AI Use case Test Case Generator

User Manual

Table of Contents

1. 🧩 Introduction

The **AI Test Case Generator** is a powerful tool designed to automate and streamline manual test case creation using Large Language Models (LLMs). It transforms natural language inputs—such as user stories, functional requirements, and acceptance criteria—into structured, human-readable test cases.

Key Benefits:

- Supports positive, negative, edge, and boundary scenarios
- Enhance test coverage and consistency
- Reduces manual QA effort
- Accelerates testing cycles across web, mobile, and enterprise platforms

2 Context File Setup with examples:

The context file is a foundational input that equips the AI-powered test case generator with deep, structured knowledge about your application. By detailing the UI screens, features, user roles, business logic, and edge cases, this file enables the Large Language Model (LLM) to interpret requirements with greater accuracy and domain awareness. The result is a set of manual test cases that are not only comprehensive and relevant, but also aligned with real-world usage scenarios, validation paths, and user interactions. This setup ensures that the generated test cases reflect the true intent of your application, reduce manual effort, and improve overall test coverage and quality.

Below are formats which the tool supports: -

2.1 JSON Format

Machine-readable, widely supported format. Below provided is the sample which can be used for designing a context setup file specific to the project

UI JSON:

```
{
  "appName": "SmartBank",
  "overview": "Mobile banking app for managing accounts, transfers, and bill payments.",
  "uiScreens": [
    {
      "screenName": "Login Screen",
      "description": "User login via credentials or biometrics.",
       "keyElements": ["Username field", "Password field", "Login button"],
      "navigation": "First screen; leads to Dashboard on success."
    }
  ],
  "functionality": [
    {
      "featureName": "Fund Transfer",
      "description": "Transfer money between accounts.",
      "relatedScreens": ["Dashboard", "Transfer Screen"],
      "businessRules": ["OTP required for transfers > $10,000"]
    }
  ],
  "userRoles": [
    {
```

```
      "roleName": "Standard User",
      "permissions": ["View accounts", "Transfer funds"],
      "featureAccess": ["Login Screen", "Dashboard", "Transfer"]
    }
  ],
  "edgeCasesAndConstraints": [
    "Login fails after 5 incorrect attempts"
  ],
  "testDataGuidelines": [
    "Use usernames like user001",
    "Test transfer amounts: $0, $10,000, $10,001"
  ],
  "additionalNotes": "Biometric login supported only on compatible
devices."
}
```

For API

```
{
  "moduleName": "UserAuth API",
  "type": "api",
  "use_gherkin": false,
  "description": "API for handling user authentication using POST
method",
  "api": {
    "endpoint": "/api/auth/login",
    "methods": ["POST"],
    "requestParams": ["username", "password"],
    "responseKeys": ["token", "user_id", "expires_in"],
     "validations": ["status_code == 200", "response contains
token"]
  }
}
```

2.2 YAML Format

Human-readable, indentation-based structure. Below provided is the sample which can be used for designing a context setup file specific to the project

YAML:

```yaml
appName: SmartBank
overview: Mobile banking app for managing accounts, transfers, and
bill payments.
uiScreens:
  - screenName: Login Screen
    description: User login via credentials or biometrics.
    keyElements: [Username field, Password field, Login button]
    navigation: First screen; leads to Dashboard on success.

functionality:
  - featureName: Fund Transfer
    description: Transfer money between accounts.
    relatedScreens: [Dashboard, Transfer Screen]
    businessRules: [OTP required for transfers > $10,000]

userRoles:
  - roleName: Standard User
    permissions: [View accounts, Transfer funds]
    featureAccess: [Login Screen, Dashboard, Transfer]

edgeCasesAndConstraints:
  - Login fails after 5 incorrect attempts

testDataGuidelines:
  - Use usernames like user001
  - Test transfer amounts: $0, $10,000, $10,001

additionalNotes: Biometric login supported only on compatible
devices.
```

2.3 Plain text Format

Document format for teams preferring word processors. Below provided is the sample which can be used for designing a context setup file specific to the project

- **App Name**: SmartBank
- **Overview**: Mobile banking app for managing accounts, transfers, and bill payments
- **UI Screens**:
    - Login Screen
        - Description: User login via credentials or biometrics
        - Key Elements: Username field, Password field, Login button
        - Navigation: Leads to Dashboard on success
- **Functionality**:
    - Fund Transfer
        - Description: Transfer money between accounts
        - Related Screens: Dashboard, Transfer Screen
        - Business Rules: OTP required for transfers > $10,000
- **User Roles**:
    - Standard User
        - Permissions: View accounts, Transfer funds
        - Feature Access: Login Screen, Dashboard, Transfer
- **Edge Cases & Constraints**: Login fails after 5 incorrect attempts
- **Test Data Guidelines**: Use usernames like user001; Test transfer amounts: $0, $10,000, $10,001
- **Additional Notes**: Biometric login supported only on compatible devices

2.4 🌐 Scrape URL

The **Scrape URL** feature allows users to automatically extract relevant information from public web pages—such as product documentation, feature descriptions, UI walkthroughs, or release notes—and convert it into structured context for generating manual test cases. This helps the AI model understand the application's functionality, UI flow, and business logic without requiring manual input for every detail.

When a user provides a valid URL, the tool:

- **Fetches the page content** (text only, no scripts or dynamic interactions)
- **Identifies key sections** such as screen names, features, user flows, and validation rules
- **Structures the extracted data** into a context file format (YAML or JSON)
- **Feeds the context** to the LLM to generate accurate, domain-aware test cases

2.5 📂 Upload Existing Test Cases

The user can upload existing test cases based on the functionality/requirement being tested which will serve as information and context for the LLM model to generate

3. 🛠 JIRA Setup

For this part, the requirement is to have a **user-specific JIRA API Key** (which can be extracted from the JIRA application).

The user needs to:

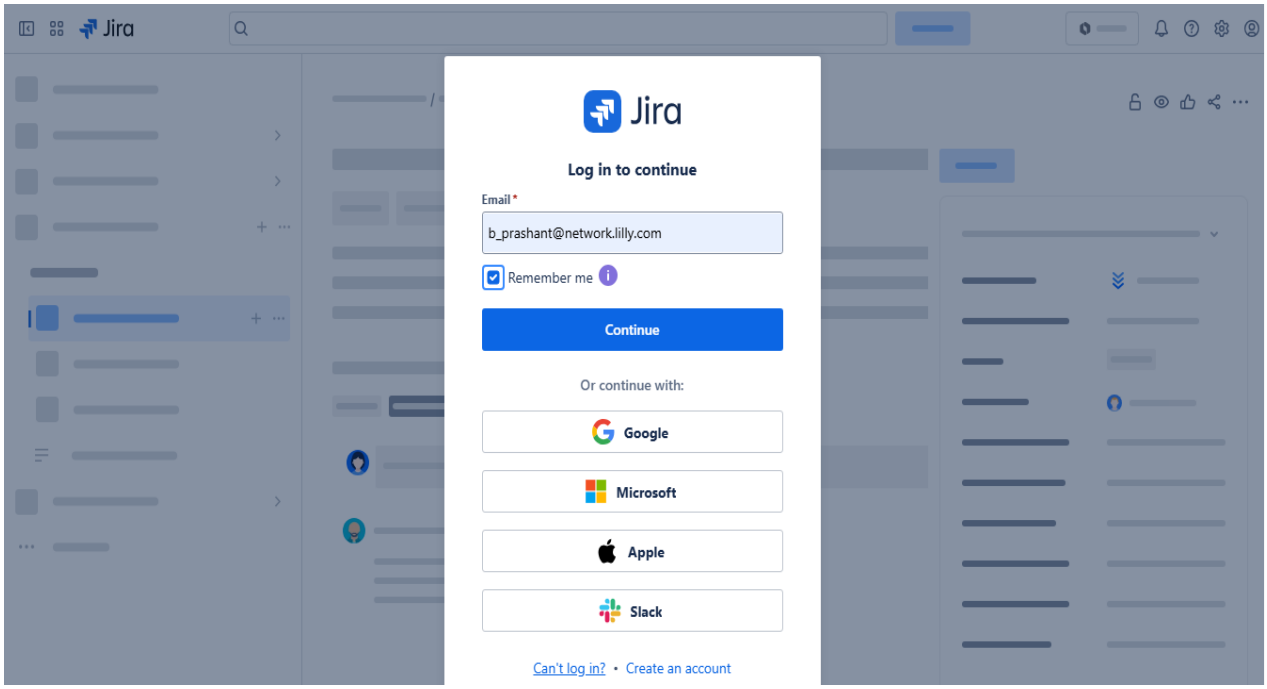
- Populate the **JIRA API Key** under the **JIRA Setup** section on the tool interface.
- Populate the **JQL query** specific to the project under use.

