

# A method to automate from test requirements to reports using GenAI



# Table of contents

Abbreviations	3
Abstract	4
Introduction	4
Market trends	5
Problem statement	6
Solution	6
Case study	7
Benefits	13
Conclusion	13
References	13
Author information	14

# Abbreviations

GenAI	Generative Artificial Intelligence
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
TC	Test Case
UI	User Interface
PDF	Portable Document Format
API	Application Programming Interface

# Abstract

In both Software Development Lifecycle (SDLC) and Software Testing Lifecycle (STLC), reviewing requirements is crucial for identifying test scenarios and user stories to develop test cases. Planning and writing these manually from requirement documents are time-consuming. Test cases may later become test scripts or detailed steps based on the test automation framework for execution and validation. This process can significantly affect the overall development timeline.

To address this challenge, this whitepaper proposes a Generative AI (GenAI) Large Language Models (LLMs) based solution using Retrieval-Augmented Generation (RAG) techniques to automate the generation of test cases from requirements. This solution allows users to input text requirements and documents to generate detailed test cases based on their queries. RAG implementations enable users to input large documents and reference documents that cannot be directly passed to LLMs due to token limitations.

The GenAI model will generate test cases in any specified format, such as functional, tabular, scripts or Gherkin formats, based on the information provided in the prompt. This approach streamlines the process of test case generation by reducing manual efforts. Users may require test cases to include Application Programming Interface (API)/Keywords from the application's dictionary in certain use cases. This solution allows for incorporating existing API/Keywords into generated test cases as necessary.

This solution uses a standalone LLM configured as a service, reducing model initialization time and increasing efficiency. Integrating these technologies reduces the time spent creating test cases, speeding up the testing phase and improving development efficiency. The generated test cases can be used in frameworks for execution with user review. This solution can be integrated with an existing test execution framework, making it highly useful for delivering responses in the desired format and working seamlessly.

## Introduction

Interpreting and converting requirements into test cases is often time-consuming and challenging. It involves understanding business requirements, breaking them into testable scenarios and writing test cases. This process can be lengthy and requires substantial collaboration among the testing team. These test cases are developed into test scripts or in BDD, template format or any standard format required to execute the test automation frameworks. This whitepaper proposes a method to automate the entire process, from the test requirements to test report generation using GenAI and AI algorithms.

This solution helps convert requirements to test cases/ test scenarios using the LLM, which will be suitable for the execution of test automation frameworks in required formats like test script, BDD, Gherkin, Template format, etc.

The scripts generated by the LLM can be reviewed and directly fed into any test automation frameworks available in the market, like modular-based, data-driven, hybrid, keyword-driven, etc., for execution. The test cases documents/Requirements may be directly passed to the LLM if the document does not exceed the supported token limits. Each LLM, depending on its efficiency, can only support certain token limits like 4096 or 8192, including both input and output. In case of passing larger documents with tokens exceeding this limit, it will create a token indexing error by the LLM. These kinds of scenarios are overcome by the implementation of the RAG algorithms that can handle single or even multiple documents for processing using LLM. Interacting with the LLM prompt works as a bridge between passing the input/query and getting the desired output.

The key LLM parameters like temperature, top-p and top-k also help with getting proper outcomes as these parameters manage the creativity, randomness and predictive capability of the text generation as outputs. The prompt conveys the instructions, conditions or thoughts to the LLM about the output response to be generated for the query passed by the user. Different kinds of prompts are implemented, like system prompt, default prompt, user prompt and ending prompt, to help the user get the test cases in the desired format. These prompts are engineered to get output test cases that ensure test coverage, sufficiency and boundary conditions are taken care of generated test cases. This solution uses a standalone/on-prem LLM configured as a service; this allows the model to be immediately available and ready to process inputs without any delay and to get a faster response.

