iQEAssists Features and Usage Guide

Table of Contents

1. Introduction

GenAI based solution for common testing activities from finding requirement gap to bug reporting in testing cycle using secure Azure native architecture. It's secure Azure environment ensures data privacy throughout compared to other AI solutions.

2. Features of iQEAssists:

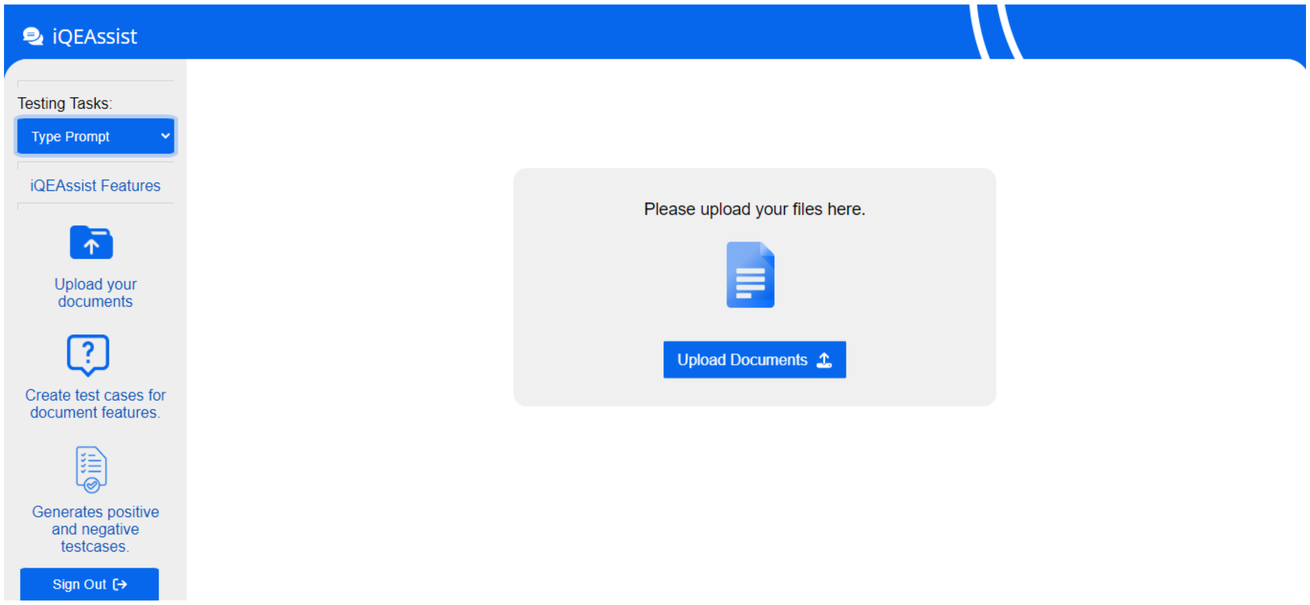
- Boost Shift left Testing with Requirement gap analysis and BDD scenario generation from user stories.
- Generates contextual test scenarios from diverse data sources for nuanced and comprehensive responses.
- Ensures security through Azure EntraID and storage, including Azure OpenAI model and Cognitive search.
- Based on user meta data will only show relevant data as per their privileges.
- iQEAssists is powered by GPT-4 providing best in class accuracy and utilizes RAG
- architecture to improve response relevance. Download all responses in csv/txt format. Accuracy between 70-80%.
- Supports all major document formats and code files which can be uploaded to iQEAssists to get contextual responses: Accepts input documents in variety of data formats like PDFs, CSVs, Text files, DOCX, EXCEL, PNG, HTML files. Code file formats java, python, c#, typescript and java script, Json, xml, feature files. In all 20 formats supported.
- Data from Confluence and Gitlab can be imported.
- Trained only on private documents and data stored securely in Vector DB Azure Cognitive Search.
- Inbuilt prompts for commonly available QA tasks ranging from testcase generation,
- requirement analysis, automation code generation, test coverage analysis, code review and documentation.
- Downloadable test artifacts in csv/txt format from chat.
- Convert code from one test automation framework to another automation framework with relevant context, explanation and comments.

3. Website

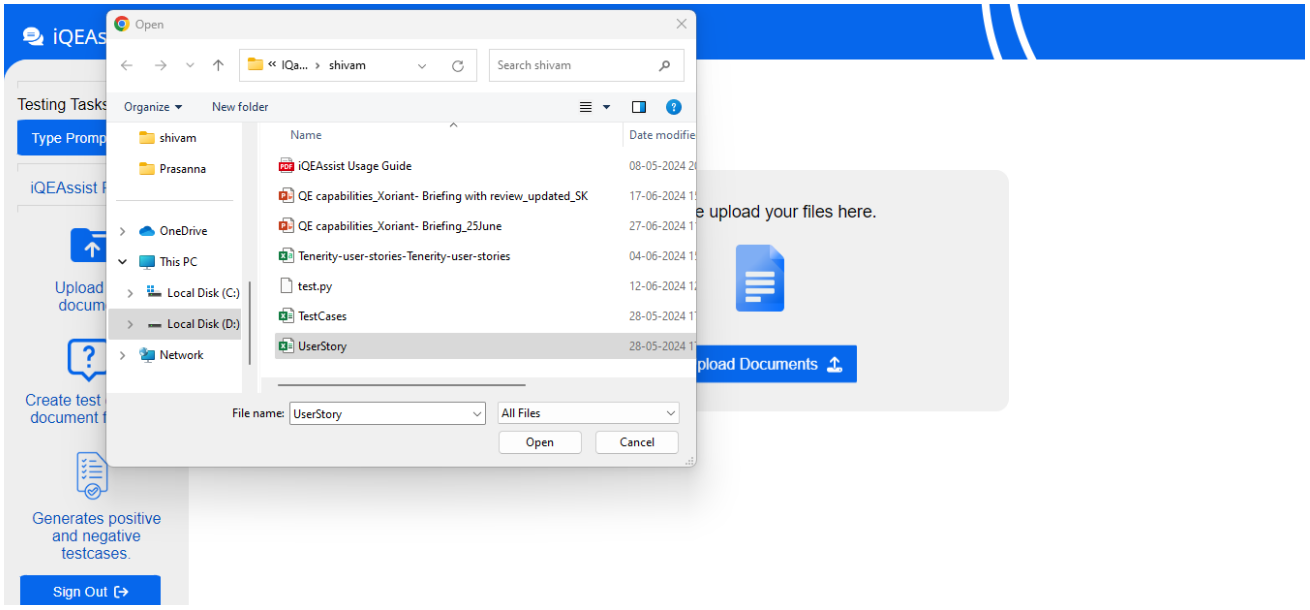
iQEAssists website link: https://iqe-assist-webapp.azurewebsites.net/

How to use iQEAssists:

1. After login this screen will be visible to you


iQEAssist
Testing Tasks:
Type Prompt
iQEAssist Features
Upload your documents
Create test cases for document features.
Generates positive and negative testcases.
Sign Out
Please upload your files here.
Upload Documents

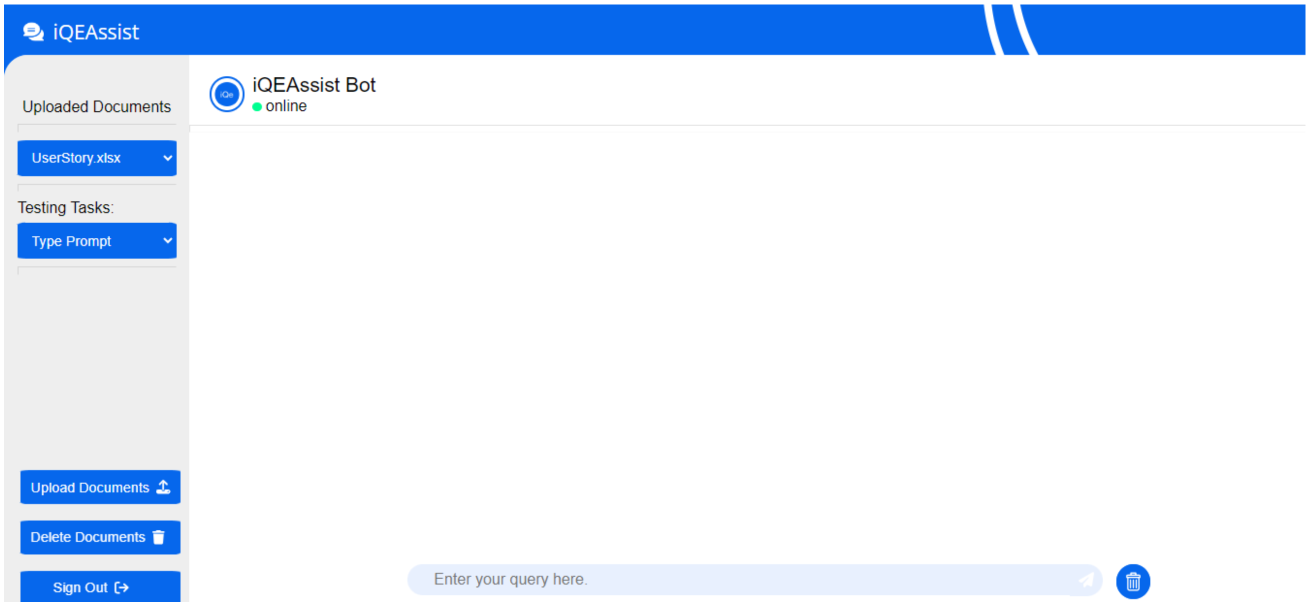
2. Upload your requirement documents which are in format txt, pdf, csv, docx or other supportedformats like FRD, U ser stories and Testcases.



Name	Date modifie
iQEAssist Usage Guide	08-05-2024 2
QE capabilities_Xoriant- Briefing with review_updated_SK	17-06-2024 1
QE capabilities_Xoriant- Briefing_25June	27-06-2024 1
Tenerity-user-stories-Tenerity-user-stories	04-06-2024 1
test.py	12-06-2024 1
TestCases	28-05-2024 1
UserStory	28-05-2024 1

File name: UserStory

3.After uploading a couple of documents, it will be available in main screen UploadedDocuments list:



