for REST APIs: a Specification-Based Approach’, IEEE 22nd International Enterprise Distributed Object Computing Conference [Preprint].

Andrea , A. (2019) ‘RESTful API Automated Test Case Generation with EvoMaster’, ACM Trans. Softw. Eng. Methodol, 28(1).

Martin-Lopez, A. (2020) ‘AI-Driven Web API Testing’, IEEE/ACM 42nd International Conference on Software Engineering: [Preprint].

Sohan, S.M., Anslow, C. and Maurer, F. (2015) ‘A case study of web api evolution’, 2015 IEEE World Congress on Services [Preprint]. doi:10.1109/services.2015.43.

Maleshkova, M., Pedrinaci, C. and Domingue, J. (2010) ‘Investigating web apis on the World Wide Web’, 2010 Eighth IEEE European Conference on Web Services [Preprint]. doi:10.1109/ecows.2010.9.

Introduction (no date) Web APIs. Available at: https://www.w3schools.com/js/js_api_intro.asp (Accessed: 25 June 2023).

Wang, S., Keivanloo, I. and Zou, Y. (2014) ‘How do developers react to restful API evolution?’, Service-Oriented Computing, pp. 245–259. doi:10.1007/978-3-662-45391-9_17.

Espinha, T., Zaidman, A. and Gross, H.-G. (2014) ‘Web api growing pains: Stories from client developers and their code’, 2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE) [Preprint]. doi:10.1109/csmr-wcre.2014.6747228.

Ackerman, S. et al. (2021) ‘Towards API Testing Across Cloud and Edge’, IBM Research [Preprint].

Golmohammadi, A., Zhang, M. and Arcuri, A. (2022) ‘Testing RESTful APIs: A Survey’, Kristiania University College and Oslo Metropolitan University.

BANGARE, S. et al. (2012) ‘AUTOMATED API TESTING APPROACH’, International Journal of Engineering Science and Technology (IJEST), 4(2).

Brown, A. (2018). The Benefits of API Testing Automation. Journal of Software Testing, 25(2), 45-58.

Garcia, M. (2019). Challenges and Solutions in API Testing and Automation, International Journal of Software Engineering, 42(3), 112-128.

Johnson, R. (2018). API Testing Approaches and Best Practices. Journal of Software Development, 35(4), 76-89.

Jones, L. (2019). Importance of API Testing in Modern Software Development, International Conference.

Smith, J. (2019). Introduction to API Testing. Journal of Software Engineering, 37(1), 23-38.

White, T. (2020). API Testing Tools and Frameworks. Journal of Software Engineering, 34(1).

Ergen, M. (2019) ‘What is Artificial Intelligence? technical considerations and future perception’, The Anatolian Journal of Cardiology [Preprint]. doi:10.14744/anatoljcardiol.2019.79091.

Du-Harpur, X. et al. (2020) ‘What is ai? applications of artificial intelligence to dermatology’, British Journal of Dermatology, 183(3), pp. 423–430. doi:10.1111/bjd.18880.

Agrawal, A., Gans, J.S. and Goldfarb, A. (2018) ‘What to expect from Artificial Intelligence Technology’, What the Digital Future Holds, pp. 23–36. doi:10.7551/mitpress/11645.003.0008.

Shihab, E. et al. (2022) ‘The present and future of Bots in Software Engineering’, IEEE Software, 39(5), pp. 28–31. doi:10.1109/ms.2022.3176864.

Erlenhov, L., Neto, F.G. and Leitner, P. (2020) ‘An empirical study of bots in software development: Characteristics and challenges from a practitioner’s perspective’, Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering [Preprint]. doi:10.1145/3368089.3409680.

King, T. M., Arbon, J., Santiago, D., Adamo, D., Chin, W., & Shanmugam, R. (2019). AI for testing Today and tomorrow: Industry perspectives. 2019 IEEE International Conference On Artificial Intelligence Testing (AITest). doi:10.1109/aitest.2019.000-3

Gao et al. (2022), ‘An Approach to GUI Test Scenario Generation Using Machine Learning’, in Proceedings - 4th IEEE International Conference on Artificial Intelligence Testing, AITest 2022, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 79–86. doi: 10.1109/AITest55621.2022.00020.

Eskonen et al. (2020), ‘Automating GUI testing with image-based deep reinforcement learning’, in Proceedings - 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2020, Institute of Electrical and Electronics Engineers Inc., Aug. 2020, pp. 160–167. doi: 10.1109/ACSOS49614.2020.00038.

Walia R. (2022), ‘Application of Machine Learning for GUI Test Automation’, in 2022 28th International Conference on Information, Communication and Automation Technologies, ICAT 2022 - Proceedings, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICAT54566.2022.9811187.

Reza, H. and Van Gilst, D. (2010) ‘A framework for testing restful web services’, 2010 Seventh International Conference on Information Technology: New Generations [Preprint]. doi:10.1109/itng.2010.175.

Mirabella, A.G. et al. (2021) ‘Deep learning-based prediction of test input validity for restful apis’, 2021 IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest) [Preprint]. doi:10.1109/deeptest52559.2021.00008.

Martin-Lopez, A., Segura, S. and Ruiz-Cortés, A. (2022) ‘Online testing of Restful apis: Promises and challenges’, Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering [Preprint]. doi:10.1145/3540250.3549144.