The generated test cases can be directly reviewed and fed into test automation frameworks. For example, the Falcon test automation is a scriptless/no-code framework that expects test cases with detailed test steps in a pre-defined template format with column names like sl.no, test step, test data and expected result. The LLM generates test cases from requirements in a format that can be reviewed for correctness and directly passed for test execution. This solution can work as a standalone or integrated with a framework like scriptless/ BDD/ no code test automation frameworks. It will use LLM to generate test cases that the framework would accept.

## Market trends

**The global GenAI market size was valued at USD 16.87 in 2024 and is projected to grow at a CAGR of 37.6% from 2025 to 2030.**

**In 2022, the global generative AI market accounted for USD 10.6 billion, estimated to register the highest CAGR of 31.4%. It is expected to reach USD 151.9 billion by 2032 .**

# Problem statement

- To automate the end-to-end process from test requirements to reports using AI and GenAI algorithms
- To generate test scenarios, user stories/ test cases/scripts from the requirement documents through automation
- To execute the GenAI-generated test cases through a test automation framework with minimal reviewing efforts

# Solution

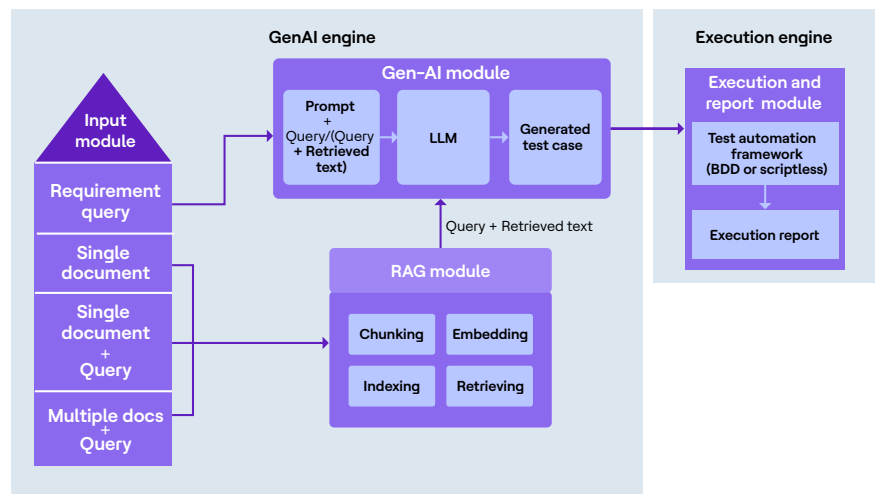


Figure 1: Block diagram

This solution transforms the requirements documents into test execution reports and is developed using four modules: the input module, the RAG module, the GenAI module inside the GenAI engine and the execution and report modules inside the execution engine. The block diagram provides the detailed flow of the end-to-end process.

The input module allows the user to provide input in various combinations like requirement query, single document, single document+ query and multiple doc+ query. The requirement query options enable users to upload the requirements in .xlsx or plain text containing one or bulk requirements to get processed. The single document options allow the user to upload a requirement document where the prompts are designed to generate test cases for the provided document with all possible scenarios. Single and multiple documents are supported in formats like .pdf, .docx, .txt, etc., along with queries expecting responses from a particular section or various sections of the documents. The requirements document tokens size is within the token limit supported by the LLM and can be directly passed using the requirement query option.

Users may face indexing errors when dealing with large requirement documents that exceed the token limit of LLMs (e.g., 4096 or 8192 tokens, including both input and output). This challenge is addressed using the

RAG module. In the case of test case generation for large documents, the RAG module divides the document into smaller chunks and processes each chunk individually with the LLM, combining the results from all chunks to provide a complete output. If a user needs to generate test cases for a specific section of the document, they can upload it along with a relevant query. The RAG module first chunks the document and then uses an embedding model to convert the chunks into a machine-readable format. These embeddings are indexed with an indexing model for faster and more effective retrieval. The user's query is indexed, enabling the system to find and retrieve the most relevant document chunk. These retrieved chunks are then processed by the LLM, along with the prompt and user query, to generate the desired test cases.