4. Now we can select required Testing Task from Testing Tasks dropdown to generate responses based on our documents

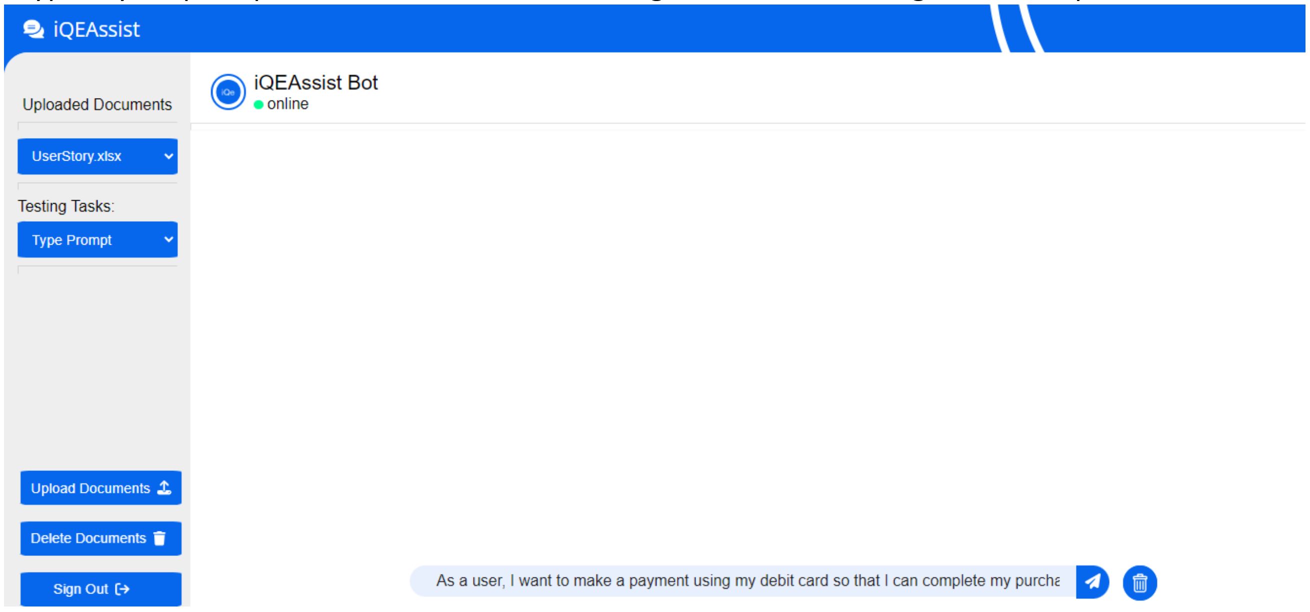
4. Usage Guide

4.1 Type Prompt

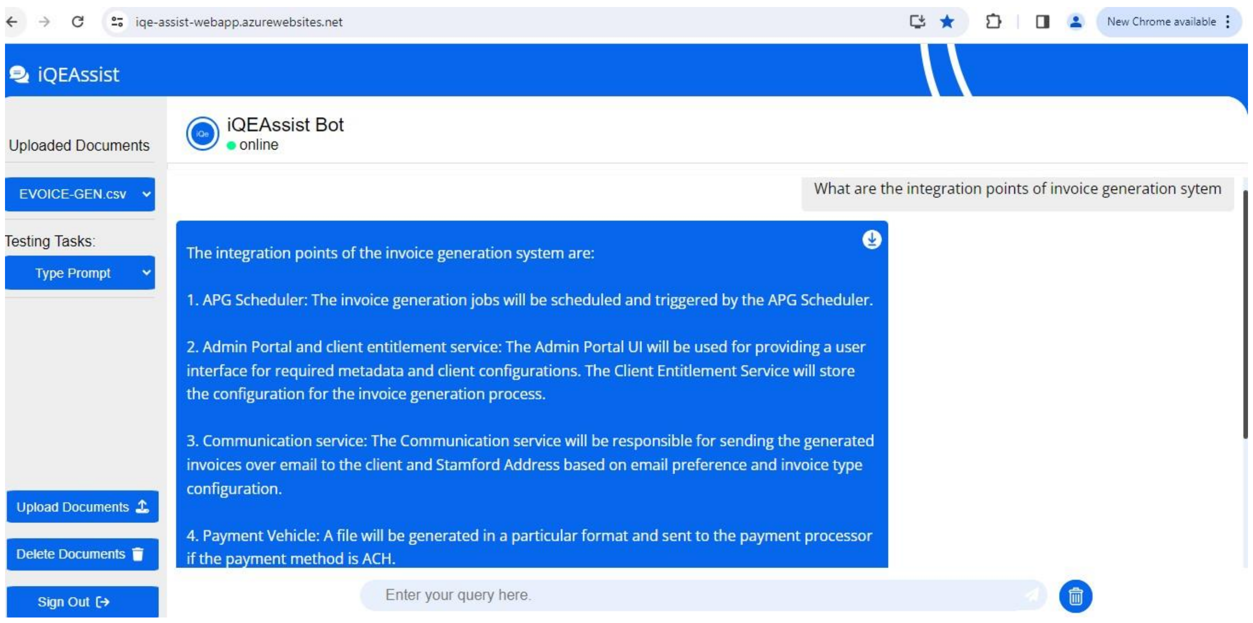
Here user can type in their own question or design prompt as per their requirements to generate response:

Our uploaded document in screenshot has a FRD named Invoicing-High Level Design.pdf. Wewill ask questions to it based on the document as Q&A.

Type in your prompt in textbox and click on the right arrow button to generate responses**.**



As a user, I want to make a payment using my debit card so that I can complete my purcha



What are the integration points of invoice generation sytem

The integration points of the invoice generation system are:

1. APG Scheduler: The invoice generation jobs will be scheduled and triggered by the APG Scheduler.

2. Admin Portal and client entitlement service: The Admin Portal UI will be used for providing a user interface for required metadata and client configurations. The Client Entitlement Service will store the configuration for the invoice generation process.

3. Communication service: The Communication service will be responsible for sending the generated invoices over email to the client and Stamford Address based on email preference and invoice type configuration.

4. Payment Vehicle: A file will be generated in a particular format and sent to the payment processor if the payment method is ACH.

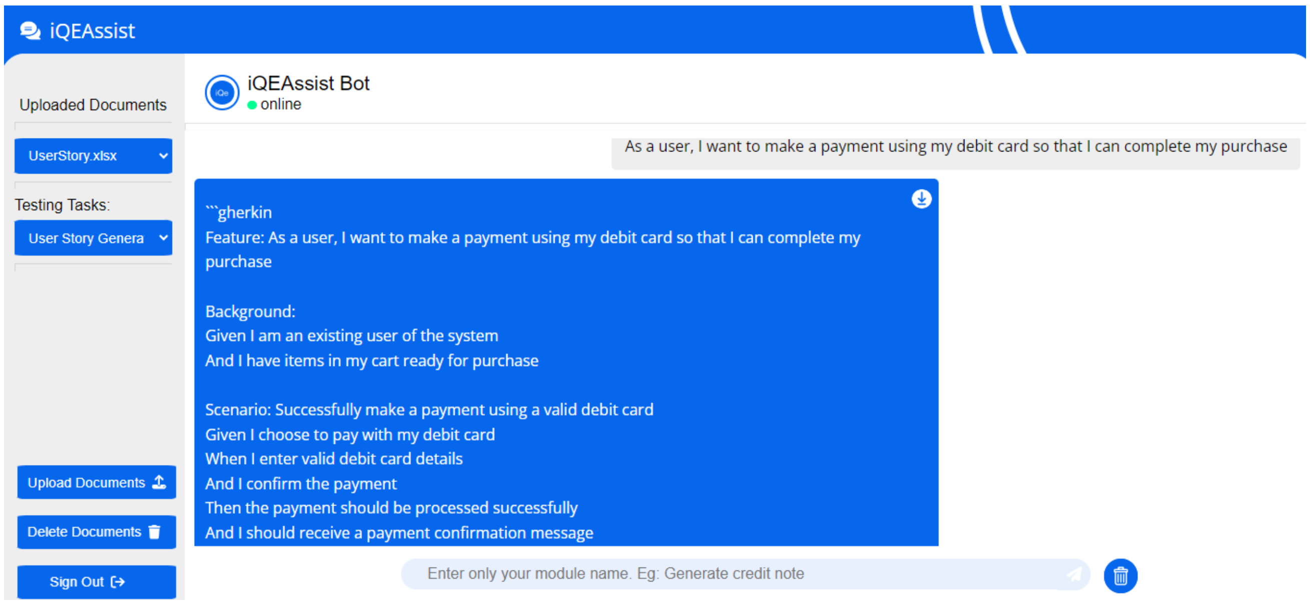
4.2 User Story Generation-BDD

Prerequisite: Upload User stories in iQEAssists

Steps: Select User Story Generate-BDD scenarios from dropdown and enter the user story name/requirement name in textbox and hit enter. We can also provide additional constraints separated by dot.

Example: Handle Email Receipt failure.

iQEAssists Output: Generates Feature file filled with all relevant scenarios for given user story

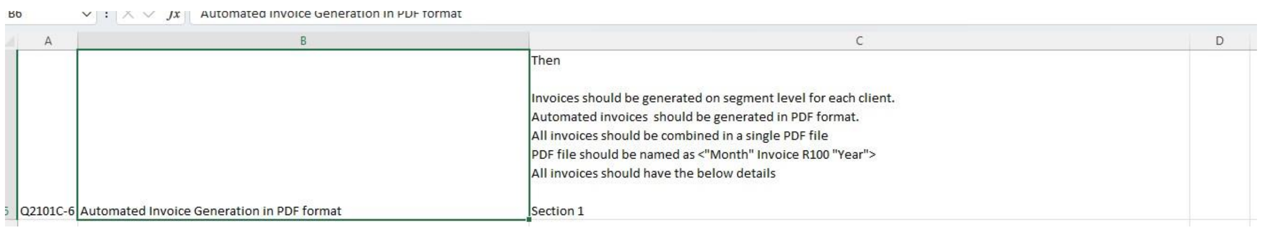


As a user, I want to make a payment using my debit card so that I can complete my purchase

~~~
```gherkin
Feature: As a user, I want to make a payment using my debit card so that I can complete my purchase

Background:
Given I am an existing user of the system
And I have items in my cart ready for purchase