Soumya Amrita Kirthi, C.N., Kumar, K.L. and Suma, H.N. (2019) ‘Novel evaluation methodology for machine learning algorithms through Api Creation’, 2019 11th International Conference on Communication Systems &amp;amp; Networks (COMSNETS) [Preprint]. doi:10.1109/comsnets.2019.8711468.

‘New Parasoft SOAtest Release Introduces Machine Learning to API Test Creation: Parasoft will showcase the next generation of its award-winning AI-powered API testing solution at STARWEST’ (2018) ProQuest [Preprint]. PR Newswire Association LLC. Available at: https://www.proquest.com/docview/2115328335?parentSessionId=BshpJ%2FLhX6i%2BtTdwQXueDiNonZjN%2B4BfuBfOIz4J91U%3D&amp;pq-origsite=primo&amp;accountid=27292 (Accessed: 29 July 2023).

Locke. E. A. 2007. The case for inductive theory building. Journal of Management, 33 (6): 867- 890

Loadmill (no date), AI - powered solution, Available at: https://docs.loadmill.com/introduction/loadmill-ai-powered-solution (Accessed: 15 December 2023).

Testsigma (No date), AI-powered, low-code test automation platform. Available at: https://testsigma.com/ai-driven-test-automation (Accessed: 15 December 2023).

Andrades, G. (2023) Top 12 best API testing tools for 2023 and beyond, ACCELQ Inc. Available at: https://www.accelq.com/blog/api-testing-tools/ (Accessed: 15 December 2023).

Pugliese, R., Regondi, S. and Marini, R. (2021) ‘Machine learning-based approach: Global Trends, Research Directions, and regulatory standpoints’, Data Science and Management, 4, pp. 19–29. doi:10.1016/j.dsm.2021.12.002.

Note: More information on referencing is available in Appendix 1.

Appendix 1. Survey

Good text processing skills make writing your final thesis and using this template easier. Therefore, you should make sure you have the sufficient basic skills to edit long documents with text processing software before you start. This involves applying the styles, understanding automatic referencing and knowing how to divide your text into sections.

Survey – AI-Driven API Testing and Automation

1. How many years of experience do you have in API testing and automation?
   - Less than one year
   - 1- 2 years
   - 2 - 5 years
   - 5 - 10 years
   - More than 10 years

2. Have you implemented API test automation in your projects?
   - Never
   - Rarely
   - Sometimes
   - Often
   - Always

3. How many years of experience do you have with API automation?
   - Less than one year
   - 1- 2 years
   - 2 - 5 years
   - 5 - 10 years
   - More than 10 years

4. Please rate how important you consider the API test automation in your development process?

- Very unimportant
- unimportant
- Neutral
- Important
- Very important

5. Which tools/frameworks have you used for API test testing and automation?
   - Postman
   - SoapUI
   - Newman
   - Restassured
   - Other (please specify)

6. What challenges have you faced while automating API tests? (Select all that apply)
   - Understanding API documentation
   - Handling authentication and authorization
   - Handling dynamic data
   - Maintaining test scripts with changing APIs
   - Integrating with CI/CD pipelines
   - Other (please specify)

7. Are you familiar with the concept of AI-powered testing?
   - Very unfamiliar
   - Unfamiliar
   - Somewhat familiar
   - Familiar
   - Very familiar

8. How do you think AI can enhance API test automation? (Select all that apply)
   - Smart test case generation and prioritization
   - Intelligent data validation and error detection
   - Predictive analysis for identifying potential bottlenecks

- Dynamic test environment provisioning and management
- Other (please specify)

9. Would you consider adopting AI-powered testing tools for your API test automation efforts?
   - Very unlikely
   - Unlikely
   - Neutral
   - Likely
   - Very likely

10. What concerns or reservations do you have about incorporating AI (Artificial Intelligence) into your testing processes?
    - Accuracy and reliability of AI algorithms
    - Complexity of implementation and maintenance
    - Cost of AI-powered tools
    - Lack of understanding about AI capabilities in testing
    - Other (please specify)

11. What specific features or functionalities would you like to see in an AI-powered API test automation tool?
    - AI-powered recommendation engine for test coverage optimization
    - Intelligent test data generation and validation
    - Other (please specify)

12. Have you use AI (Artificial Intelligence) in to your API testing and automation?
    - Yes
    - No

13. Answer if yes to the question 11, have you experienced any limitations or challenges in implementing AI (Artificial Intelligence) in your API test automation efforts?
    - Yes

- No

14. Answer if yes to the question 12, please specify limitations or challenges you experienced

15. Are you aware of any AI (Artificial Intelligence) powered API testing or automation tool?
    - Very unaware
    - Unaware
    - Neither aware or unaware
    - Aware
    - Very aware

16. Please specify what are those tools you are aware about AI (Artificial Intelligence) powered API testing or automation?

17. In your opinion, what are the key benefits of using AI in API test automation? (Select all that apply)
    - Improved test coverage and efficiency
    - Faster identification and resolution of issues
    - Enhanced accuracy and reliability of test results
    - Reduction in manual effort and time
    - Other (please specify)

18. How do you see AI to affect testing work in the future?

19. Give your thoughts about this survey?