This process is also applicable to multiple documents, which are handled similarly through the RAG and GenAI modules, making it particularly useful for scenarios involving document referencing or substitution. The GenAI module receives the query along with retrieved text information from the RAG module or receives requirements directly along with the query and appends prompt information from the input module, where the content token limit is supported by LLM for processing and generating relevant responses/test cases.

The GenAI module is engineered with various prompts that support the generation of test cases based on user expectations. The LLM in the GenAI module is configured as a service that reduces the loading time of the model each time and helps users get swift responses. This GenAI module acts as an agent that efficiently converts test requirements into test cases with the standalone LLM. These generated test cases can be in a suitable format as required by the test automation framework for execution with a user review. These generated test cases are then sent to the execution and report module. The generated test case will be passed to a test automation framework like BDD or scriptless to execute and create test reports. In addition, a general chatbot is integrated to assist users with generic queries, script generations, etc. This solution can work individually to generate test cases from requirements in various formats, such as UI or can be integrated with test automation frameworks. For example, this can be directly integrated with test automation frameworks like No-code/scriptless and BDD, where the outputs can be directly passed for execution and reports. This will help overcome the manual efforts involved in developing test cases/user stories/test scenarios, from requirements to seamless execution.

## Case study

In the market, various test automation frameworks are being launched day to day. The input format for these frameworks varies, like gherkin steps, functional test steps, tabular format, etc. In this case study, the solution is integrated with a scriptless test automation framework, Falcon, an HCLTech IP that takes input in tabular format with simple english test steps containing si.no, test steps, test data and expected result. The integration of this solution with the Falcon test automation framework is termed Falcon Neo for this case study. When requirements are passed

through this integrated framework, the output test cases are generated in template format, which the framework accepts for test execution.

The following are the UI screenshots from the solution that portray how the inputs are given to get both functional test case outputs and Falcon Neo format test case outputs based on the test automation demand.

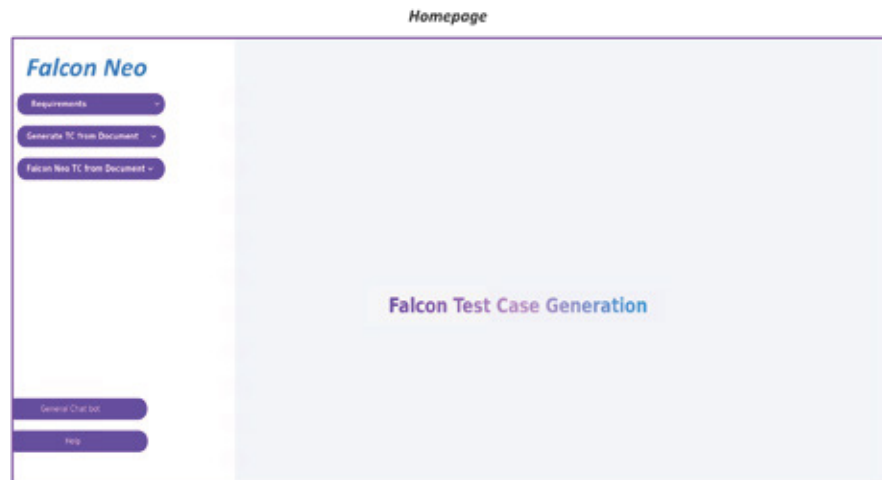


Figure 2

Figure 2 is the home page of the UI view for this solution. This contains options like requirements (conversion of plain requirements to test cases), generate TC from document (functional test case creation from single or multiple documents), Falcon Neo TC from document (Falcon Neo format, i.e., table format test case creation from single or multiple documents), general chat bot for general exploration and Help option helps the user with user guide. The Falcon Neo logo also works as a home button.