Scenario: Successfully make a payment using a valid debit card
Given I choose to pay with my debit card
When I enter valid debit card details
And I confirm the payment
Then the payment should be processed successfully
And I should receive a payment confirmation message
~~~

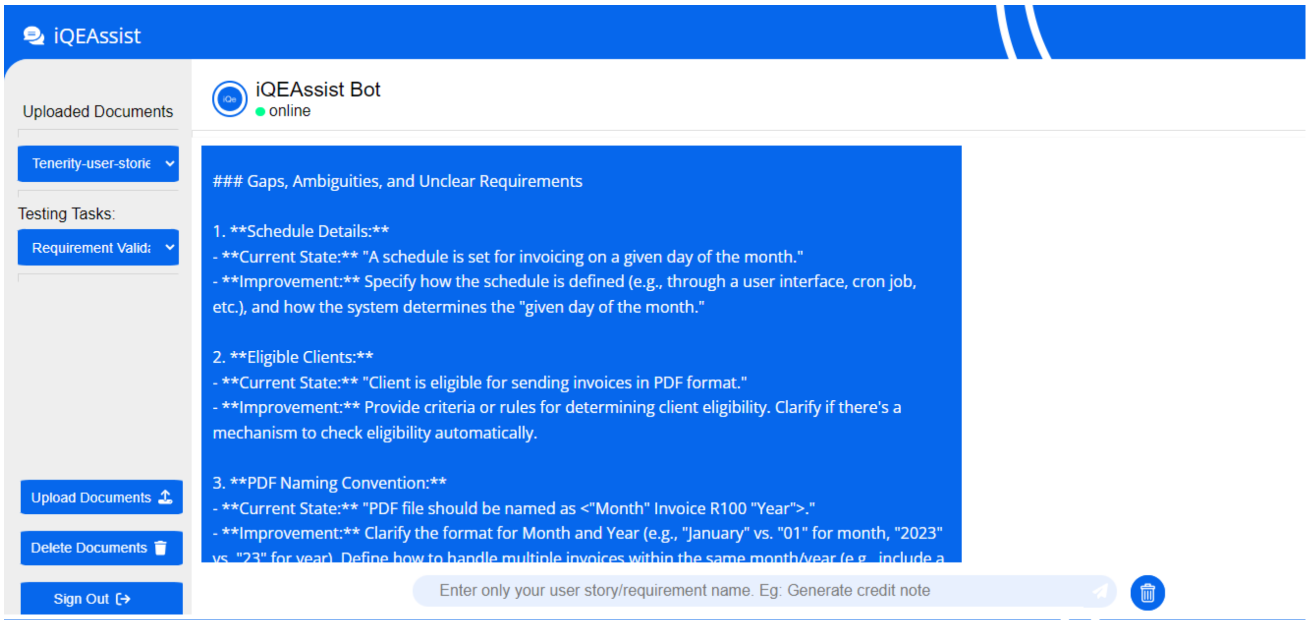
4.3 Requirement Validation

This is the user story along with its description in document Tenerity.csv



	A	B	C	D
5	Q2101C-6	Automated Invoice Generation in PDF format	Then Invoices should be generated on segment level for each client. Automated invoices should be generated in PDF format. All invoices should be combined in a single PDF file PDF file should be named as <"Month" Invoice R100 "Year"> All invoices should have the below details Section 1	

Enter this test case in prompt and select requirement validation in Testingtasks and hit enter

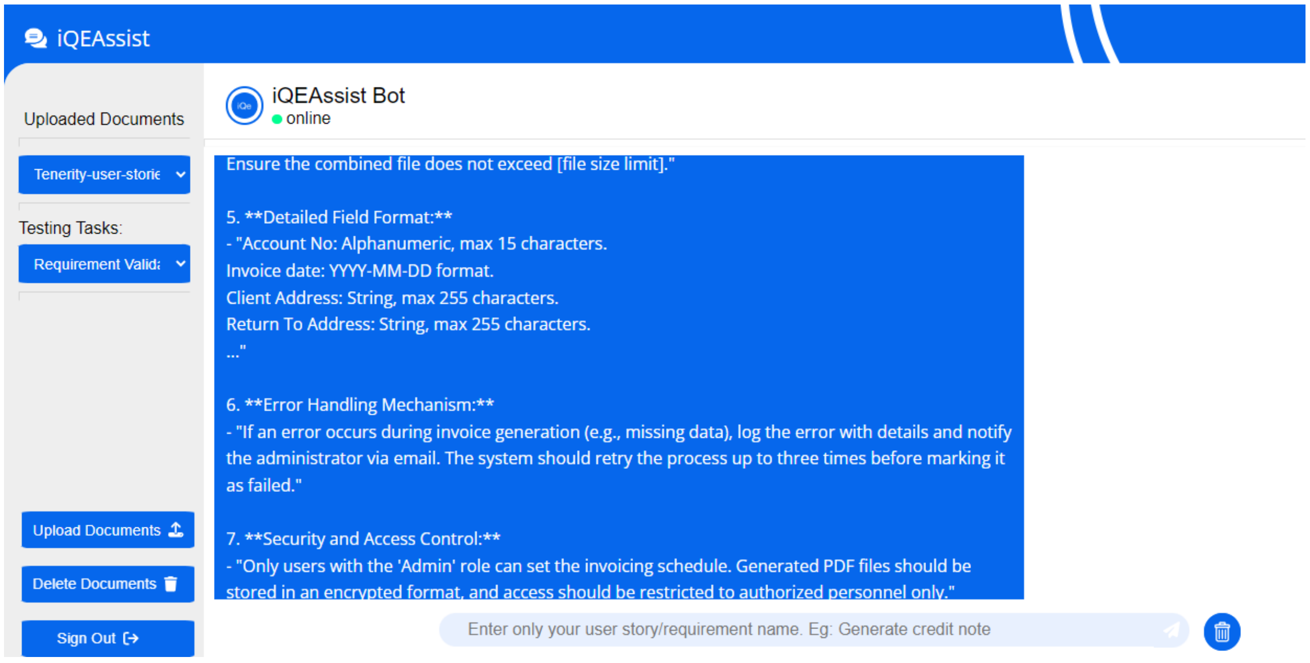


Gaps, Ambiguities, and Unclear Requirements

1. **Schedule Details:**
- **Current State:** "A schedule is set for invoicing on a given day of the month."
- **Improvement:** Specify how the schedule is defined (e.g., through a user interface, cron job, etc.), and how the system determines the "given day of the month."

2. **Eligible Clients:**
- **Current State:** "Client is eligible for sending invoices in PDF format."
- **Improvement:** Provide criteria or rules for determining client eligibility. Clarify if there's a mechanism to check eligibility automatically.

3. **PDF Naming Convention:**
- **Current State:** "PDF file should be named as <"Month" Invoice R100 "Year">."
- **Improvement:** Clarify the format for Month and Year (e.g., "January" vs. "01" for month, "2023" vs. "23" for year). Define how to handle multiple invoices within the same month/year (e.g., include a



Ensure the combined file does not exceed [file size limit]."

5. **Detailed Field Format:**
- "Account No: Alphanumeric, max 15 characters.
Invoice date: YYYY-MM-DD format.
Client Address: String, max 255 characters.
Return To Address: String, max 255 characters.
..."

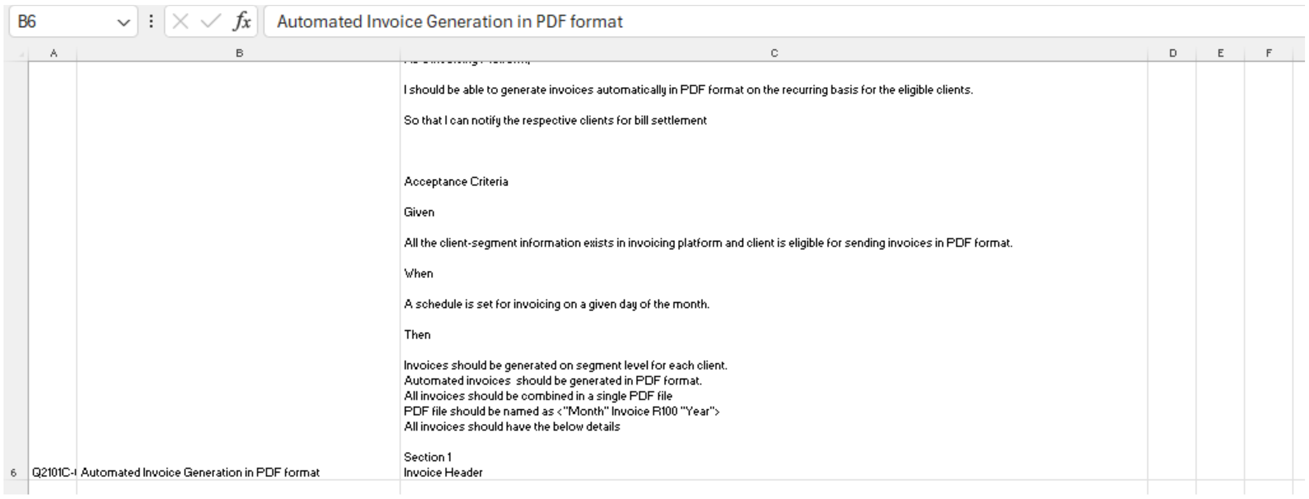
6. **Error Handling Mechanism:**
- "If an error occurs during invoice generation (e.g., missing data), log the error with details and notify the administrator via email. The system should retry the process up to three times before marking it as failed."

7. **Security and Access Control:**
- "Only users with the 'Admin' role can set the invoicing schedule. Generated PDF files should be stored in an encrypted format, and access should be restricted to authorized personnel only."

4.4 Generate Function Testcase

This is the manual test case along with its description in document Terenity-user-story.csv

Enter this test case in prompt and select Generate Function Testcase in Testingtasks and hit enter


B6 Automated Invoice Generation in PDF format

I should be able to generate invoices automatically in PDF format on the recurring basis for the eligible clients.

So that I can notify the respective clients for bill settlement

Acceptance Criteria

Given

All the client-segment information exists in invoicing platform and client is eligible for sending invoices in PDF format.

When

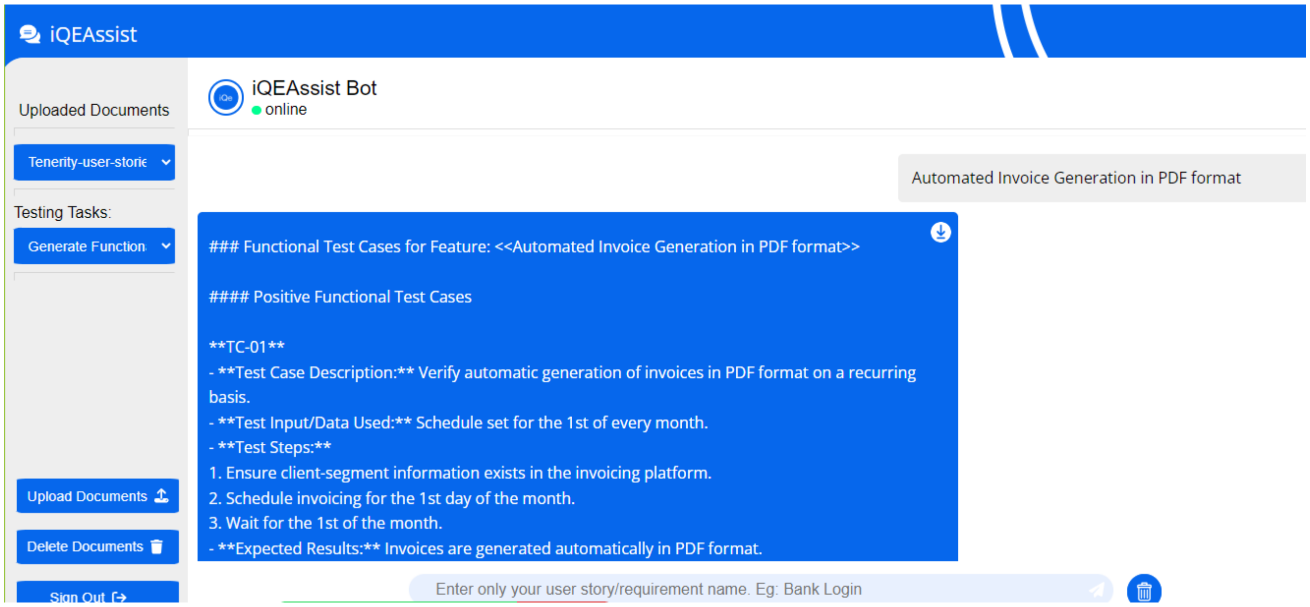
A schedule is set for invoicing on a given day of the month.

Then

Invoices should be generated on segment level for each client.
Automated invoices should be generated in PDF format.
All invoices should be combined in a single PDF file
PDF file should be named as <"Month" Invoice R100 "Year">
All invoices should have the below details

Section 1
Invoice Header

6 Q2101C-I Automated Invoice Generation in PDF format



Automated Invoice Generation in PDF format

Functional Test Cases for Feature: <<Automated Invoice Generation in PDF format>>

Positive Functional Test Cases

TC-01
- **Test Case Description:** Verify automatic generation of invoices in PDF format on a recurring basis.
- **Test Input/Data Used:** Schedule set for the 1st of every month.
- **Test Steps:**
1. Ensure client-segment information exists in the invoicing platform.
2. Schedule invoicing for the 1st day of the month.
3. Wait for the 1st of the month.
- **Expected Results:** Invoices are generated automatically in PDF format.

Enter only your user story/requirement name. Eg: Bank Login

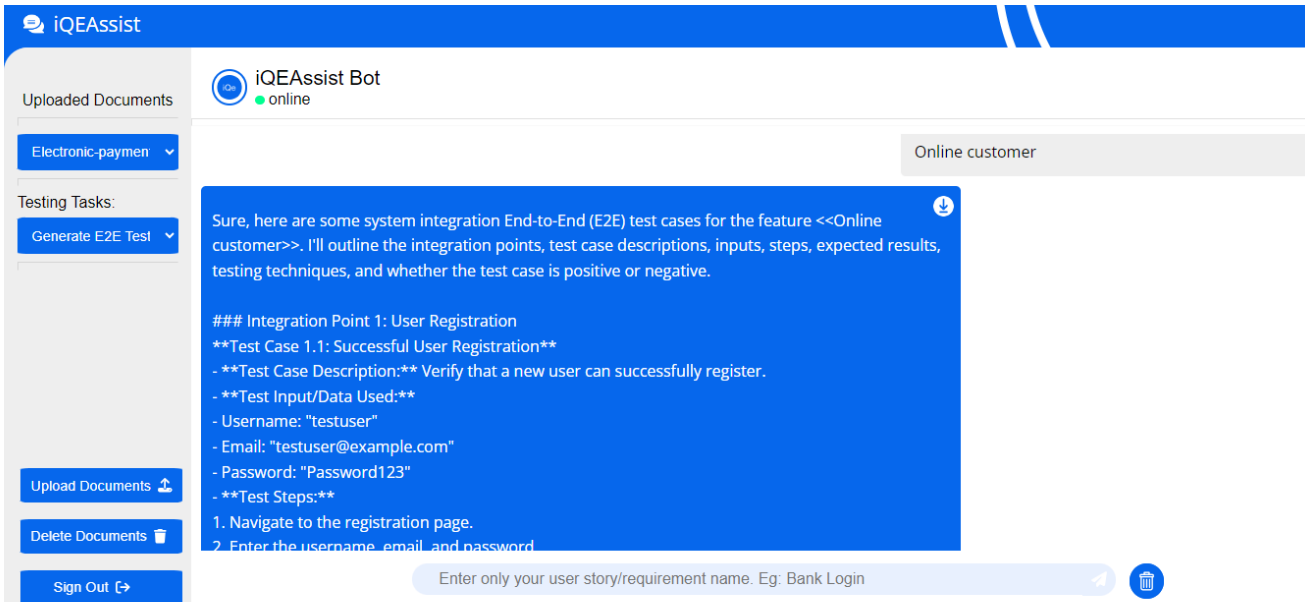
4.5 Generate E2E Testcases

Prerequisite: Upload Diagrams in iQEAssists

Steps: Select your Generate E2E Testcase scenarios from dropdown and enter the user requirement name in textbox and hit enter. We can also provide additional constraintsseparated by dot.

Example: Handle Email Receipt failure.

iQEAssists Output: Generates end to end test case with all relevant scenarios for given diagram.



Online customer

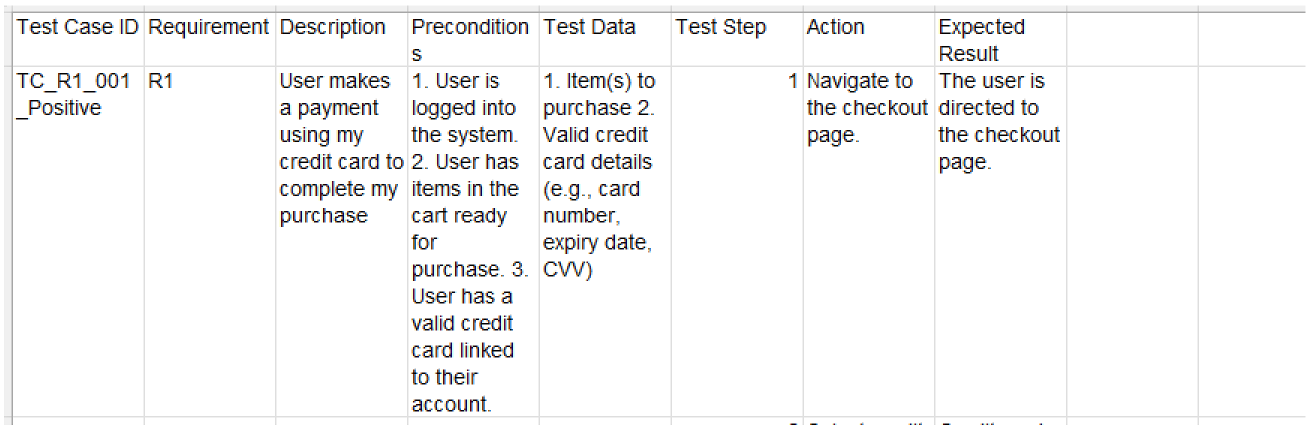
Sure, here are some system integration End-to-End (E2E) test cases for the feature <<Online customer>>. I'll outline the integration points, test case descriptions, inputs, steps, expected results, testing techniques, and whether the test case is positive or negative.

Integration Point 1: User Registration
Test Case 1.1: Successful User Registration
- **Test Case Description:** Verify that a new user can successfully register.
- **Test Input/Data Used:**
- Username: "testuser"
- Email: "testuser@example.com"
- Password: "Password123"
- **Test Steps:**
1. Navigate to the registration page.
2. Enter the username, email, and password

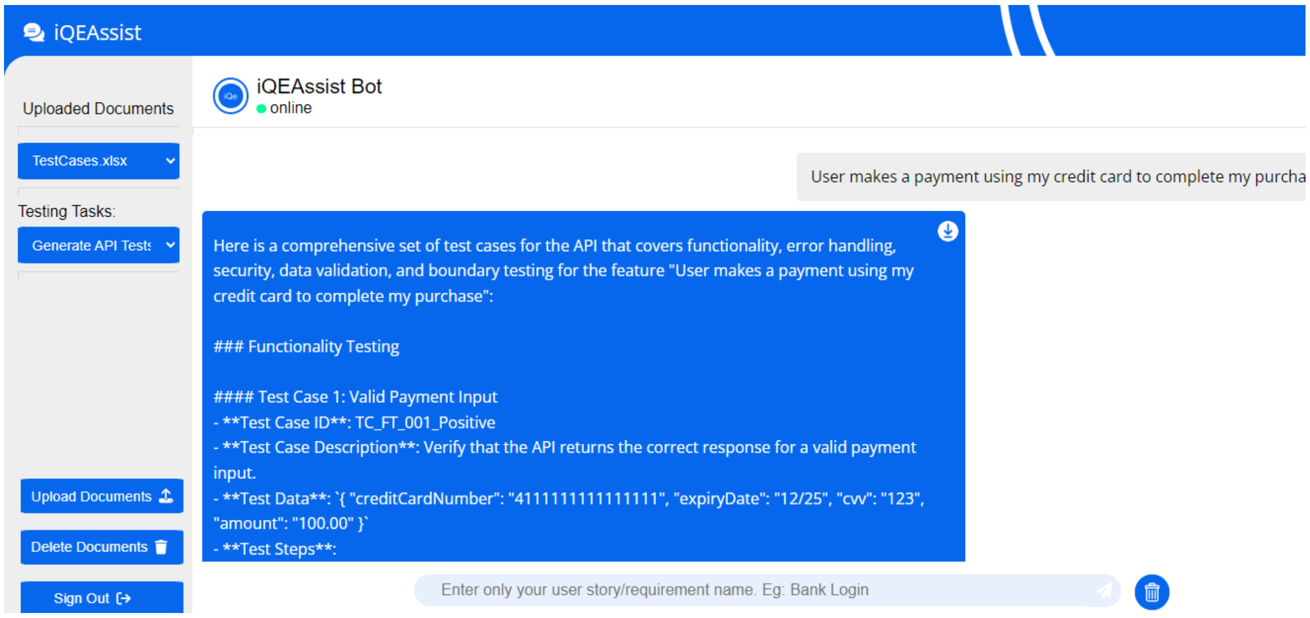
4.6 Generate API Tests

This is the manual test case along with its description in document TestCase.csv

Enter this test case in prompt and select Generate API Tests in Testingtasks and hit the enter



Test Case ID	Requirement	Description	Preconditions	Test Data	Test Step	Action	Expected Result
TC_R1_001_Positive	R1	User makes a payment using my credit card to complete my purchase	1. User is logged into the system. 2. User has items in the cart ready for purchase. 3. User has a valid credit card linked to their account.	1. Item(s) to purchase 2. Valid credit card details (e.g., card number, expiry date, CVV)	1	Navigate to the checkout page.	The user is directed to the checkout page.


iQEAssist

Uploaded Documents

TestCases.xlsx

Testing Tasks:

Generate API Tests

Upload Documents

Delete Documents

Sign Out

iQEAssist Bot
online

User makes a payment using my credit card to complete my purcha

Here is a comprehensive set of test cases for the API that covers functionality, error handling, security, data validation, and boundary testing for the feature "User makes a payment using my credit card to complete my purchase":

Functionality Testing

Test Case 1: Valid Payment Input
- **Test Case ID**: TC_FT_001_Positive
- **Test Case Description**: Verify that the API returns the correct response for a valid payment input.
- **Test Data**: `{ "creditCardNumber": "4111111111111111", "expiryDate": "12/25", "cvv": "123", "amount": "100.00" }`
- **Test Steps**:

Enter only your user story/requirement name. Eg: Bank Login

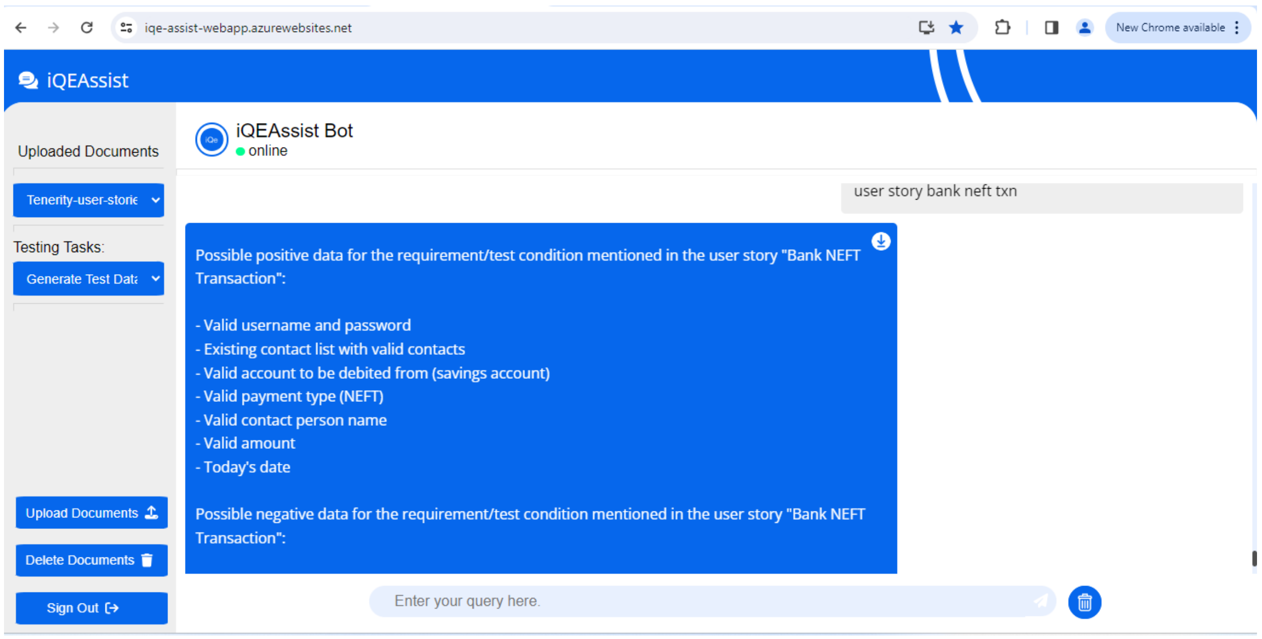
4.7 Generate Test Data

Prerequisite: Upload your detailed positive testcase so that iQEAssists generates accurate data combinations

Steps: Select your Generate Test Data from dropdown and enter the manual test name in textbox. Additional constraints can be added separated by a dot.

Example: Bank Login

iQEAssists Output: Generates detailed Positive and Negative data combinations.


iqe-assist-webapp.azurewebsites.net

New Chrome available

iQEAssist

Uploaded Documents

Tenerity-user-storie

Testing Tasks:

Generate Test Data

Upload Documents

Delete Documents

Sign Out

iQEAssist Bot
online

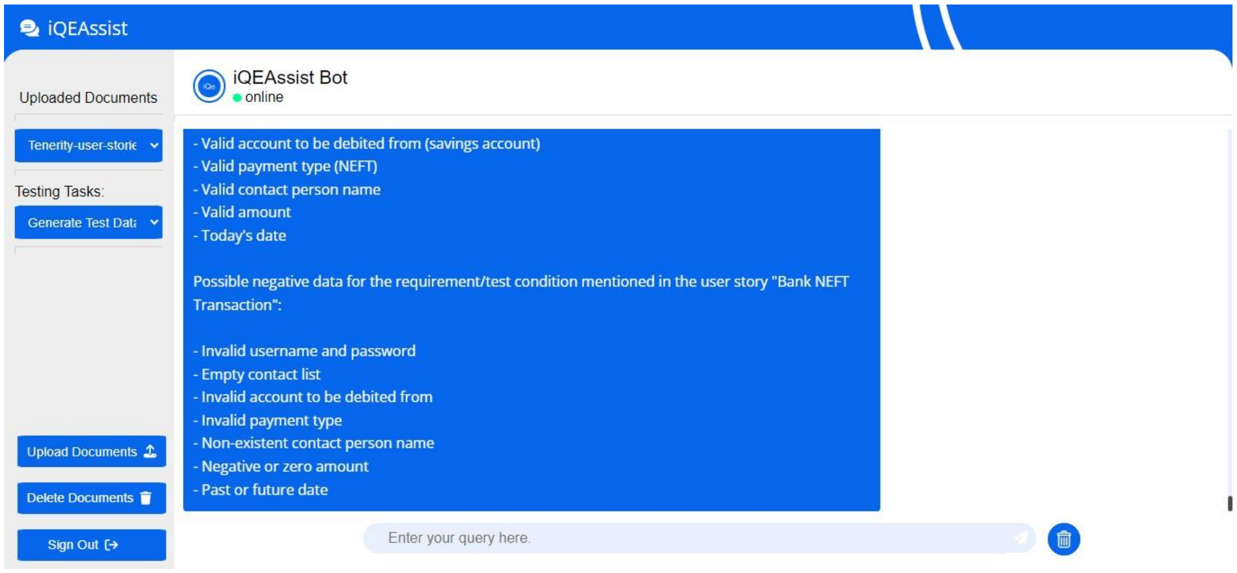
user story bank neft txn

Possible positive data for the requirement/test condition mentioned in the user story "Bank NEFT Transaction":

- Valid username and password
- Existing contact list with valid contacts
- Valid account to be debited from (savings account)
- Valid payment type (NEFT)
- Valid contact person name
- Valid amount
- Today's date

Possible negative data for the requirement/test condition mentioned in the user story "Bank NEFT Transaction":

Enter your query here.



- Valid account to be debited from (savings account)
- Valid payment type (NEFT)
- Valid contact person name
- Valid amount
- Today's date

Possible negative data for the requirement/test condition mentioned in the user story "Bank NEFT Transaction":

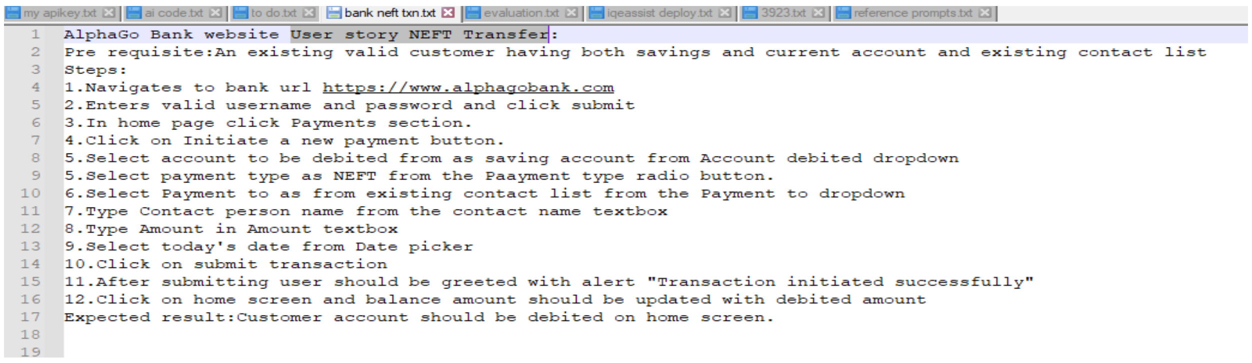
- Invalid username and password
- Empty contact list
- Invalid account to be debited from
- Invalid payment type
- Non-existent contact person name
- Negative or zero amount
- Past or future date

4.8 Generate Automation TC

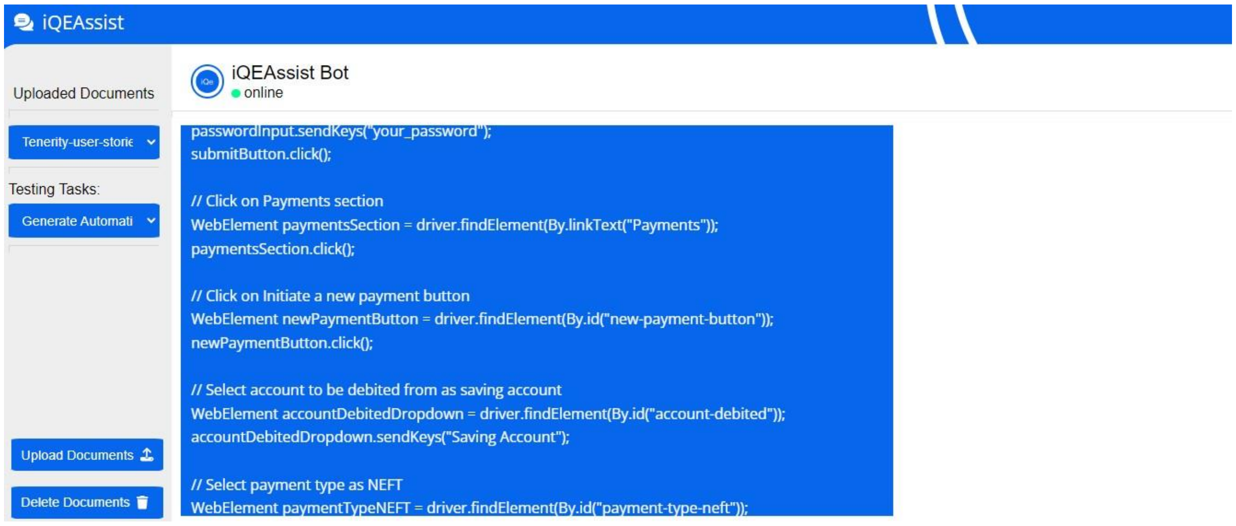
Prerequisite: For this detailed manual testcase containing step by step instructions to be performed on web/mobile browser/app needs to upload into iQEAssists.

Steps: Select Generate Automation TC from dropdown and in textbox write the manual testcase name and then separated by a # write the desired language the script should be generated.

For example: User Story NEFT Transfer#selenium java.