🔹 **Sample JQL Queries**

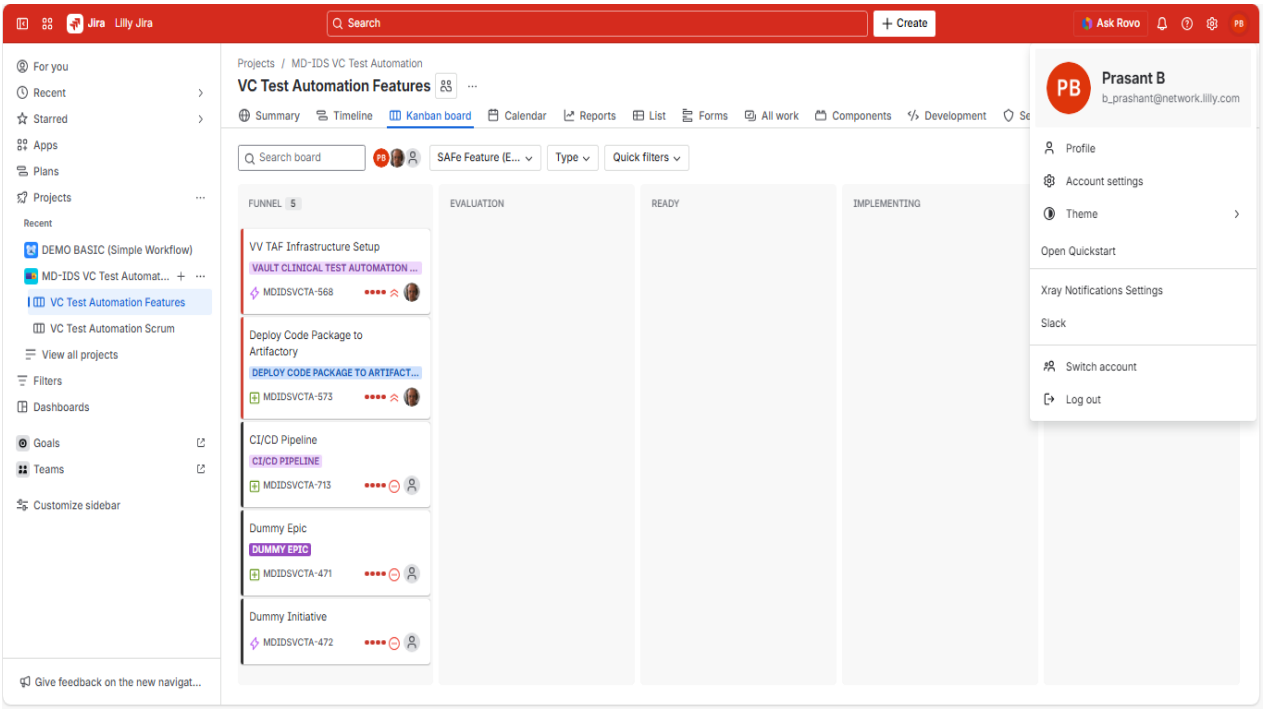
- TESTAUTO AND issuetype=Story
- project=TESTAUTO AND issuetype=Story AND status = Approved ORDER BY create

🔹 **Steps to Extract the JIRA API Token along with screenshots for reference**

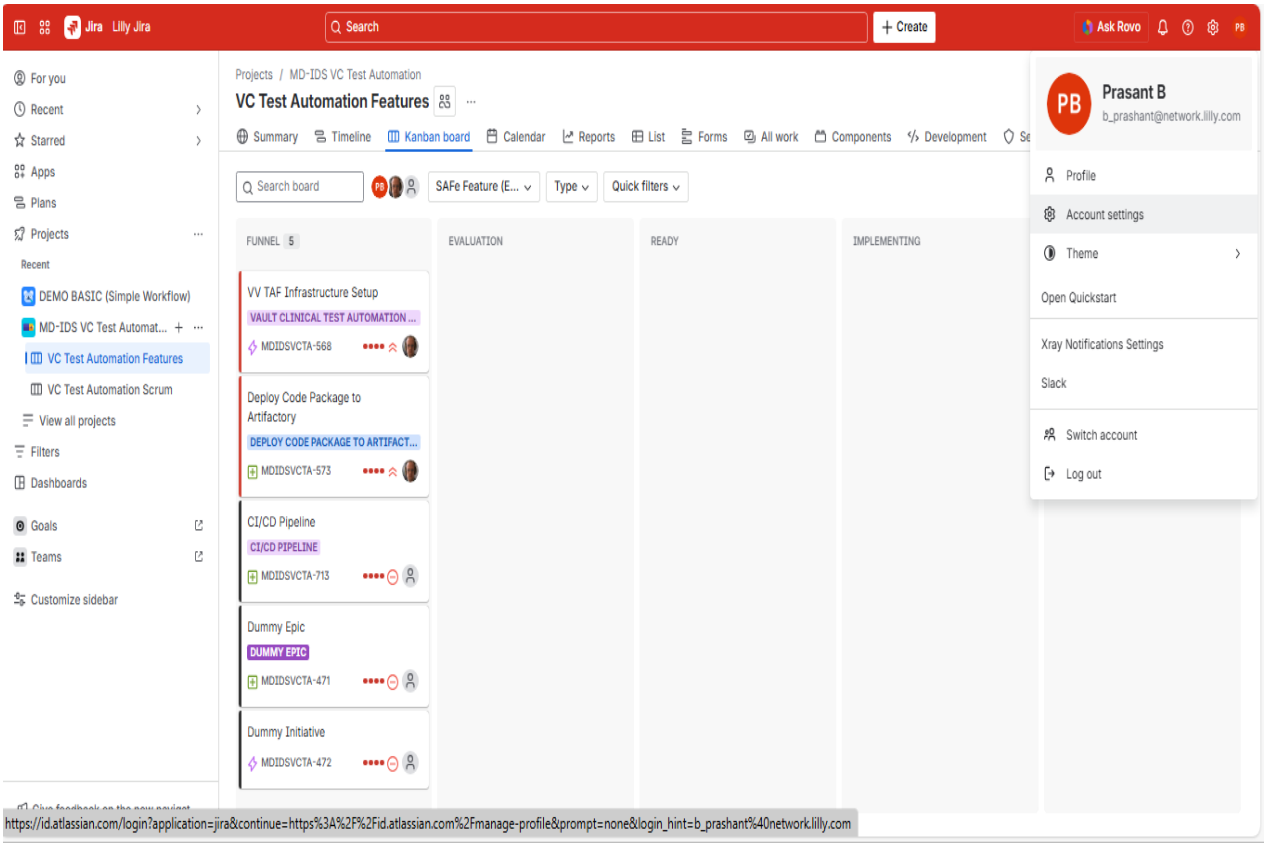
1. **Login to JIRA**



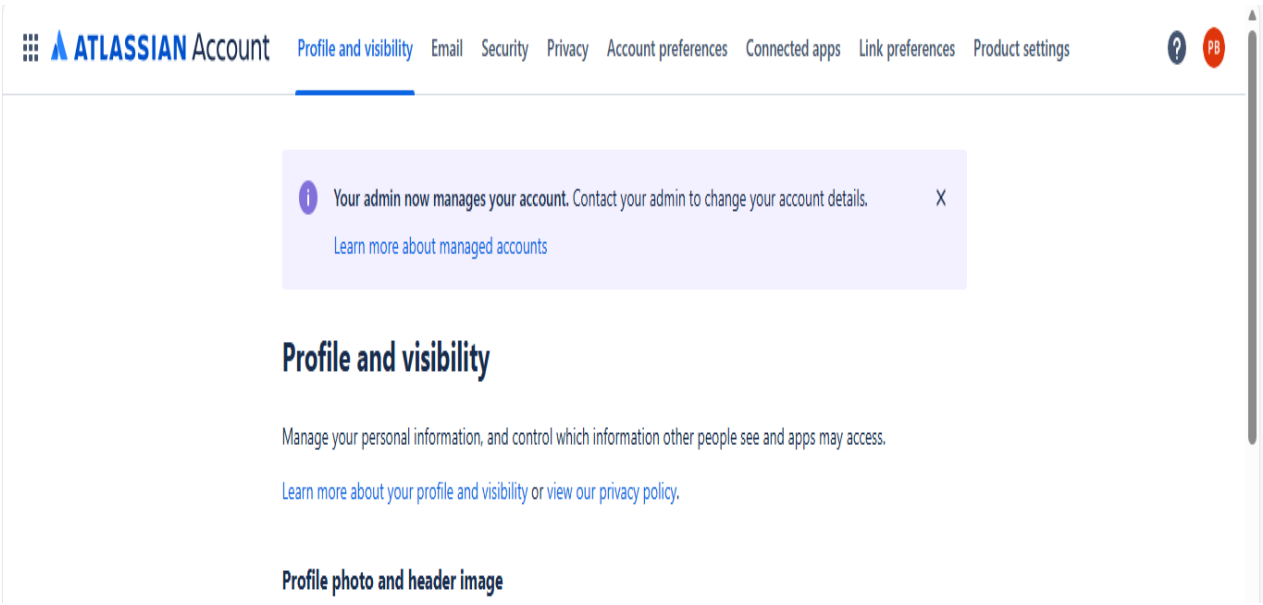
2. Click on your account (top right-hand side of the screen)