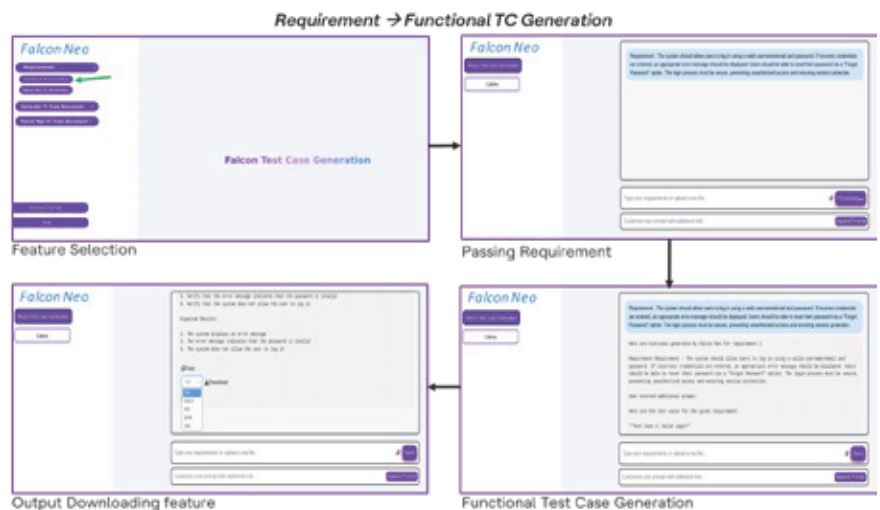


Figure 3

Above provided Figure 3 displays the entire flow in which this solution allows the user to pass a requirement in plain text and to get a functional test case generated. The user can download the output in different formats, such as .txt, .docx, .pdf, .json, .xml, etc. The solutions also allow

users to select the model from which they would like to generate outputs; this feature is explained in detail in further content.

A		B	
1	Customer	ABC	
2	Functional requirements	To validate all the options available in a web application	
3	Req Id	Requirement	
4		Requirement : The system should allow users to log in using a valid username/email and password. If incorrect credentials are entered, an appropriate error message should be displayed. Users should be able to reset their password via a "Forgot Password" option. The login process must be secure, preventing unauthorized access and ensuring session protection.	

Figure 4

The sample provided in Figure 4 above is the example requirement format to pass the requirement from the .xlsx file. It is required for the user to pass Excel in a format as in the figure, which is accepted by this solution.

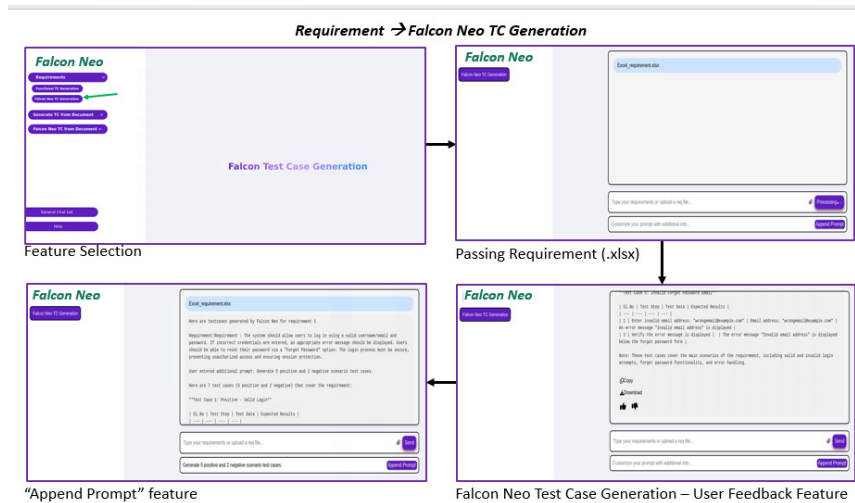


Figure 5

Figure 5 shows flow as an example of passing an Excel file as a requirement to generate the Falcon Neo Test case. The user can share feedback for the generated output using the thumbs-up/down buttons. Users can use the "Append Prompt" feature when there is a need to give certain conditions/rules to generate the current input or when the previously generated outputs need some modification. The user can provide their prompt through this feature and then pass the input to attain the modified test case to be generated.