```
AlphaGo Bank website User story NEFT Transfer:
Pre requisite:An existing valid customer having both savings and current account and existing contact list
Steps:
1.Navigates to bank url https://www.alphagobank.com
2.Enters valid username and password and click submit
3.In home page click Payments section.
4.Click on Initiate a new payment button.
5.Select account to be debited from as saving account from Account debited dropdown
5.Select payment type as NEFT from the Paayment type radio button.
6.Select Payment to as from existing contact list from the Payment to dropdown
7.Type Contact person name from the contact name textbox
8.Type Amount in Amount textbox
9.Select today's date from Date picker
10.Click on submit transaction
11.After submitting user should be greeted with alert "Transaction initiated successfully"
12.Click on home screen and balance amount should be updated with debited amount
Expected result:Customer account should be debited on home screen.
```



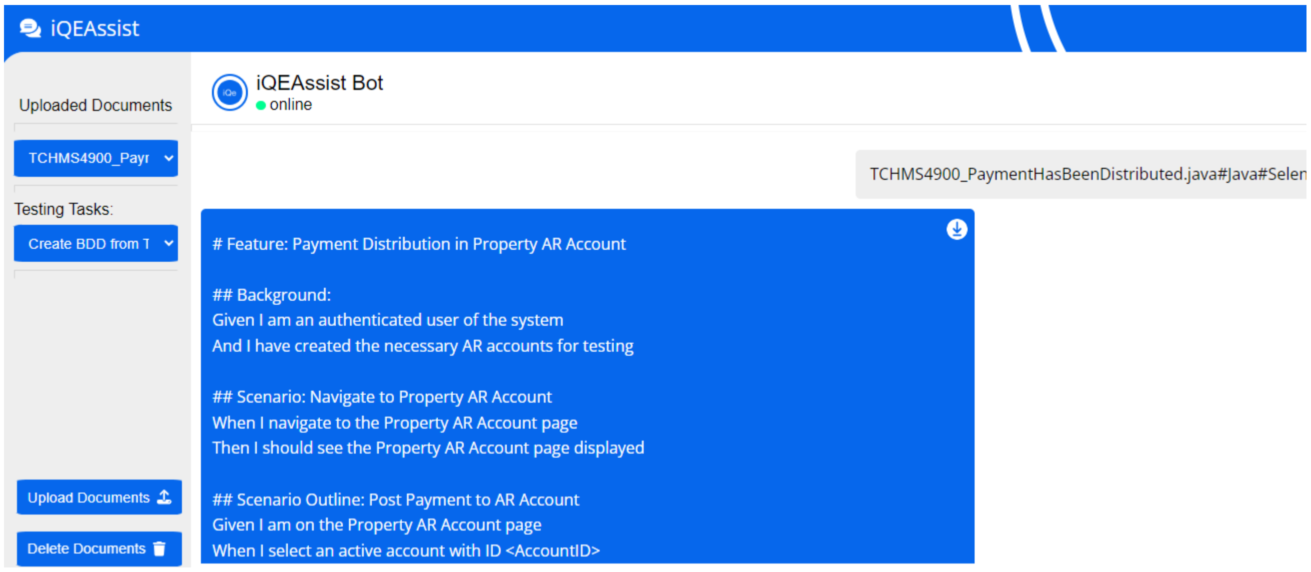
```
passwordInput.sendKeys("your_password");
submitButton.click();