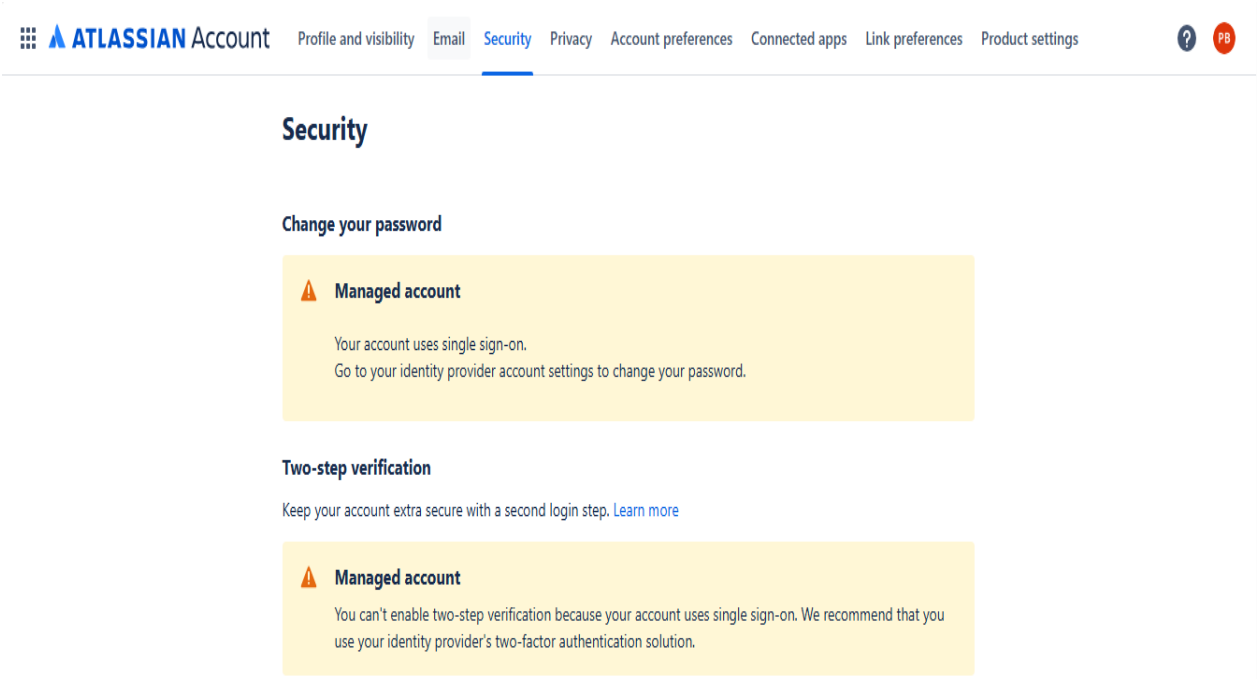
3. Click on **Account Settings**



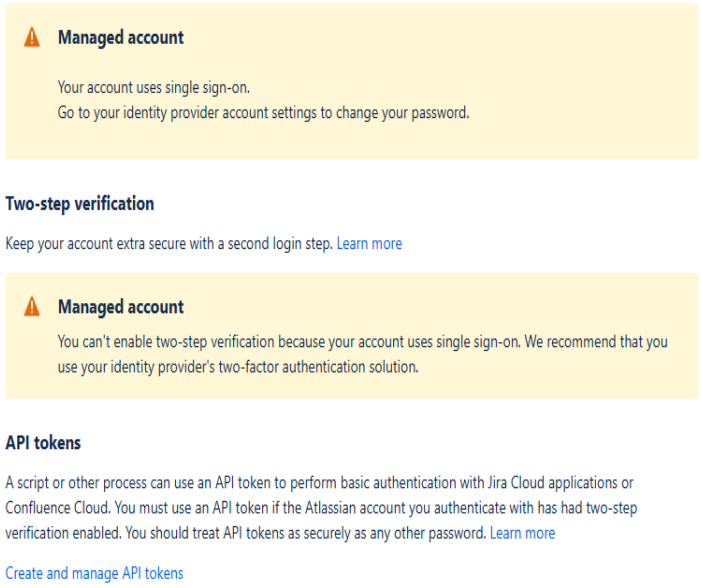
4. You will land on the **Atlassian Account** page



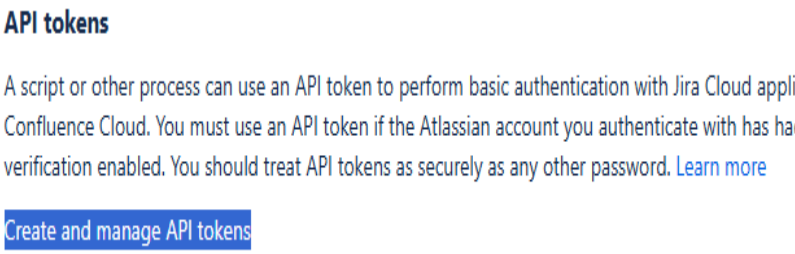
5. Click on the **Security** tab



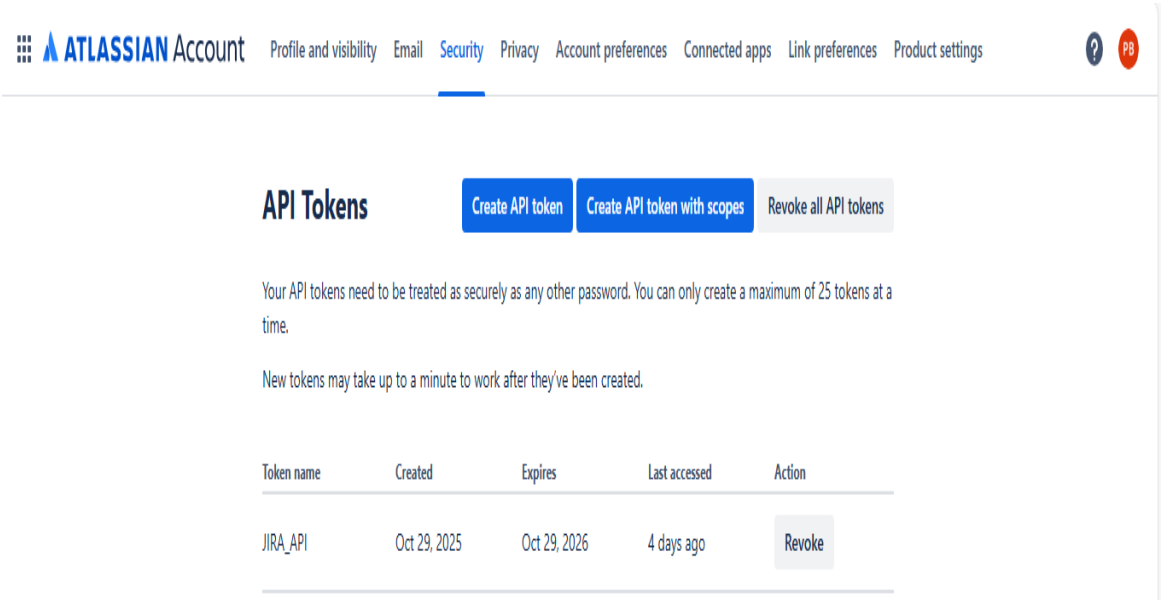
6. On the Security page, scroll down to the **API Tokens** section