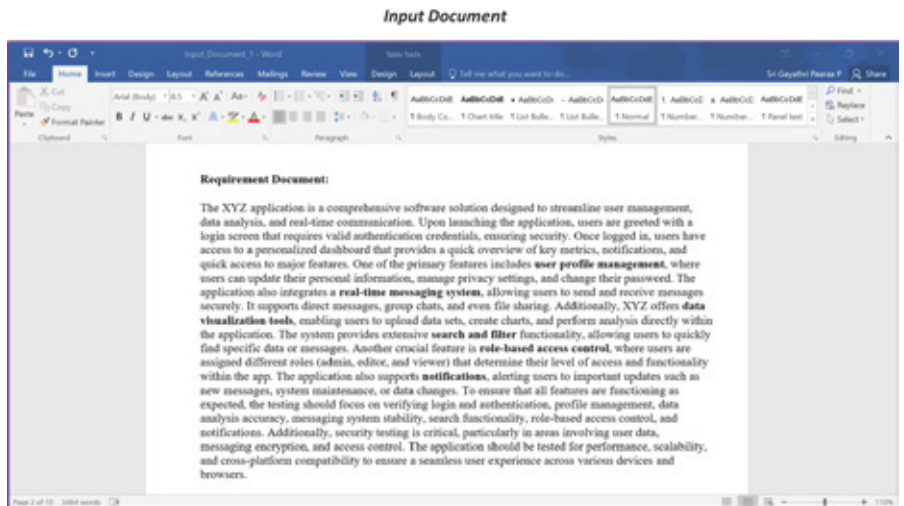


Figure 6

Figure 6 is the input document that can be used to generate a test case. Since the token limit might exceed the model's overall token limit, the RAG module will handle this challenge.

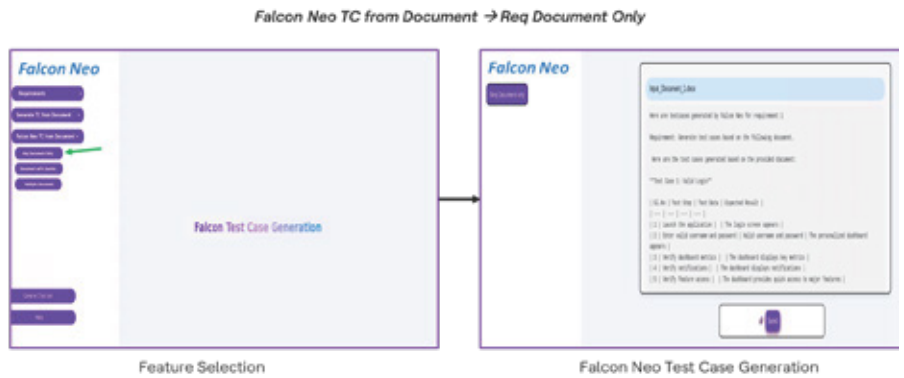


Figure 7

Figure 7 is the Falcon Neo Test Case Generation using a single document as input. As the input document contains more tokens, the input will be passed through the RAG algorithm and then passed through the GenAI module for test case generation.

The screenshot shows an Excel spreadsheet titled "Falcon Neo Output saved as .xlsx". The spreadsheet contains a table with test cases and their expected results. The columns are labeled "Test Case ID", "Test Case Description", "Test Data", and "Expected Result".

