GenAI @ Xoriant
iQEAssist

Sep 2024

iQEAssist

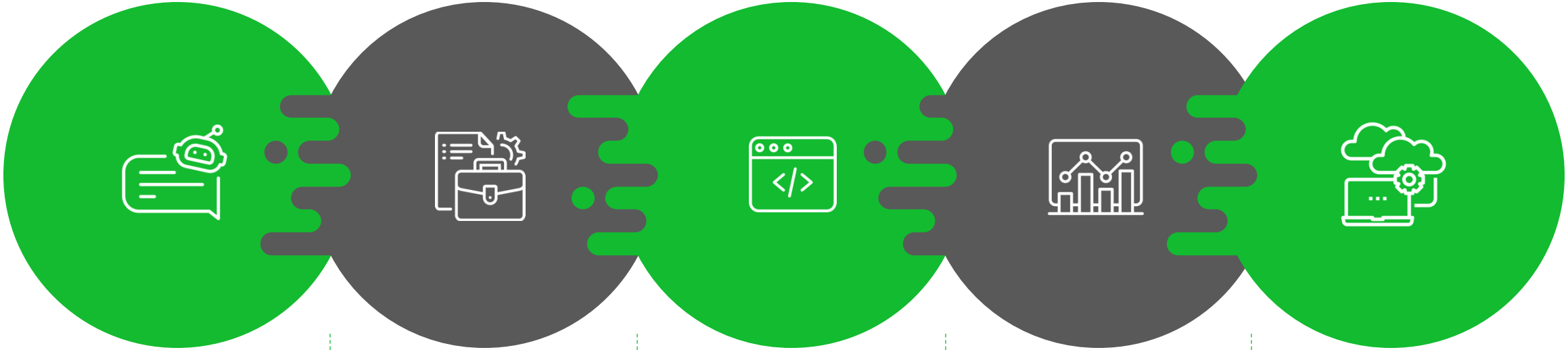
1. GenAI based solution for common testing activities

2. Testing activities
    1. Creation of feature files in gherkin syntax
    2. Finding requirement gaps
    3. Generate functional testcases
    4. Generate E2E testcases
    5. Generate API tests
    6. Generate Test data
    7. Generate Automation testcases
    8. Generate Bug reports
    9. Automate code review
    10. Testcase Optimization

3. Secure Azure native architecture.

4. Ensures data privacy throughout compared to other AI solutions.

iQEAssist Product features



Prebuilt prompts & agents to achieve common QE tasks

10+ file formats for requirement supported: txt, docx, excel, csv, pptx, pdf, html , png , feature

All major coding file formats supported

Data connectors for Gitlab, JIRA, Confluence supported

Azure Native architecture for enhanced security and scaling

Only Microsoft Entra id users permitted access

Achieve accelerated Shift Left Testing with iQEAssist

Xoriant offers an effortless solution to address the common QE activities leveraging GenAI capabilities

Requirement engineering by automatically discovering user stories from your documents using BDD

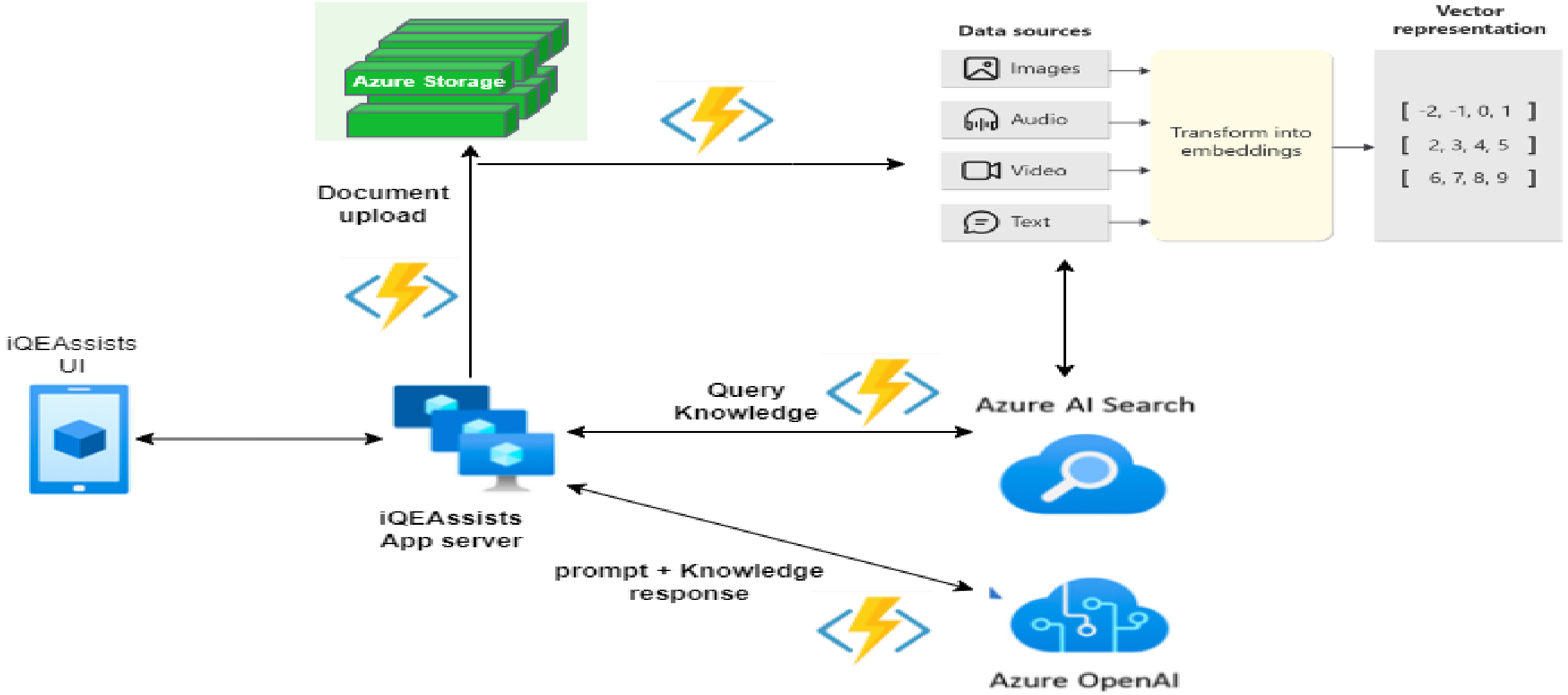
Requirement validation to suggest potential gaps and ambiguities