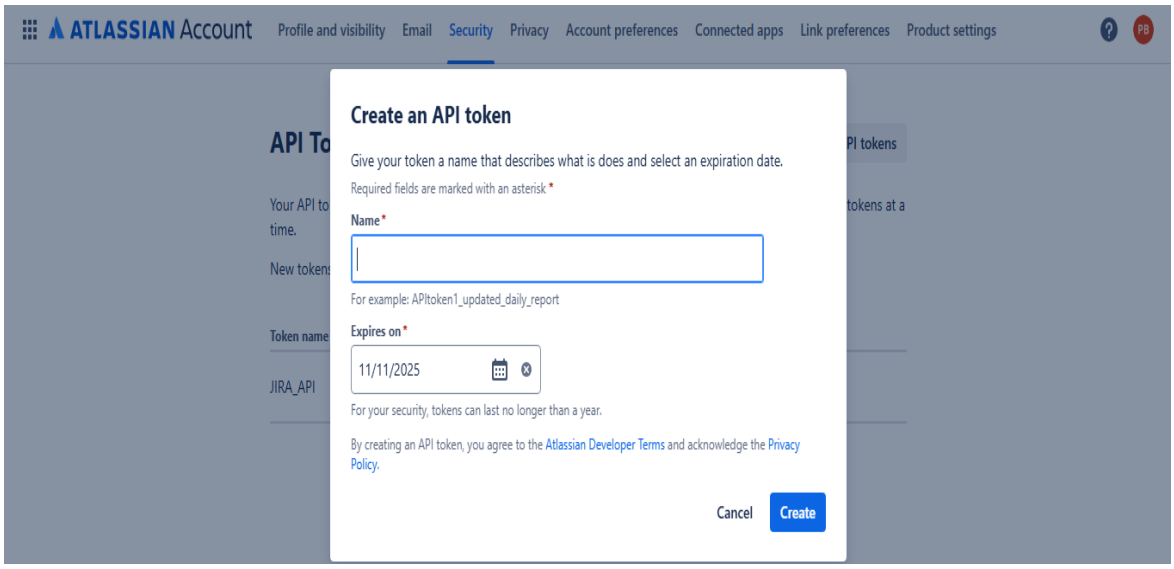
7. Click on the hyperlink **Create and manage API tokens**



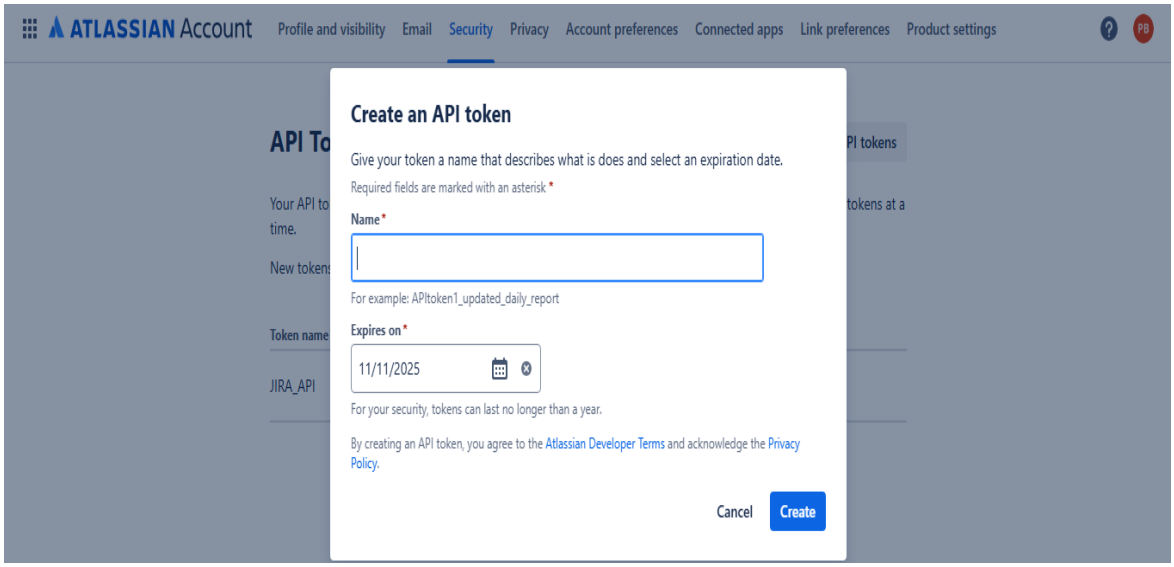
8. A new **API Tokens** page will open



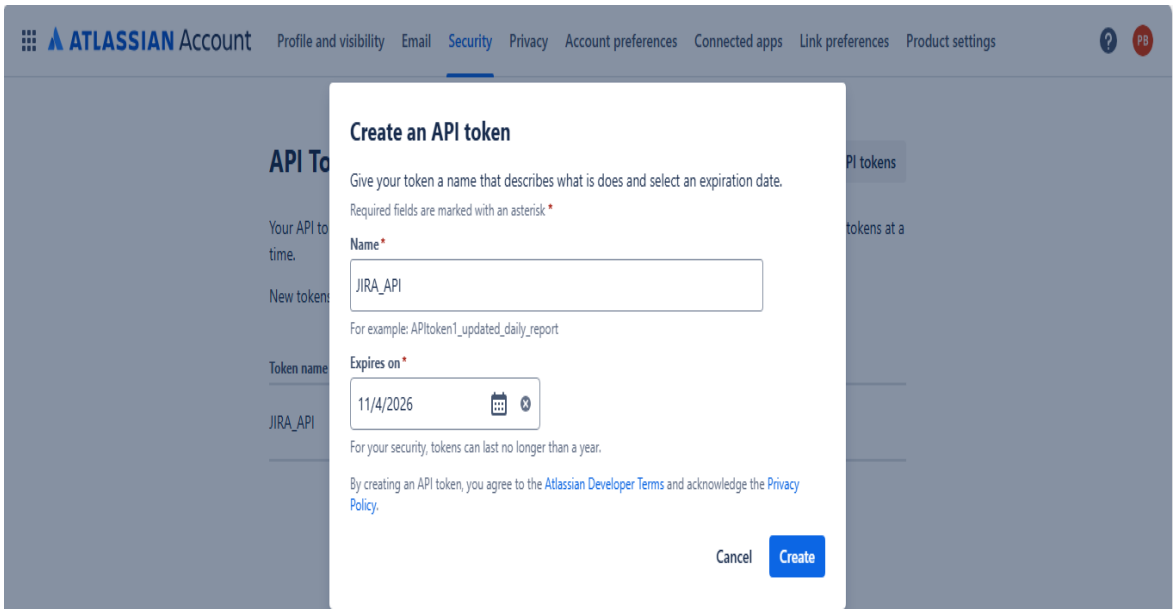
9. Click on the **Create API Token** button



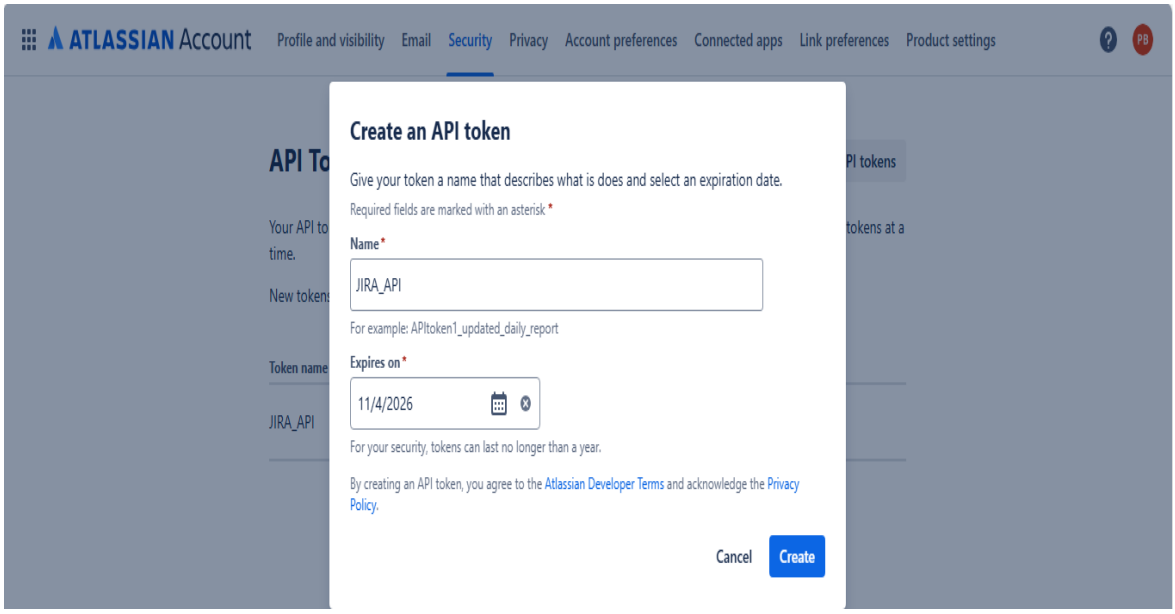
10. A dialog box named **Create an API Token** will appear