// Click on Payments section
WebElement paymentsSection = driver.findElement(By.linkText("Payments"));
paymentsSection.click();

// Click on Initiate a new payment button
WebElement newPaymentButton = driver.findElement(By.id("new-payment-button"));
newPaymentButton.click();

// Select account to be debited from as saving account
WebElement accountDebitedDropdown = driver.findElement(By.id("account-debited"));
accountDebitedDropdown.sendKeys("Saving Account");

// Select payment type as NEFT
WebElement paymentTypeNEFT = driver.findElement(By.id("payment-type-neft"));
```

4.9 Create BDD from TDD

Prerequisite: Upload code file in iQEAssist

Steps: Select Create BDD from TDD from dropdown and in textbox write the file name and then separated by a # write the desired language the script should be generated.

For example: filename.java#Java#Selenium.



TCHMS4900_PaymentHasBeenDistributed.java#Java#Selen

```
# Feature: Payment Distribution in Property AR Account

## Background:
Given I am an authenticated user of the system
And I have created the necessary AR accounts for testing

## Scenario: Navigate to Property AR Account
When I navigate to the Property AR Account page
Then I should see the Property AR Account page displayed

## Scenario Outline: Post Payment to AR Account
Given I am on the Property AR Account page
When I select an active account with ID <AccountID>
```

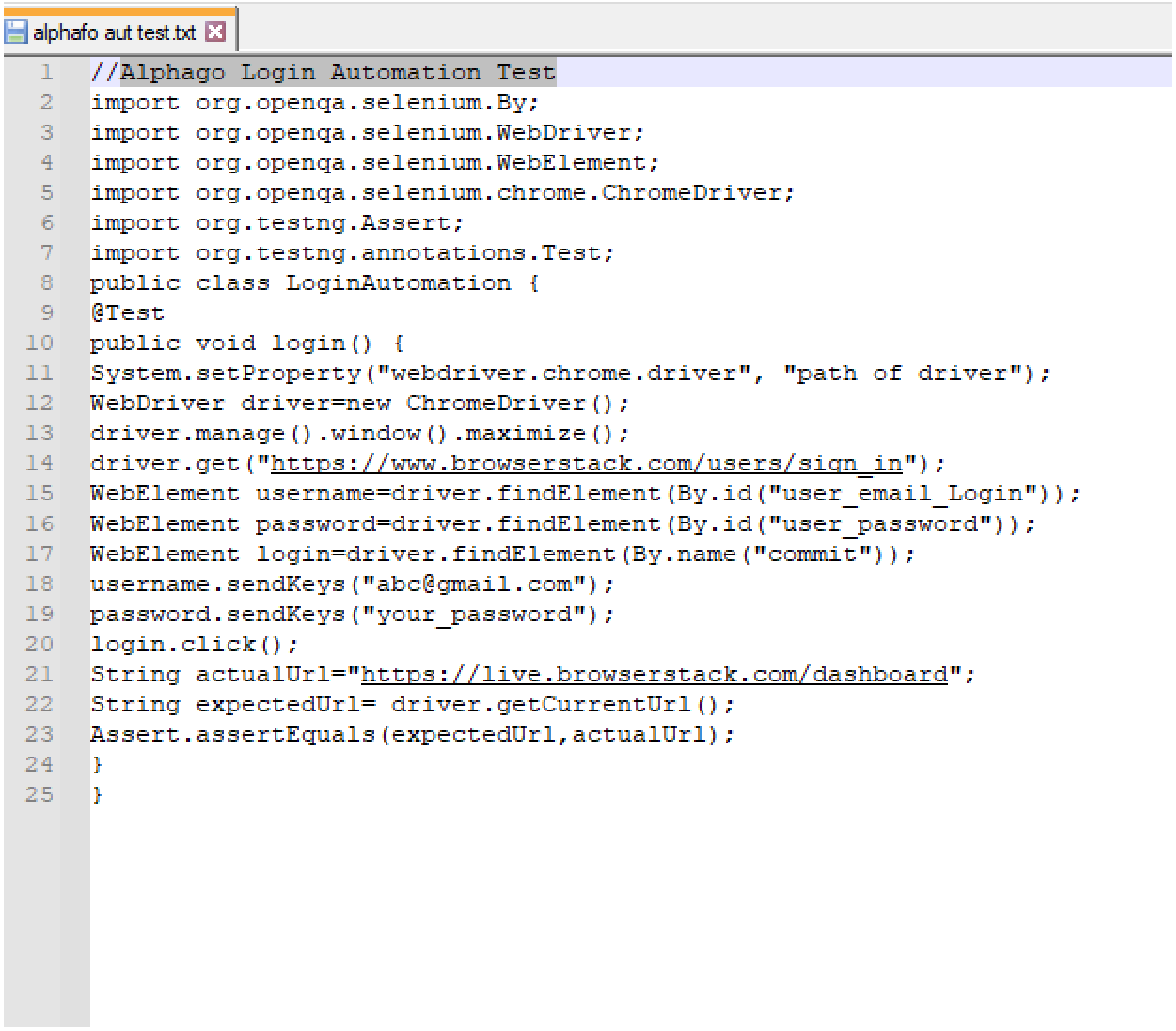
4.10 Automation Code Review

Prerequisite: Upload Automation Test case code in iQEAssists. Here in this example, I have created a text file containing automation code and the first line is commented andcontains testcase name.

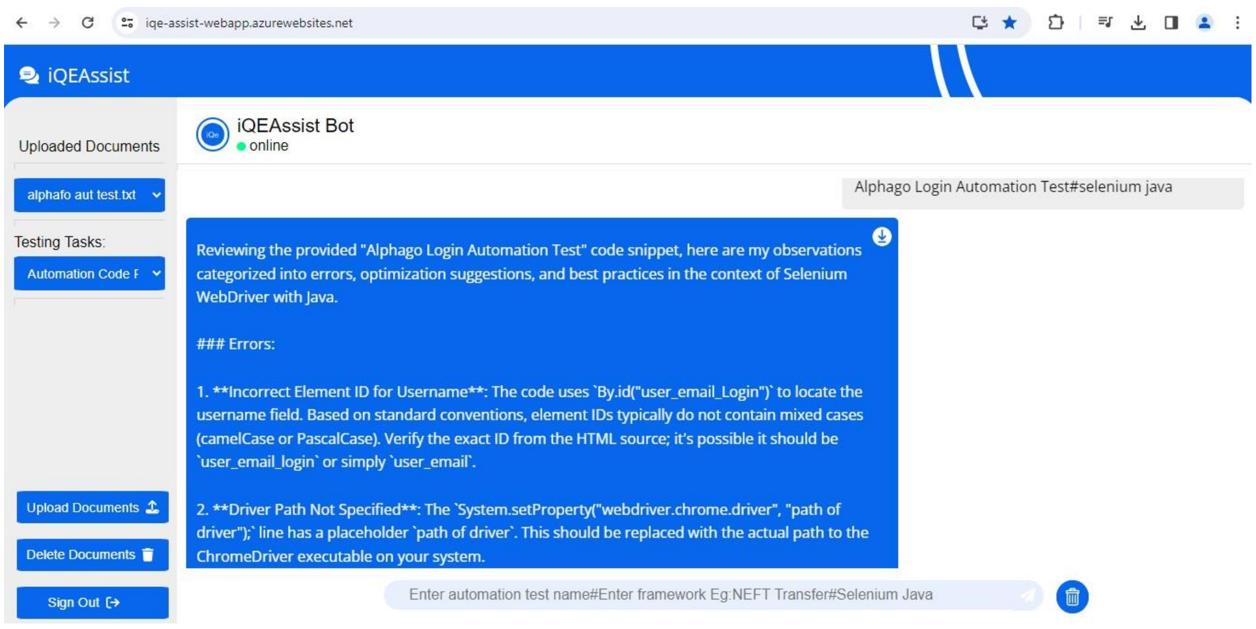
Steps: Select your **Automation Code Review** from dropdown and enter the automation testname in textbox and hit enter. Then enter the tech stack in which it should be reviewed likeselenium java.

Example: Alphago Login Automation Test#selenium java.

iQEAssists Output: Generates suggestions and improvements on code like Tech Lead.