Test Case ID	Test Case Description	Test Data	Expected Result
TC-001	Valid login	Valid username and password	The login screen appears
TC-002	Valid login	Valid username and password	The personalized dashboard appears
TC-003	Valid login	Valid username and password	The dashboard displays key metrics
TC-004	Valid login	Valid username and password	The dashboard displays notifications
TC-005	Valid login	Valid username and password	The dashboard provides quick access to major features

Figure 8

Once the generated Falcon Neo Test Case is downloaded in Excel format, the output looks as in the above image. The test cases will be properly split into separate sheets. Test cases (sl.no. test step, test data and expected Result) will be properly aligned as per column order in the sheets. As this is the format accepted by the Falcon Test Automation Framework, the prompt is created to provide output in this format consistently so it can be passed directly to the framework with minimal human intervention only for review.

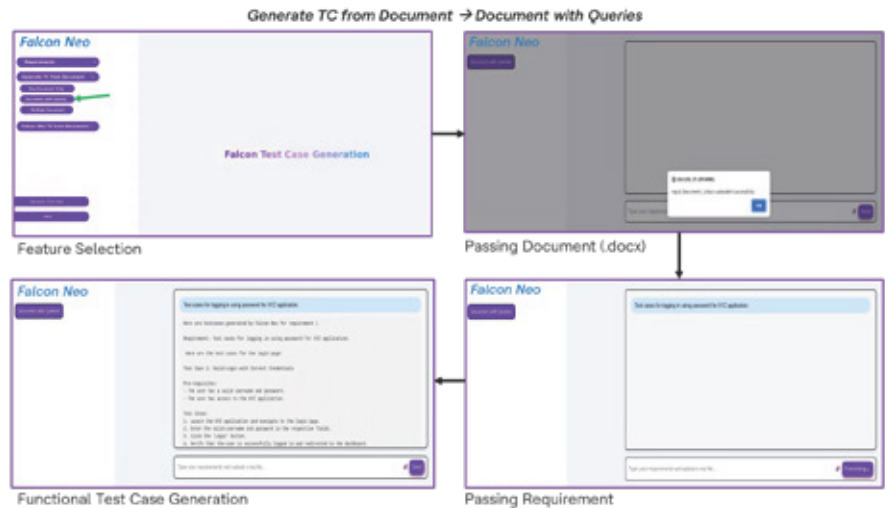


Figure 9

In Figure 9, the user can provide their requirement after uploading the input document. Using RAG algorithms, the text related to the query in the document will be retrieved to be generated into test cases.

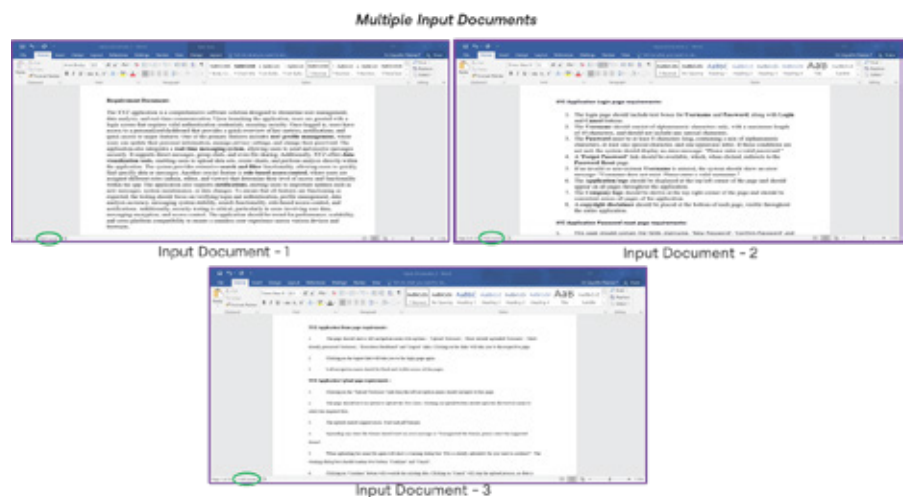


Figure 10

Figure 10 provides a sample of multiple input documents to be uploaded to the solution. Input document 1 has 3464 words, input document 2 has 5600 words and input document 3 has 11385 words; when passed these documents as input, the token limit of the LLM will be obviously exceeded. So, in situations like this RAG module helps with retrieving only the part related to the query passed and then passing it to the GenAI module for test case generation.