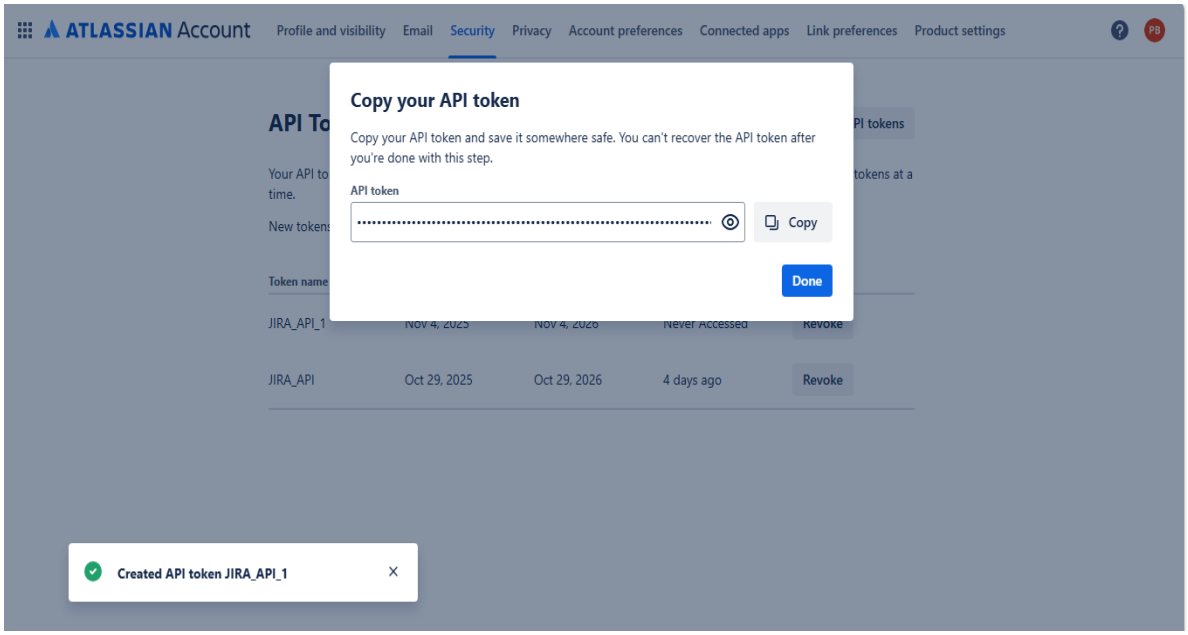
11. Enter the **name of the token** and set the **expiry date** (maximum up to 1 year)



12. Click on **Create**



13. The token will be created — **save the token ID** by clicking on Copy icon.



14. Later, use this token ID to populate the **JIRA API Key** field under the **JIRA Setup** section on the tool UI

Token name	Created	Expires	Last accessed	Action
JIRA_API_1	Nov 4, 2025	Nov 4, 2026	Never Accessed	Revoke

4. Process of Generating Test Cases

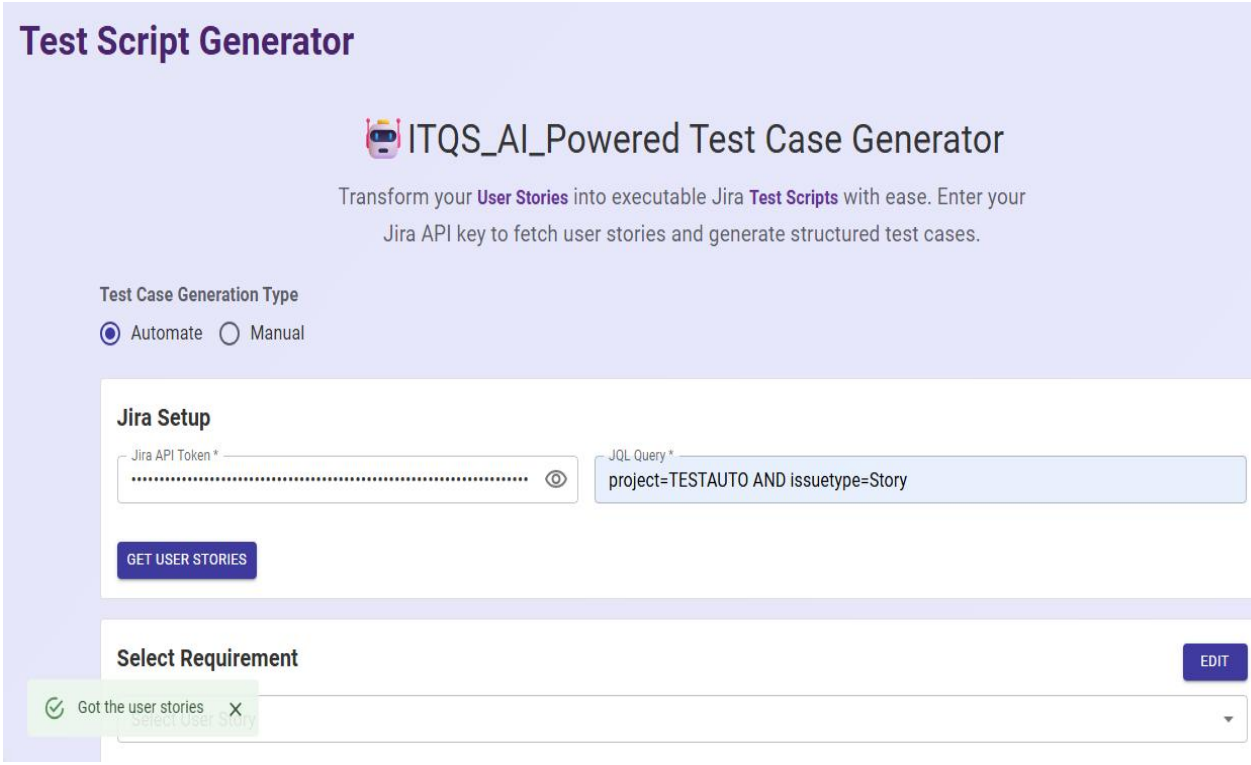
Post the application launch, follow the instructions below to generate the test cases and upload them to the JIRA project workspace:

Step 1

- By default, the radio button selected for Test Case Generation Type field will be Automate. No changes are required here.

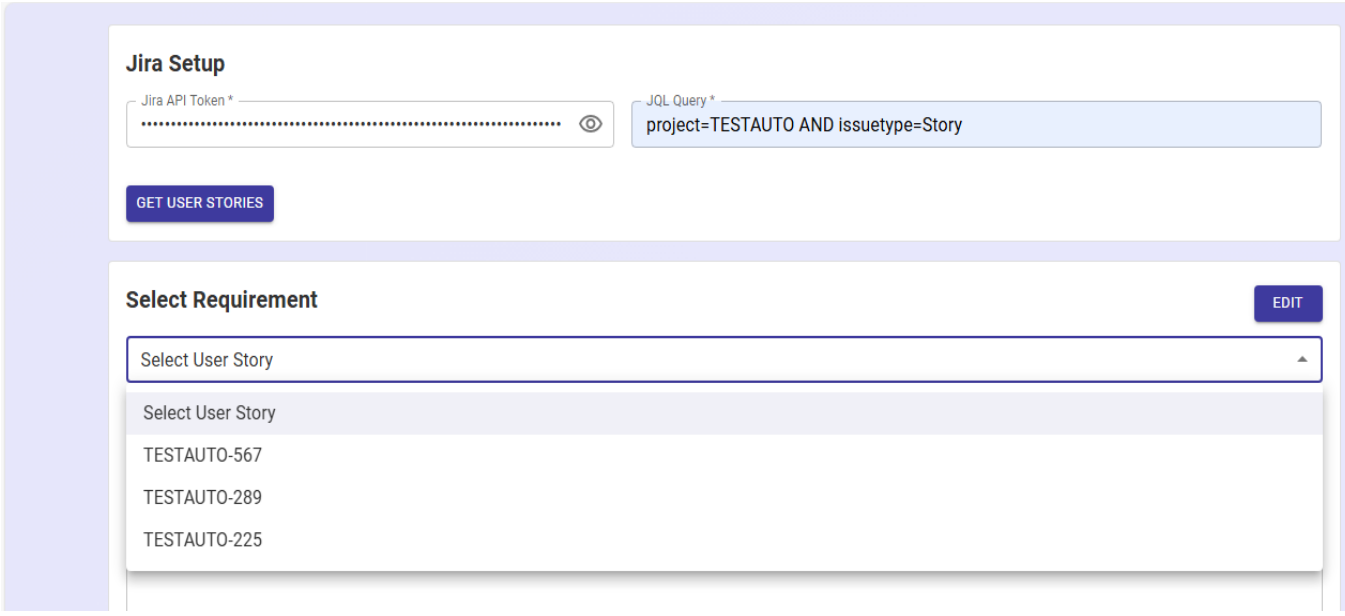
Step 2

- Under the JIRA Setup section:
    - Enter the JIRA API Token generated as stated above (Note -: keep this encrypted).
    - Enter the JQL query.
    - Click on the Get User Stories button.