```
//Alphago Login Automation Test
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.Test;
public class LoginAutomation {
@Test
public void login() {
System.setProperty("webdriver.chrome.driver", "path of driver");
WebDriver driver=new ChromeDriver();
driver.manage().window().maximize();
driver.get("https://www.browserstack.com/users/sign_in");
WebElement username=driver.findElement(By.id("user_email_Login"));
WebElement password=driver.findElement(By.id("user_password"));
WebElement login=driver.findElement(By.name("commit"));
username.sendKeys("abc@gmail.com");
password.sendKeys("your_password");
login.click();
String actualUrl="https://live.browserstack.com/dashboard";
String expectedUrl= driver.getCurrentUrl();
Assert.assertEquals(expectedUrl,actualUrl);
}
}
```


iqe-assist-webapp.azurewebsites.net

Alphago Login Automation Test#selenium java

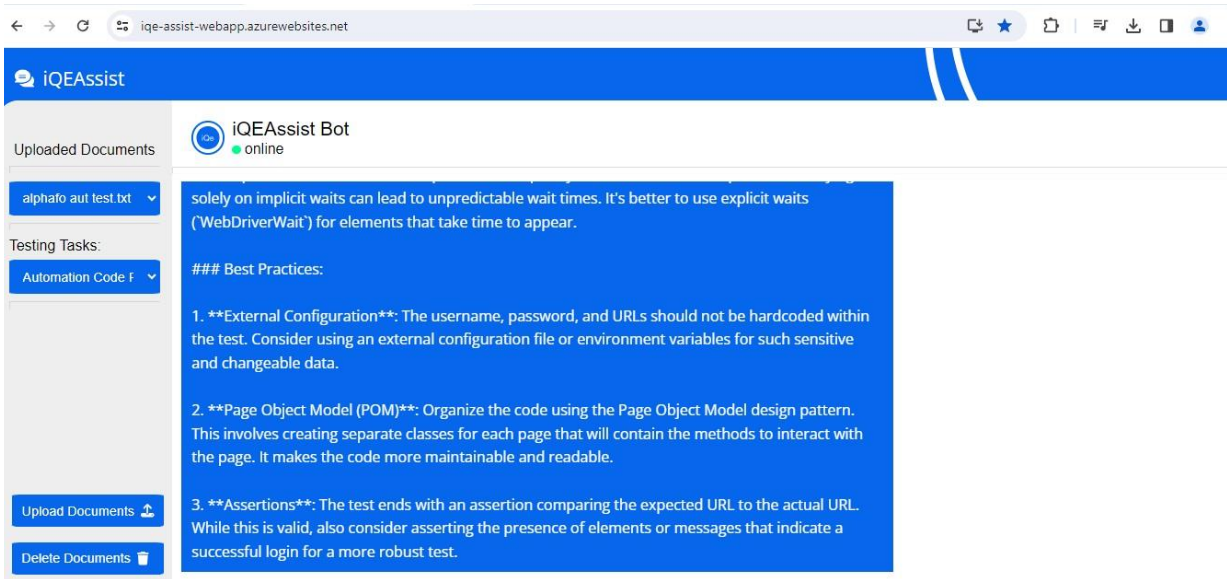
Reviewing the provided "Alphago Login Automation Test" code snippet, here are my observations categorized into errors, optimization suggestions, and best practices in the context of Selenium WebDriver with Java.

Errors:

1. **Incorrect Element ID for Username**: The code uses `By.id("user_email_Login")` to locate the username field. Based on standard conventions, element IDs typically do not contain mixed cases (camelCase or PascalCase). Verify the exact ID from the HTML source; it's possible it should be `user_email_login` or simply `user_email`.

2. **Driver Path Not Specified**: The `System.setProperty("webdriver.chrome.driver", "path of driver");` line has a placeholder `path of driver`. This should be replaced with the actual path to the ChromeDriver executable on your system.

Enter automation test name#Enter framework Eg:NEFT Transfer#Selenium Java


iqe-assist-webapp.azurewebsites.net

solely on implicit waits can lead to unpredictable wait times. It's better to use explicit waits (`WebDriverWait`) for elements that take time to appear.

Best Practices:

1. **External Configuration**: The username, password, and URLs should not be hardcoded within the test. Consider using an external configuration file or environment variables for such sensitive and changeable data.

2. **Page Object Model (POM)**: Organize the code using the Page Object Model design pattern. This involves creating separate classes for each page that will contain the methods to interact with the page. It makes the code more maintainable and readable.

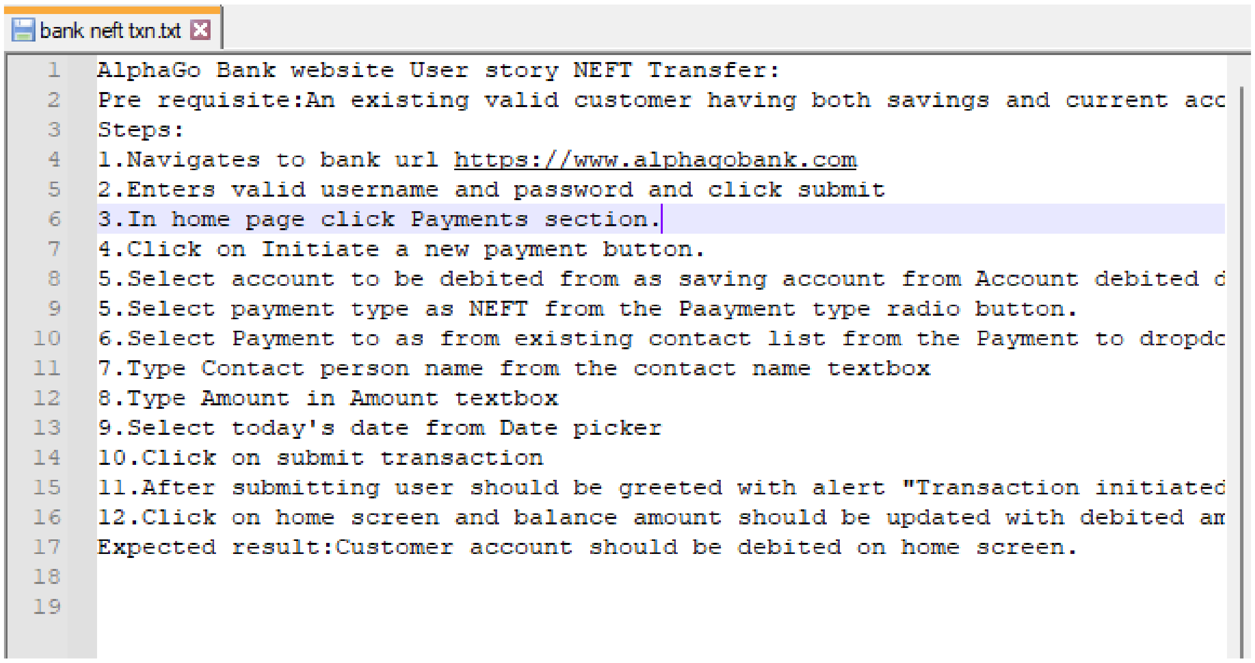
3. **Assertions**: The test ends with an assertion comparing the expected URL to the actual URL. While this is valid, also consider asserting the presence of elements or messages that indicate a successful login for a more robust test.

4.11 Generate Bug Report

Prerequisite: Upload Manual Test case in iQEAssists.

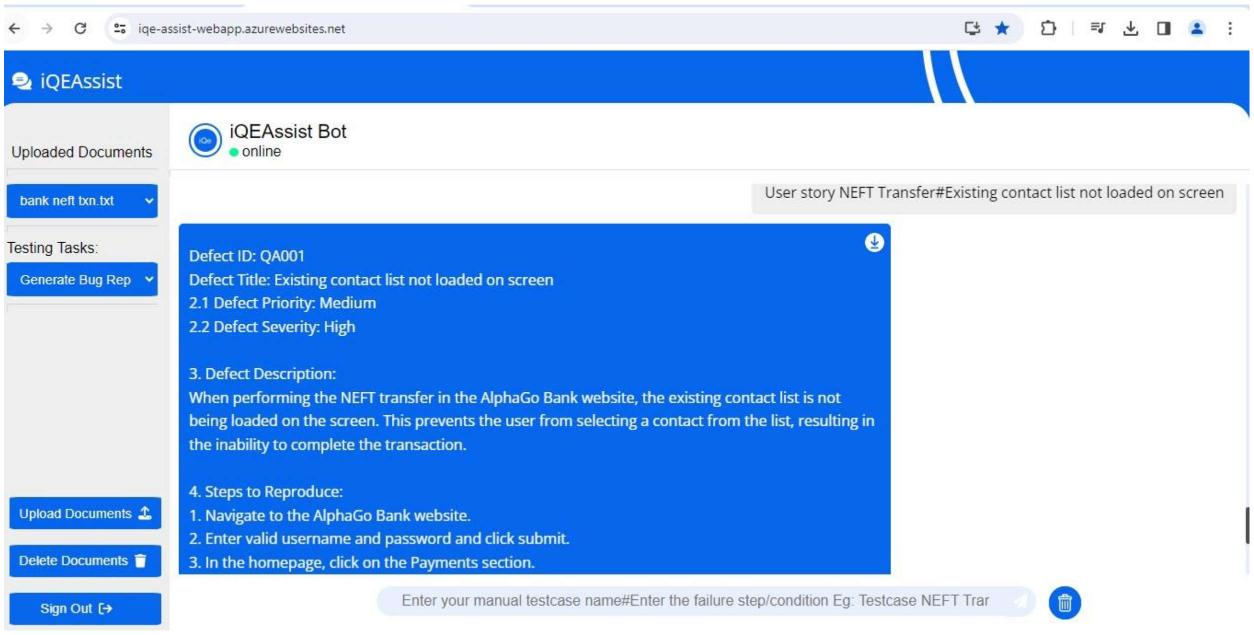
Steps: Select your Generate Bug Report from dropdown and enter the manual test name in textbox and then separated by a # enter the test condition/step where it fails and hit enter.

Example: User story NEFT Transfer#Existing contact list not loaded on screen iQEAssists Output: Generates detailed Bug report with defect id, descriptions, steps toreproduce, expected result and actual result.



bank neft txn.txt