# Benefits

- The UI design makes this solution highly user-friendly
- Manual efforts required to convert requirements into test cases are significantly reduced
- LLM as a service reduces the model loading time
- Solution can work as a standalone or can be integrated into any test automation framework
- Reduces manual effort by 50 to 60% with accelerated speed by 2X times
- Large documents supported through RAG

# Conclusion

In conclusion, this solution utilizes GenAI and RAG techniques to automate the conversion of requirements into actionable test cases. The RAG methods allow users to upload large requirements documents to LLM for test case generation, which cannot be directly passed to an LLM. This will help retrieve the relevant information from the documents, simplifying the test case generation process. This allows the generated test cases to be reviewed and fed directly into test automation frameworks for seamless execution and results. This will reduce the manual efforts and streamline the process; it also accelerates the development and testing phases, enhancing overall productivity and efficiency.

The ability to generate test cases in various formats further tailors the output to meet specific needs and ensure higher accuracy and relevance in test scenarios. The entire end-to-end process of requirements to test reports can be automated using GenAI and traditional AI-integrated test automation frameworks.

# References

[Generative AI Market Size and Share | Industry Report, 2030](#)

[Generative AI Market Size to hit USD 151.9 Bn by 2032](#)

[What is Retrieval-Augmented Generation \(RAG\) ? - GeeksforGeeks](#)

[What is a Large Language Model \(LLM\) - GeeksforGeeks](#)

[Text generation](#)

# Author information



## **Narender S**

Narendra holds a Master's degree and has over 20 years of experience in software engineering. He has worked in various domains of product and sustenance engineering for several years. He has extensive experience designing and developing next-generation GenAI-based automation solutions to accelerate the QA journey and enhance quality.



## **Srihari V**

Srihari has worked in Telecom and networking for the past 20 years. He has managed various testing teams and created next-gen solutions as value-added for leading OEM clients. He is currently part of the Solutions team and generates AI-based solutions to support business needs.



## **Nidhin A Unnithan**

Nidhin is an AI Engineer with a Master's degree in Computational Engineering and Networking (COE) and six years of experience in machine learning and deep learning, including work experience in image processing, NLP, LLM and other AI-related fields. He is currently developing solutions based on Gen AI and LLM.



## **Sri Gayathri Paaraa P**

Gayathri holds a Computer Science and Engineering degree and has accumulated three years of experience in AI/ML, Image Processing, Deep Learning, Computer Vision and GenAI.



## **Ravi Tanniru**

Ravi is a Solution Architect working with HCLTech. He currently owns the design and development of CoE (Centre of Excellence) Solutions, viz Falcon and LAMS. He has designed/implemented test solutions across multiple domains, such as Banking, Finance and Insurance. He actively engages with the delivery teams to identify test improvement opportunities and adopt test optimization practices through test automation solutions.



## **Harandra J B**

Harandra is a Tech-Bee candidate pursuing a degree in computing and design. He has one year of experience working with UI frameworks, AI and ML.

# HCLTech | Supercharging Progress™

HCLTech is a global technology company, home to more than 223,000 people across 60 countries, delivering industry-leading capabilities centered around digital, engineering, cloud and AI, powered by a broad portfolio of technology services and products. We work with clients across all major verticals, providing industry solutions for Financial Services, Manufacturing, Life Sciences and Healthcare, Technology and Services, Telecom and Media, Retail and CPG and Public Services. Consolidated revenues as of 12 months ending March 2025 totaled \$13.8 billion. To learn how we can supercharge progress for you, visit [hcltech.com](https://hcltech.com).

[hcltech.com](https://hcltech.com)