Generate functional/E2E test scenarios from user stories

Analyze Testcases to identify missing coverage

Automation framework code generation and migration

iQEAssist Architecture



Our Success Story Snapshot

Global Leader in business cloud software products for companies in industry specific markets

Problem: Transitioning an existing automation framework with 2000+ testcases, employing Selenium with Java and TestNG, to a Behavior-Driven Development approach, but lacking sufficient product documentation tailored for BDD practices

iQEAssist Solution: iQEAssist facilitated the seamless transition from traditional test automation to a Behavior-Driven Development (BDD) approach by transforming feature-wise automation scripts into BDD scenarios. The solution successfully developed a proof-of-concept that leveraged existing test automation reports and the Functional Requirements Document (FRD) to automatically generate comprehensive documentation. This documentation outlined system behavior and scenario descriptions, aiding in the creation of accurate BDD scenarios. Additionally, iQEAssist integrated with GPT-4o LLM to enhance the automatic generation of BDD scenarios from existing code.

Leading Provider Of Intelligent Compliance Solutions

Problem: Requirement gaps pose a significant challenge because discrepancies between what is needed and what is currently available or understood. These gaps can lead to misunderstandings, delays, and inefficiencies in project execution and resulted in project failure, budget overruns, and dissatisfaction among stakeholders.

iQEAssist Solution: By integrating GPT-4o LLM, iQEAssist was able to automatically identify and highlight requirement gaps in real-time. Leveraging the capabilities of GPT-4o, the platform could not only pinpoint discrepancies but also regenerate BDD scenarios and perform feature mapping, ensuring comprehensive test coverage and traceability. This integration resulted in 100% coverage of requirements, where each new change request was intelligently mapped to the correct set of test assets. As a result, test execution efforts were significantly reduced, streamlining the process and improving overall project efficiency and success.

5-sprint plan for BDD adoption by a scrum team

1 BDD Orientation

- Adopt 3-amigos practice for better collaboration
- Demonstrate visible progress with example mapping sessions
- Coaching sessions on requirements best-practices with GWT (given-when-then) structure of Gherkin languate
- Tools setup is complete for all stakeholders

2 Scenarios based requirements

- 3-amigos write requirements in Gherkin language
  - Practice story-mapping and example-mapping techniques
  - Maintain link between feature files and requirements in ALM tool
- Developers write code with TDD approach to develop code to meet expected behaviors

3 Testable Requirements w/ test data

- Team uses example mapping to write comprehensive feature requirements
  - Special attention given to boundary value conditions and error conditions
  - Feature files stored in VCS along with the code
- Developers write code with traceability between Unit Tests and BDD features; tests executed through CI pipeline
- Version Control System is used to maintain feature files history

4 Incorporate NFRs

- Incorporate some of the NFRs (e.g. SAST controls, response time thresholds, code metrics thresholds) into the feature files
- Adopt naming themes / conventions for tagging features to identify different levels of test suites (smoke tests, regression tests, special situations etc.)
- Measure test coverage and code coverage metrics based on the execution of feature files

5 Data Driven Continuous Improvements

- Identify areas of improvements based on metrics such as code coverage, test coverage and change frequency
- *Set up dashboards and notifications to measure and publish process performance, historical trends and action triggers*

Cost: $15,000 for 1 resource – 5 sprints

xoriant


xoriant
Imagination Realized
IMAGINATION
REALIZED