```
AlphaGo Bank website User story NEFT Transfer:
Pre requisite:An existing valid customer having both savings and current acc
Steps:
1.Navigates to bank url https://www.alphagobank.com
2.Enters valid username and password and click submit
3.In home page click Payments section.
4.Click on Initiate a new payment button.
5.Select account to be debited from as saving account from Account debited d
5.Select payment type as NEFT from the Paayment type radio button.
6.Select Payment to as from existing contact list from the Payment to dropdc
7.Type Contact person name from the contact name textbox
8.Type Amount in Amount textbox
9.Select today's date from Date picker
10.Click on submit transaction
11.After submitting user should be greeted with alert "Transaction initiated
12.Click on home screen and balance amount should be updated with debited am
Expected result:Customer account should be debited on home screen.
```



iqe-assist-webapp.azurewebsites.net

iQEAssist

Uploaded Documents: bank neft txn.txt

Testing Tasks: Generate Bug Rep

iQEAssist Bot
online

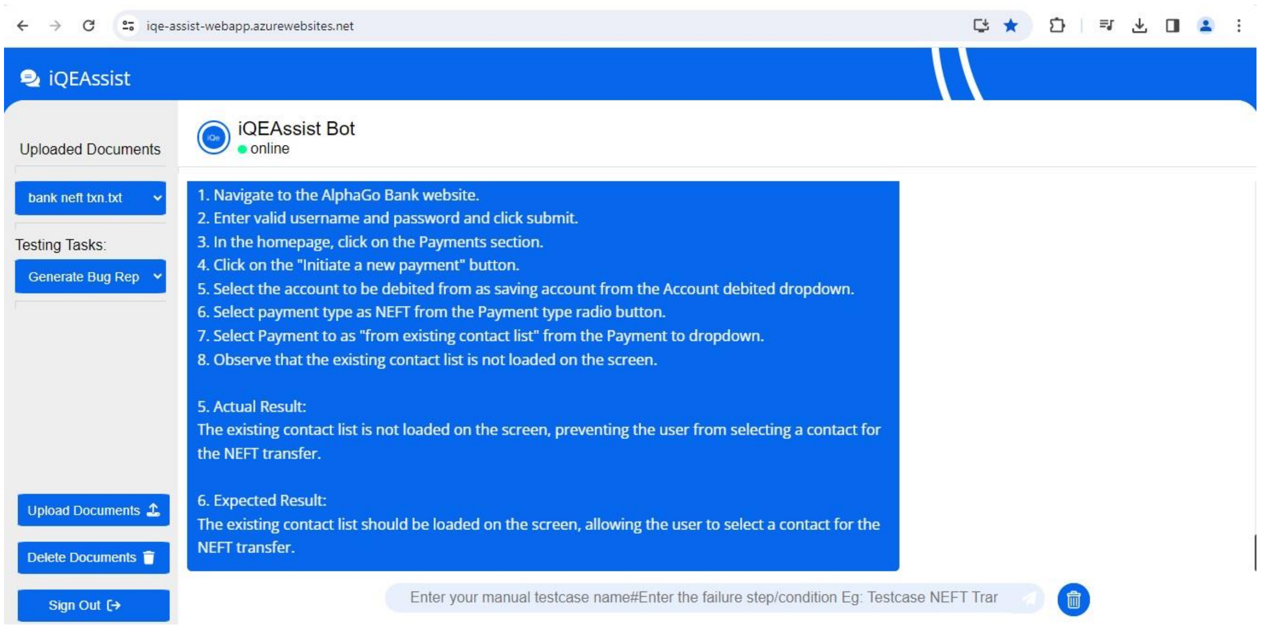
User story NEFT Transfer#Existing contact list not loaded on screen

Defect ID: QA001
Defect Title: Existing contact list not loaded on screen
2.1 Defect Priority: Medium
2.2 Defect Severity: High

3. Defect Description:
When performing the NEFT transfer in the AlphaGo Bank website, the existing contact list is not being loaded on the screen. This prevents the user from selecting a contact from the list, resulting in the inability to complete the transaction.

4. Steps to Reproduce:
1. Navigate to the AlphaGo Bank website.
2. Enter valid username and password and click submit.
3. In the homepage, click on the Payments section.

Upload Documents
Delete Documents
Sign Out
Enter your manual testcase name#Enter the failure step/condition Eg: Testcase NEFT Trar



1. Navigate to the AlphaGo Bank website.
2. Enter valid username and password and click submit.
3. In the homepage, click on the Payments section.
4. Click on the "Initiate a new payment" button.
5. Select the account to be debited from as saving account from the Account debited dropdown.
6. Select payment type as NEFT from the Payment type radio button.
7. Select Payment to as "from existing contact list" from the Payment to dropdown.
8. Observe that the existing contact list is not loaded on the screen.

5. Actual Result:
The existing contact list is not loaded on the screen, preventing the user from selecting a contact for the NEFT transfer.

6. Expected Result:
The existing contact list should be loaded on the screen, allowing the user to select a contact for the NEFT transfer.

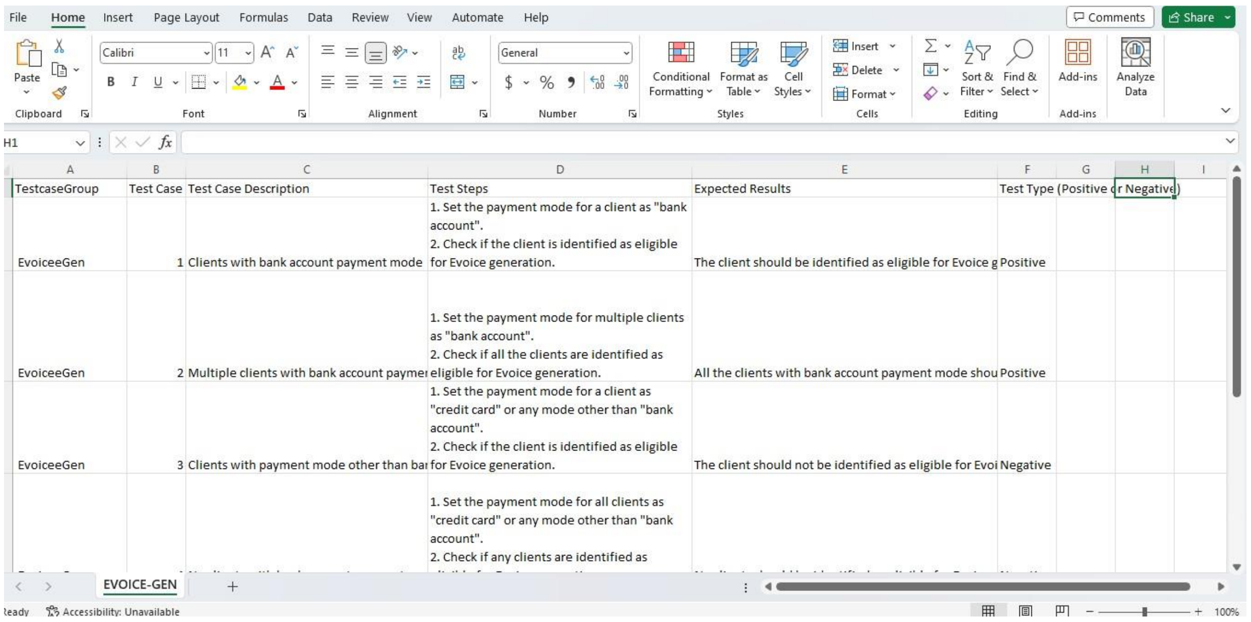
4.12 Test Coverage Analysis

Prerequisite: Upload Manual Test cases with a common Test case Group name isuploaded in iQEAssists. Also, the user story against which review is to be done.

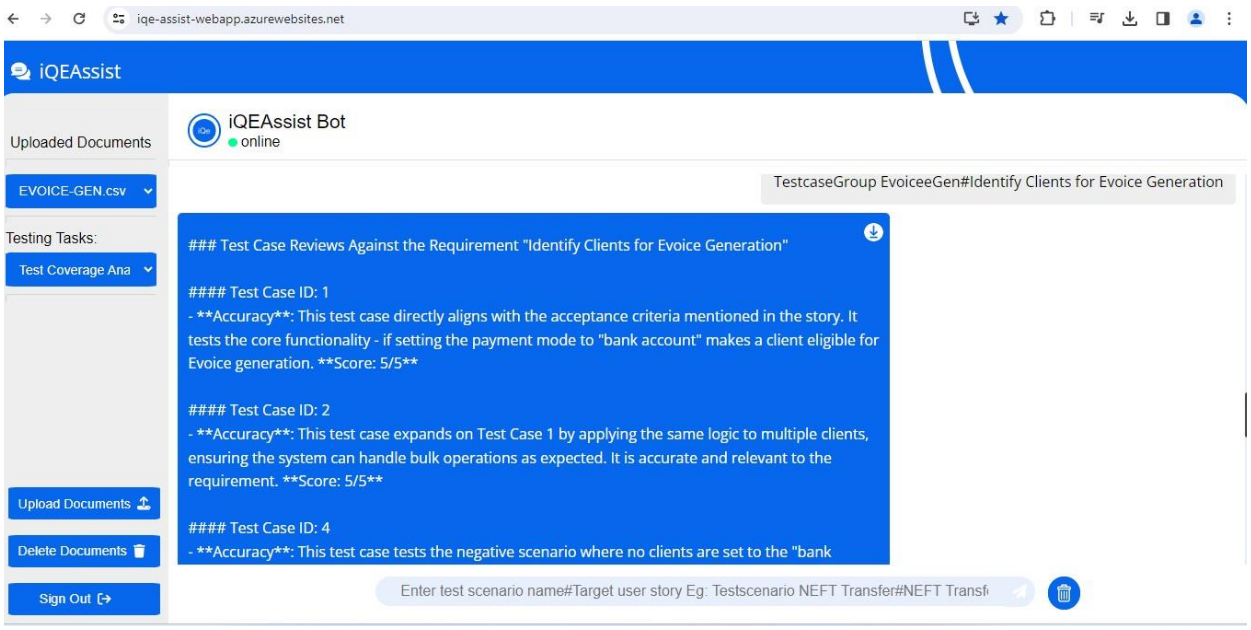
Steps: Select your Test Coverage Analysis from dropdown and enter the manual test group name and then separated by a # enter the user story. /requirement against which it should be compared.

Example: Testcase Group InvoiceGen # Identify Clients for Invoice Generation

iQEAssists Output: Generates coverage report for given requirement like a Test Lead.



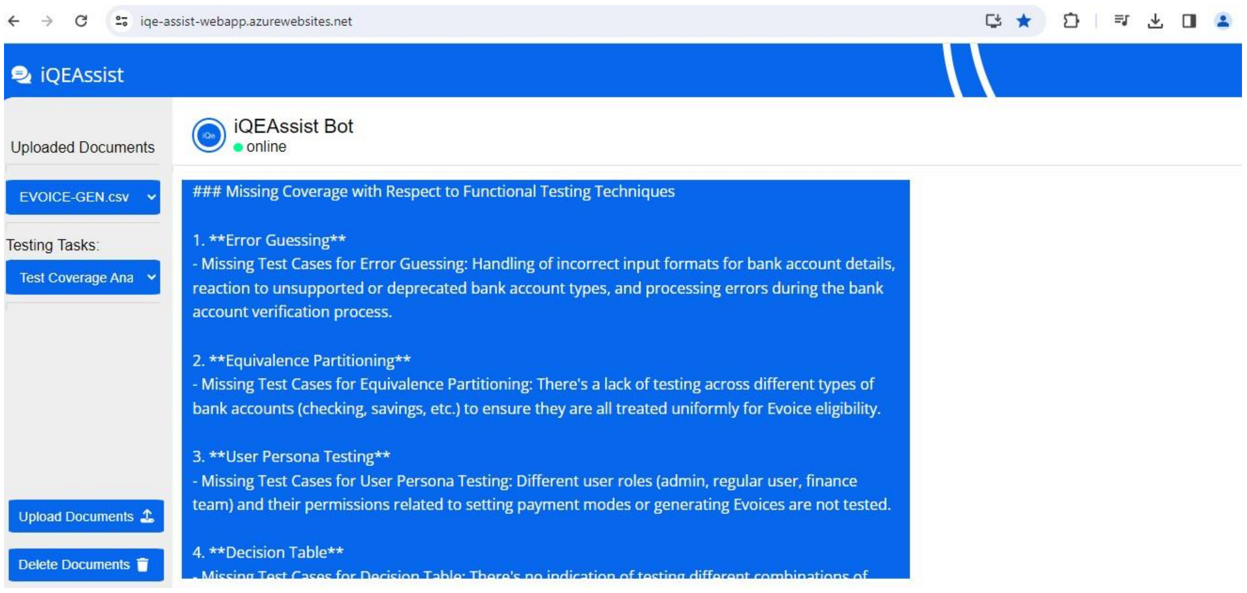
TestcaseGroup	Test Case	Test Case Description	Test Steps	Expected Results	Test Type (Positive or Negative)
EvoiceeGen	1	Clients with bank account payment mode	1. Set the payment mode for a client as "bank account". 2. Check if the client is identified as eligible for Evoice generation.	The client should be identified as eligible for Evoice g	Positive
EvoiceeGen	2	Multiple clients with bank account paymer	1. Set the payment mode for multiple clients as "bank account". 2. Check if all the clients are identified as eligible for Evoice generation.	All the clients with bank account payment mode shou	Positive
EvoiceeGen	3	Clients with payment mode other than bar	1. Set the payment mode for a client as "credit card" or any mode other than "bank account". 2. Check if the client is identified as eligible for Evoice generation.	The client should not be identified as eligible for Evoi	Negative
			1. Set the payment mode for all clients as "credit card" or any mode other than "bank account". 2. Check if any clients are identified as		



TestcaseGroup EvoiceeGen#Identify Clients for Evoice Generation

```
### Test Case Reviews Against the Requirement "Identify Clients for Evoice Generation"

#### Test Case ID: 1
- **Accuracy**: This test case directly aligns with the acceptance criteria mentioned in the story. It tests the core functionality - if setting the payment mode to "bank account" makes a client eligible for Evoice generation. **Score: 5/5**

#### Test Case ID: 2
- **Accuracy**: This test case expands on Test Case 1 by applying the same logic to multiple clients, ensuring the system can handle bulk operations as expected. It is accurate and relevant to the requirement. **Score: 5/5**