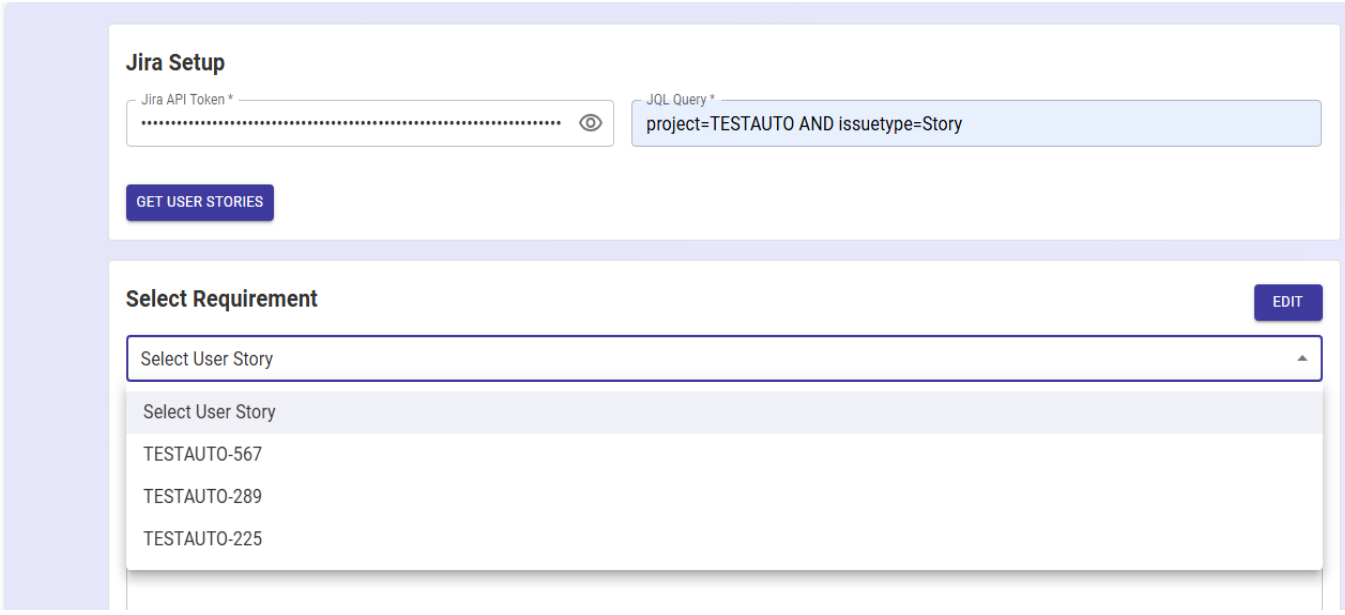


Step 3

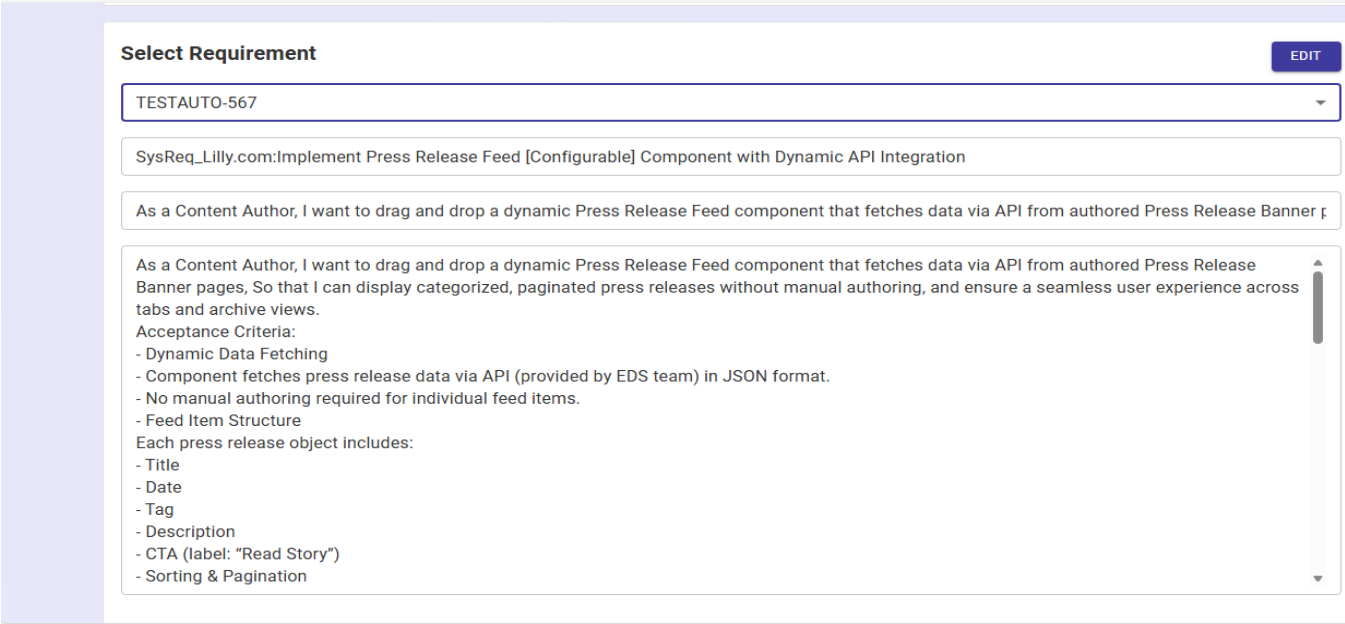
- User stories/requirements are extracted from the JIRA project workspace.



- They will be available under the Select Requirements section in the Select User Story dropdown.
- Click on the dropdown and select the desired User Story.



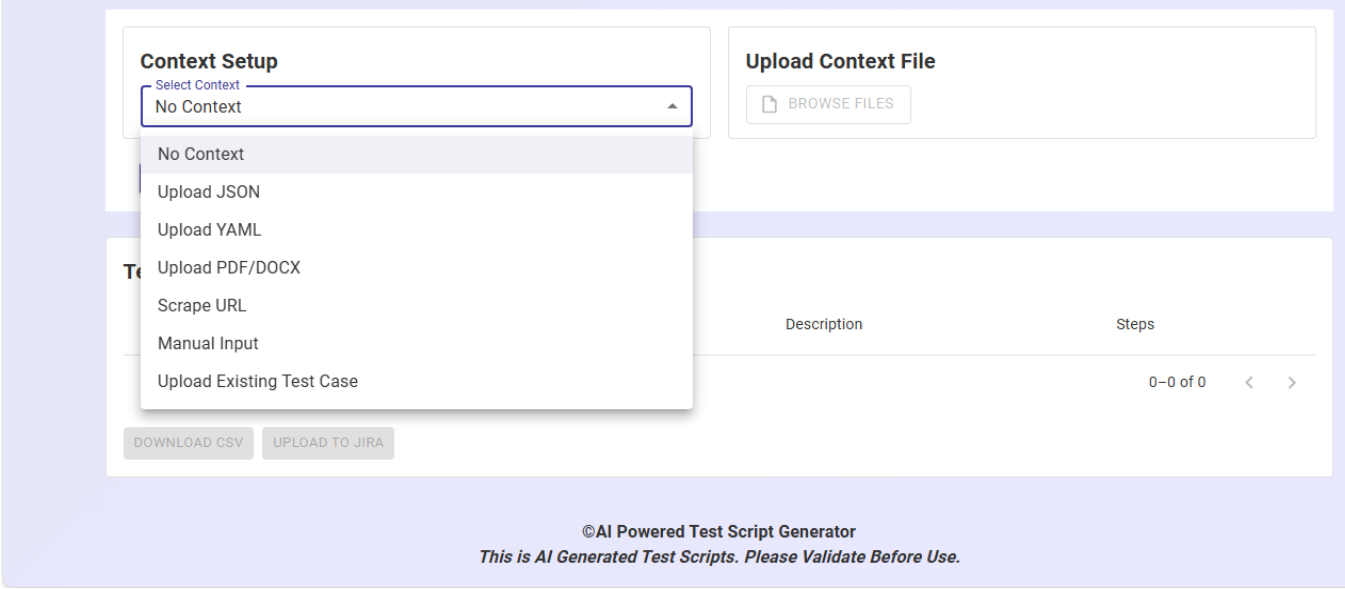
- Once selected, the Summary, Description, and Acceptance Criteria are auto populated based on the information pulled from JIRA using the JQL query.



