Optimizing Data Creation For CMS

Overview:

The client is a leading broadcaster of news and intelligence for specialized businesses, with a strong focus on the finance sector. They have been in the industry for 28 years, establishing themselves as a premier source of news and intelligence. Through innovative content management and a comprehensive suite of publications, they cater to a global audience, supporting the business acumen of hundreds of thousands of readers around the globe.

We have provided the client with complete testing solutions for all their products, ensuring the highest level of quality and reliability in their offerings. With the introduction of a new release, the client faced the challenge of having to create a vast amount of data necessary for automation testing. This process was anticipated to be time-consuming and resource-intensive. To address this, we developed GraphQL Test Data Injection for Automation Testing, a time-efficient data creation method that significantly reduces the time required for testing and enables a smoother, faster release process.

Client Details:

Name: Confidential | **Industry:** Publishing | **Location:** USA

Technologies:

Python, Behave, GraphQL, Jenkins, Postman, Allure, GIT

Optimizing Data Creation For CMS

Project Description:

With the new software release, the client was required to generate a large volume of publishable data for automation testing. Generating this data through the user interface (UI) method was projected to be highly time-consuming and inefficient. To address this challenge, the client sought a more efficient method in order to facilitate data creation for automation testing.

Solution: Leveraging our existing understanding of the products, we proposed an alternative solution utilizing "GraphQL Test Data Injection." i.e. GraphQL for backend data creation. This approach was selected for its ability to efficiently handle large volumes of data and its compatibility with Behavior-Driven Development (BDD) methodologies.

Our solution involved the following steps:

1. **Backend Framework Analysis:** We began by thoroughly analyzing the client's backend infrastructure, which employs GraphQL for API operations. It included user authentication, data creation, updates, deletions, and publishing.

2. **Documentation and Manual Testing:** We meticulously documented our understanding of the backend framework and developed Postman scripts to manually validate the functionality of the APIs. This step ensured that our approach would be robust and reliable.

3. **Development of Python Modules:** We designed and implemented Python modules to automate critical functions, including:
    - **Login to CMS:** Automating user authentication processes including 2FA.
    - **Data CRUD Operations:** Automated creating, reading, updating, and deleting data. Each publishable content contains around 10 to 15 parameters, some of which have up to 10 different options to choose from while automating the data creation. This included attachments of different media types like images, polls, videos, slideshow, etc. This component represented the most intricate aspect of the automation process, as our goal was to achieve comprehensive automation for the client. We aimed to not only address the current needs but also ensure that the solution was scalable and adaptable for future requirements, thereby positioning the client for long-term success and flexibility in their testing processes.

- **Data Publishing and Email Options:** This step automated the publishing of content to the appropriate publications directly from the CMS. Following the publication, we ensured that the content was indexed in the database, taking into account its relationship with other content items. Additionally, an email notification was triggered based on the predefined options set during the automation process.

4. **Adding BDD layer on top:** These modules were then integrated into our existing end-to-end (E2E) testing framework. To enhance alignment with business requirements, we added a Behavior-Driven Development (BDD) layer on top of the Python modules.

By adopting this approach, we significantly reduced the time required for data creation, leading to streamlining of the automation testing process and improvement of the overall efficiency. This solution not only addressed the immediate challenge but also provided a scalable and maintainable method for future testing needs.

Optimizing Data Creation For CMS

Screenshots:

Reports :

1 Content

#6 Create article - API	Morning, FF, 2 September 2024	33s 633ms
#9 Create article - UI	qaautouser2, Fundfire, 1 September 2024	2m 09s

5 Content

#7 Create article - API	Morning, FF, 2 September 2024	42s 940ms
#9 Create article - UI	qaautouser2, Fundfire, 1 September 2024	6m 21s

10 Content

#8 Create article - API	Morning, FF, 2 September 2024	1m 12s
#9 Create article - UI	qaautouser2, Fundfire, 1 September 2024	13m 31s