#### Test Case ID: 4
- **Accuracy**: This test case tests the negative scenario where no clients are set to the "bank
```

Enter test scenario name#Target user story Eg: Testscenario NEFT Transfer#NEFT Transf



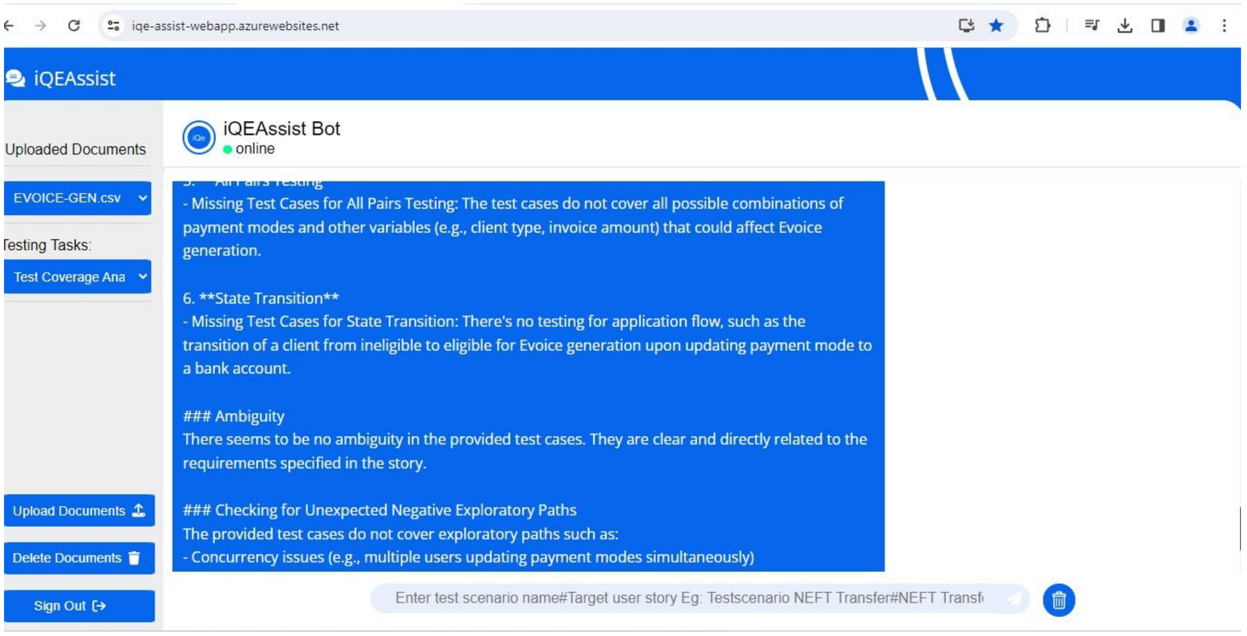
Missing Coverage with Respect to Functional Testing Techniques

1. **Error Guessing**
- Missing Test Cases for Error Guessing: Handling of incorrect input formats for bank account details, reaction to unsupported or deprecated bank account types, and processing errors during the bank account verification process.

2. **Equivalence Partitioning**
- Missing Test Cases for Equivalence Partitioning: There's a lack of testing across different types of bank accounts (checking, savings, etc.) to ensure they are all treated uniformly for Evoice eligibility.

3. **User Persona Testing**
- Missing Test Cases for User Persona Testing: Different user roles (admin, regular user, finance team) and their permissions related to setting payment modes or generating Evoices are not tested.

4. **Decision Table**



- Missing Test Cases for All Pairs Testing: The test cases do not cover all possible combinations of payment modes and other variables (e.g., client type, invoice amount) that could affect Evoice generation.

6. **State Transition**
- Missing Test Cases for State Transition: There's no testing for application flow, such as the transition of a client from ineligible to eligible for Evoice generation upon updating payment mode to a bank account.

Ambiguity
There seems to be no ambiguity in the provided test cases. They are clear and directly related to the requirements specified in the story.

Checking for Unexpected Negative Exploratory Paths
The provided test cases do not cover exploratory paths such as:
- Concurrency issues (e.g., multiple users updating payment modes simultaneously)

4.13 Code Conversion/Review Agent

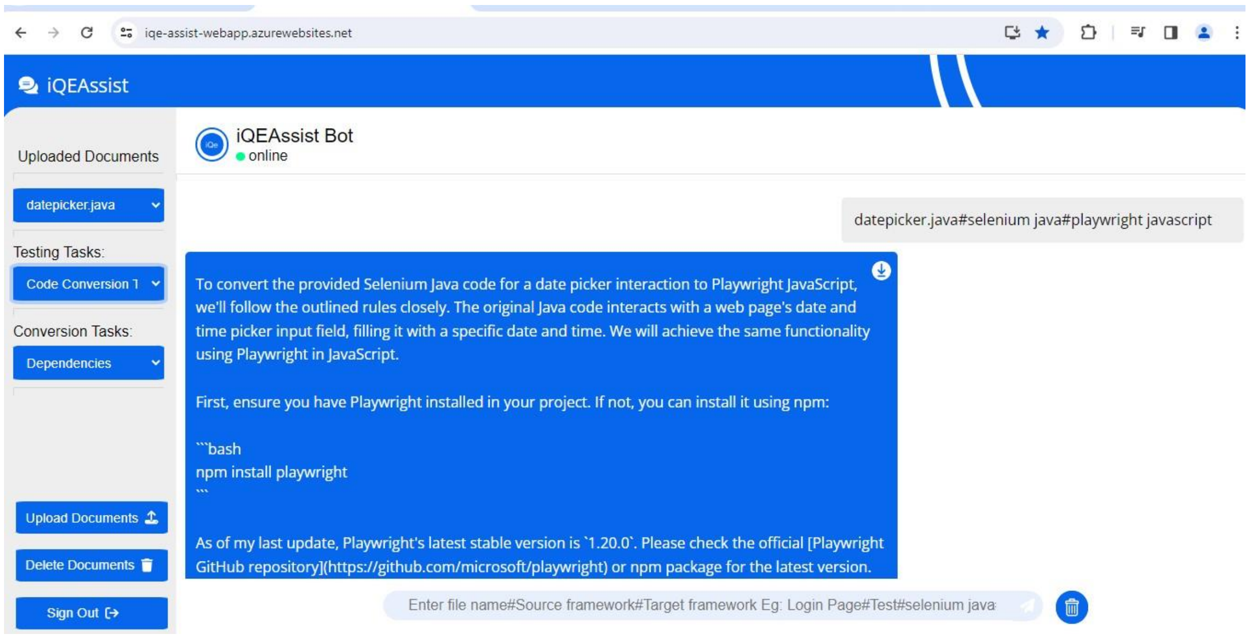
This task converts source framework and language code to target framework language and code. You can upload files formats of major four languages (java, python, JavaScript, C#) and dataformats properties, Json and xml. You can also connect your Gitlab data by providing credentialsand it will be available in uploaded files.

Prerequisite: Your code file to be converted should be present in iQEAssists

Steps: Select your **Code Conversion** from dropdown and enter your code file name and then separated by a # enter source framework and language followed by # and target framework and language

Example:

iQEAssists Output: iQEAssists generates target framework code with comments and explanations



datepicker.java#selenium java#playwright javascript

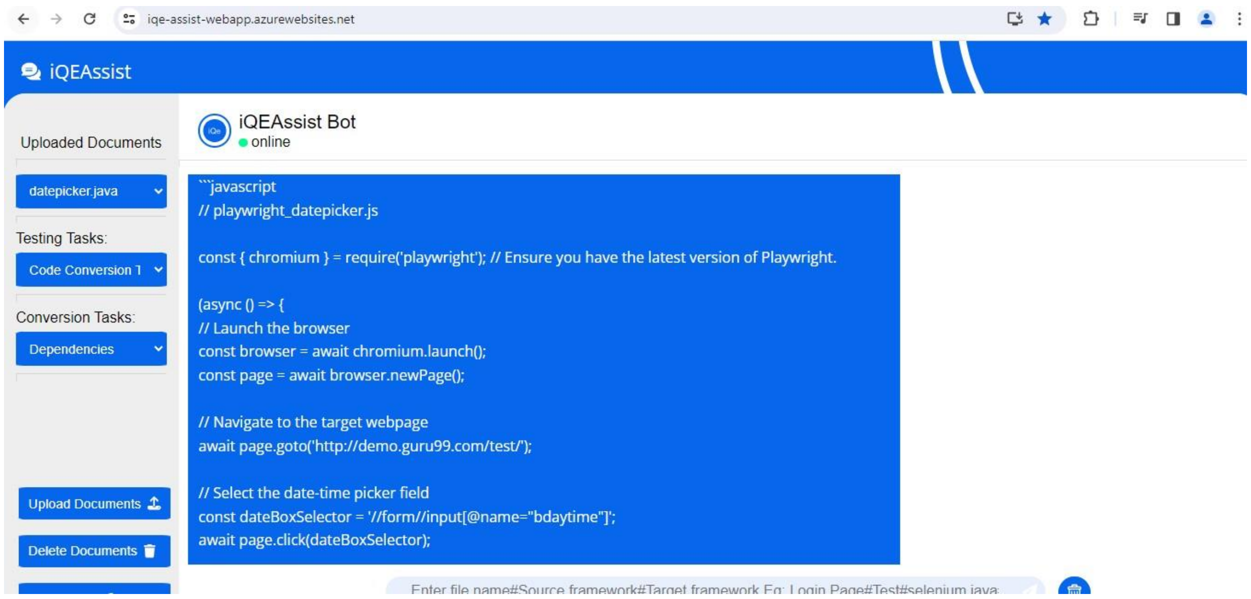
To convert the provided Selenium Java code for a date picker interaction to Playwright JavaScript, we'll follow the outlined rules closely. The original Java code interacts with a web page's date and time picker input field, filling it with a specific date and time. We will achieve the same functionality using Playwright in JavaScript.

First, ensure you have Playwright installed in your project. If not, you can install it using npm:

```bash
npm install playwright
```

As of my last update, Playwright's latest stable version is `1.20.0`. Please check the official [Playwright GitHub repository](https://github.com/microsoft/playwright) or npm package for the latest version.

Enter file name#Source framework#Target framework Eg: Login Page#Test#selenium java



```javascript
// playwright_datepicker.js

const { chromium } = require('playwright'); // Ensure you have the latest version of Playwright.

(async () => {
// Launch the browser
const browser = await chromium.launch();
const page = await browser.newPage();

// Navigate to the target webpage
await page.goto('http://demo.guru99.com/test/');

// Select the date-time picker field
const dateBoxSelector = '//form//input[@name="bdaytime"]';
await page.click(dateBoxSelector);
```

Enter file name#Source framework#Target framework Eg: Login Page#Test#selenium java

Full output downloaded from

To convert the provided Selenium Java code for a date picker interaction to Playwright JavaScript, we'll follow the outlined rules closely. The original Java code interacts with a web page's date and time picker input field, filling it with a specific date and time. We will achieve the same functionality using Playwright in JavaScript.

First, ensure you have Playwright installed in your project. If not, you can install it using npm:

```bash
npm install playwright
```

As of my last update, Playwright's latest stable version is `1.20.0`. Please check the official [Playwright GitHub repository] (https://github.com/microsoft/playwright) or npm package forthe latest version.

Here's how the conversion looks:

```javascript
// playwright_datepicker.js

const { chromium } = require('playwright'); // Ensure you have the latest version of Playwright.

(async () => {
  // Launch the browser
  const browser = await chromium.launch();
  const page = await browser.newPage();

  // Navigate to the target webpage
  await page.goto('http://demo.guru99.com/test/');
  // Select the date-time picker field
  const dateBoxSelector = '//form//input[@name="bdaytime"]';
  await page.click(dateBoxSelector);

  // Fill the date as mm/dd/yyyy - For example, 09/25/2013
  // Then, press Tab to shift focus to the time field and fill the time as 02:45 PM
  // Playwright allows us to send both date and time together if the field permits it
  await page.fill(dateBoxSelector, '09252013');
  await page.keyboard.press('Tab');
  await page.keyboard.type('0245PM');

  // Code assumes the existence of the date picker control on the page.
  // Ensure the selectors used match those of your specific date picker control.

  // Close the browser
  await browser.close();
})();
```

Explanation and Best Practices:

1. **Dependencies and Versioning:** We're using Playwright's `chromium` browser instance. Ensure you have the latest version of Playwright installed for new features and security updates.

2. **Functionality Equivalence:** The converted code accomplishes the same task as the original Selenium Java code—interacting with a date-time picker on a webpage. However, Playwright's approach simplifies the process by allowing more direct interactions with the

webpage, such as filling in text fields and simulating keyboard presses without needing to manage drivers explicitly.

3. **Optimization and Best Practices:** The code is optimized by using Playwright's promises and async/await syntax for better readability and flow control. Playwright automatically waits for elements to be ready before interacting with them, which reduces the need for explicit wait statements.

4. **Exception Handling:** Playwright operations that fail (e.g., due to missing elements) will throw errors. It's best practice to wrap Playwright operations in try/catch blocks if you need custom error handling or logging.

5. **External References:** The code assumes that the date picker exists and that its input can accept date and time as typed text. If the date picker implementation on the target page requires different interactions (e.g., selecting dates from a calendar widget), additional steps would be needed.

This JavaScript code using Playwright is a direct translation of the provided Selenium Java code, adhering to the specified conversion rules and focusing on maintaining functional equivalence, optimization, and adherence to Playwright best practices.