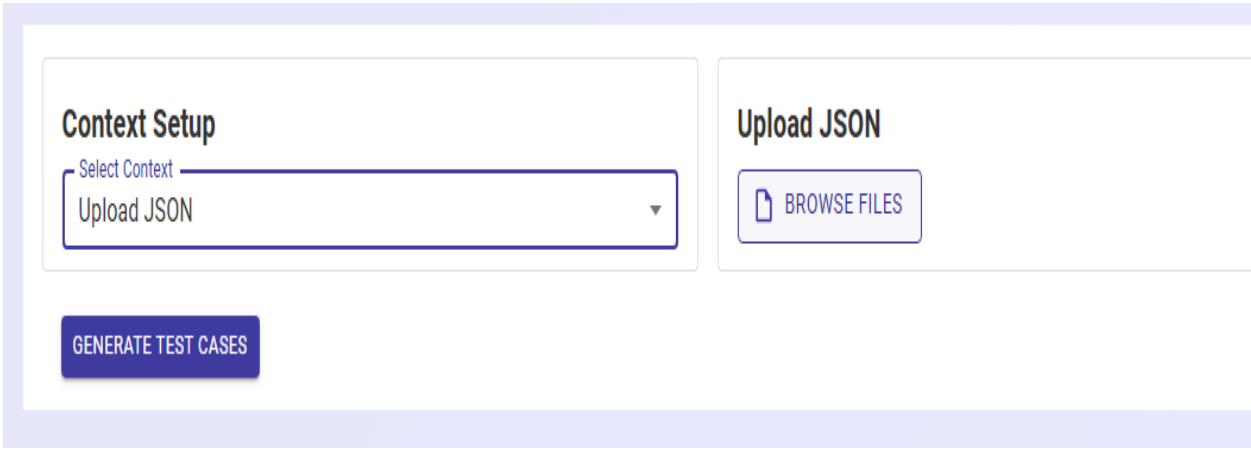
- Edit operation is available (optional) by clicking on the Edit button in the same section.

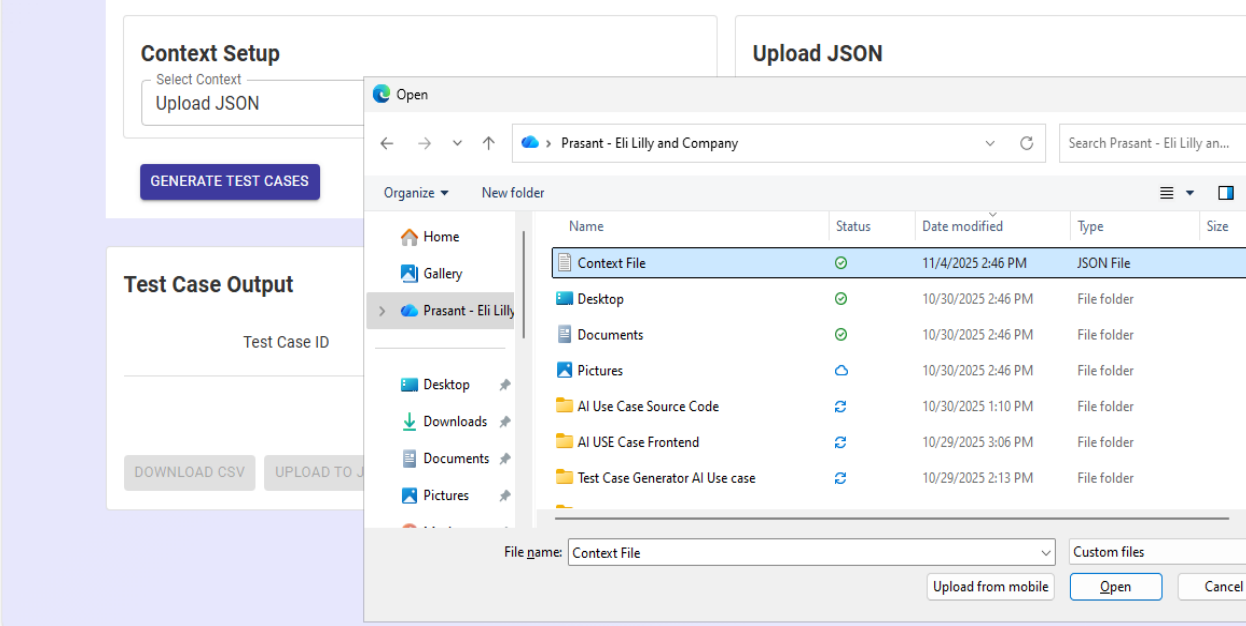
Step 4

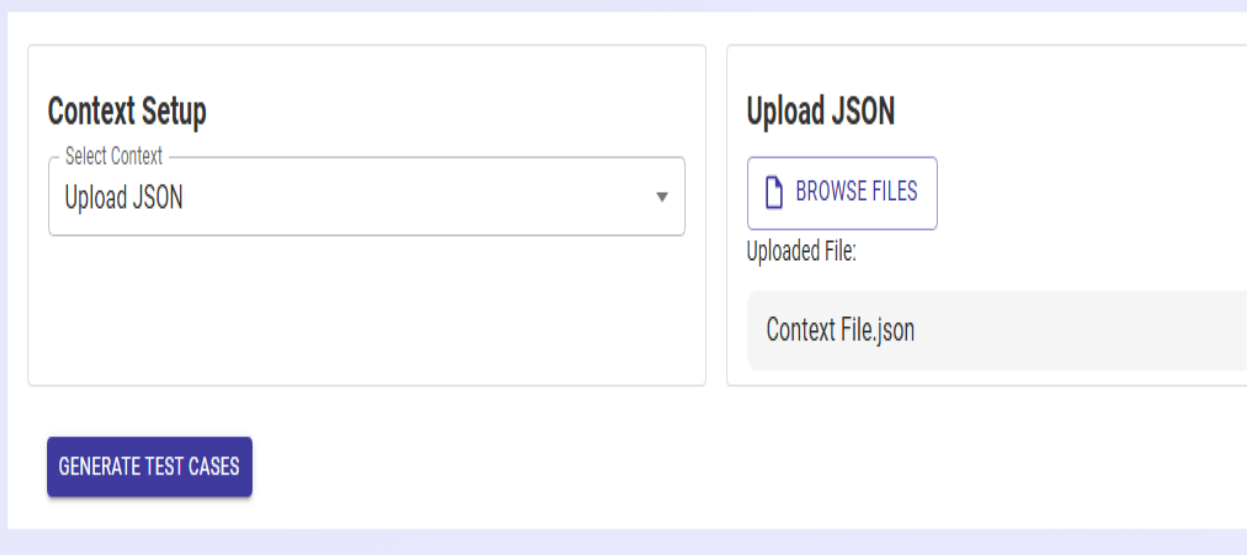
- Set up the context under the Context Setup section by selecting any of the desired options from the dropdown.



- If Upload JSON/YAML/Plain Text is selected:
    - Upload the specific file in a properly formatted way with all required information as mentioned under the Context File Setup section of this manual.



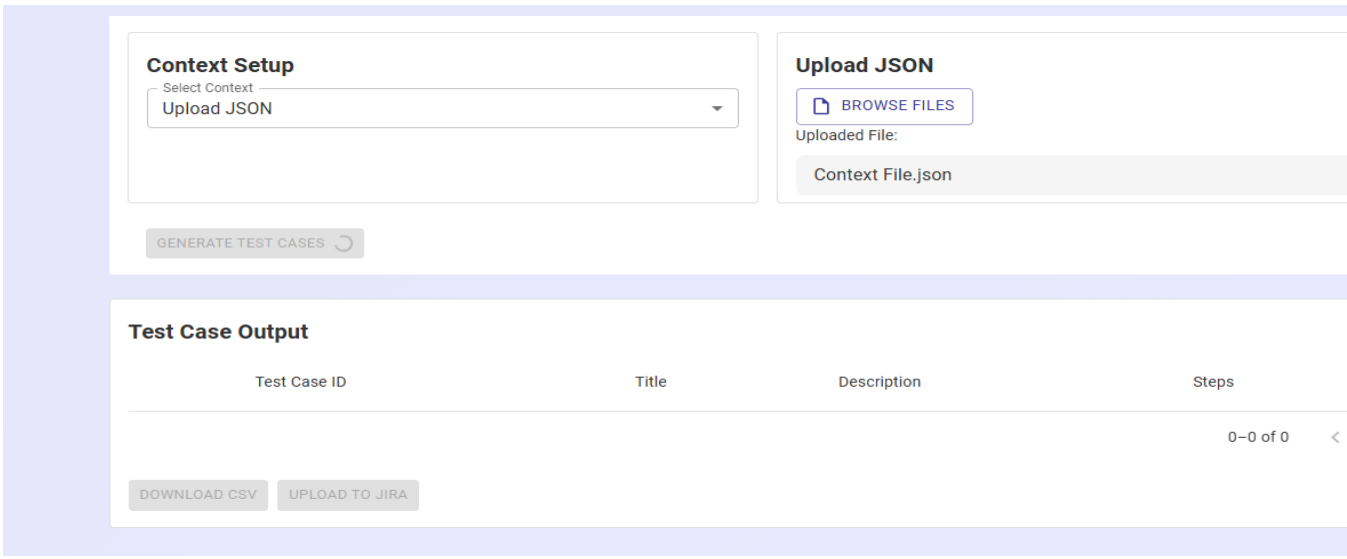


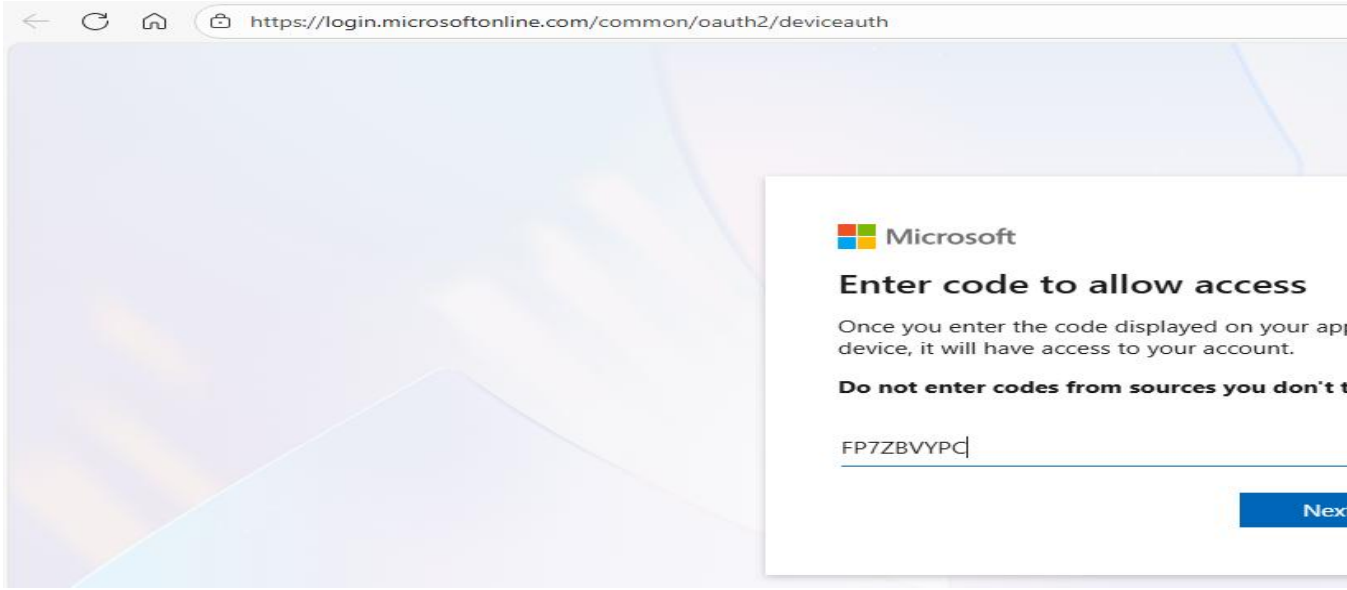


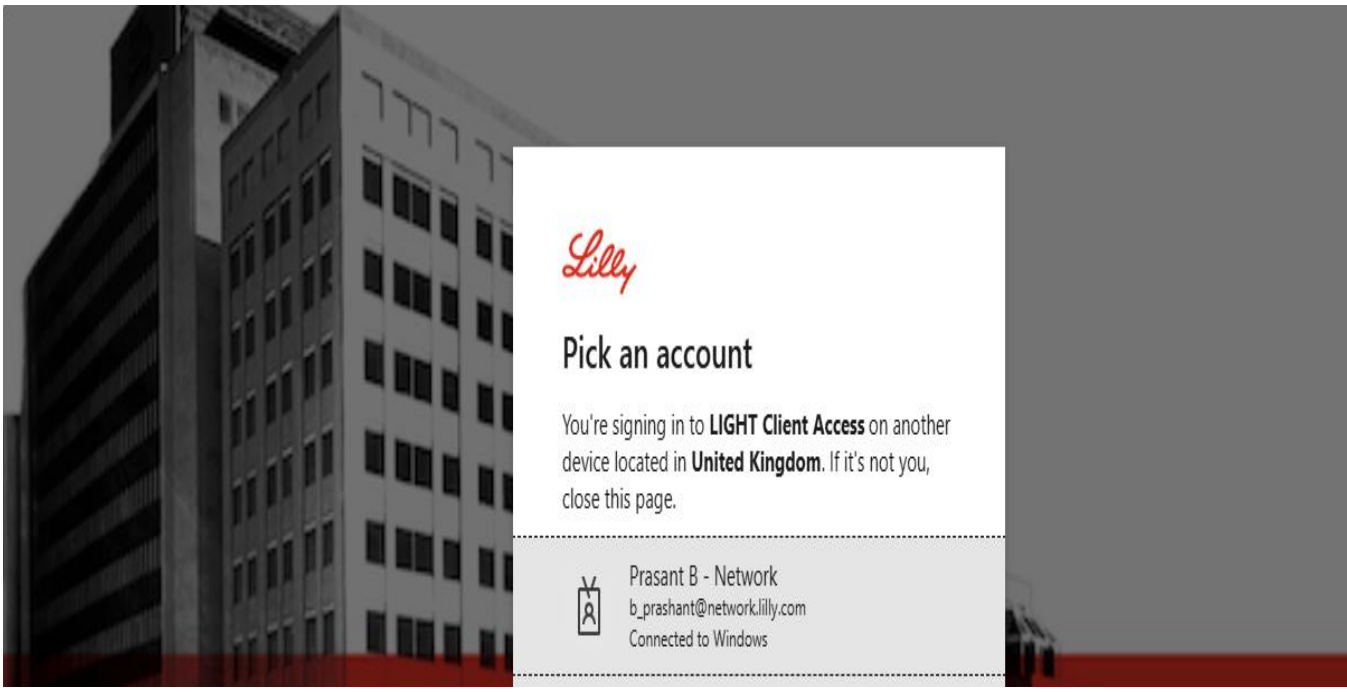
- If Scrape URL is selected:
    - Provide the appropriate URL so that the LLM model can scrape the web page and extract the required information.
- If Upload Existing Test Case is selected:
    - Upload the existing test case strictly in CSV format as reference.
- Note: The Browse Files upload option will only be applicable and enabled for the options starting with Upload in the Select Context dropdown under the Context Setup section.

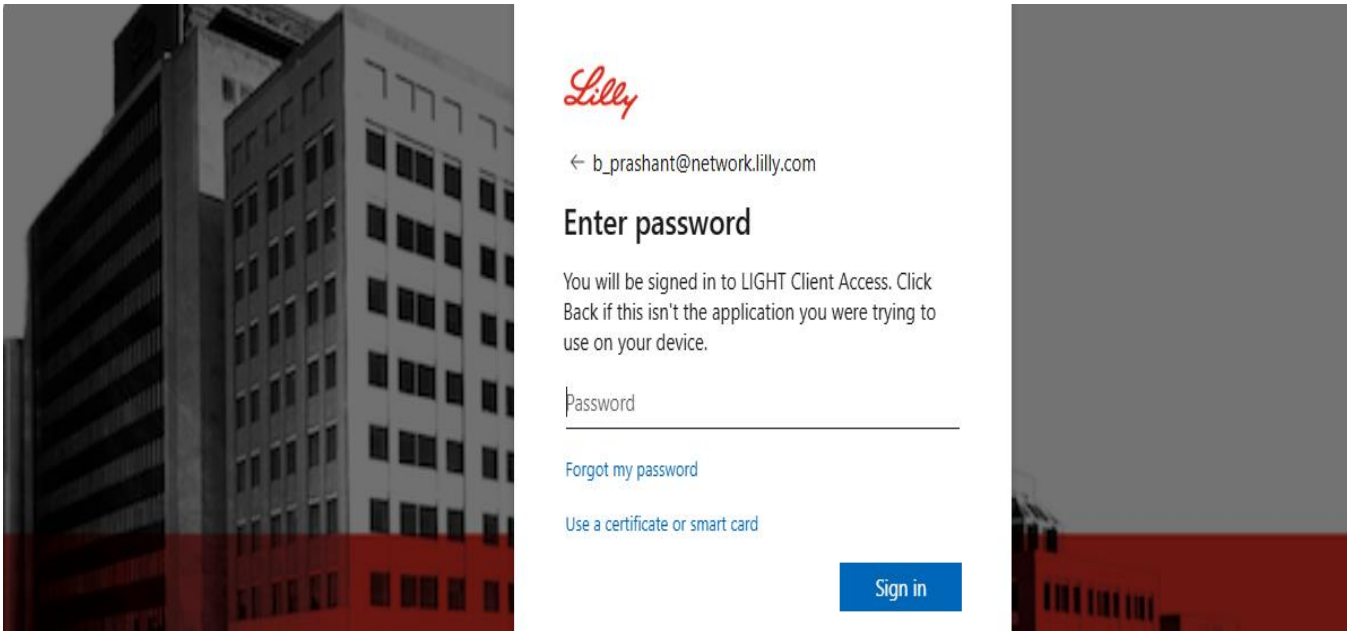
Step 5

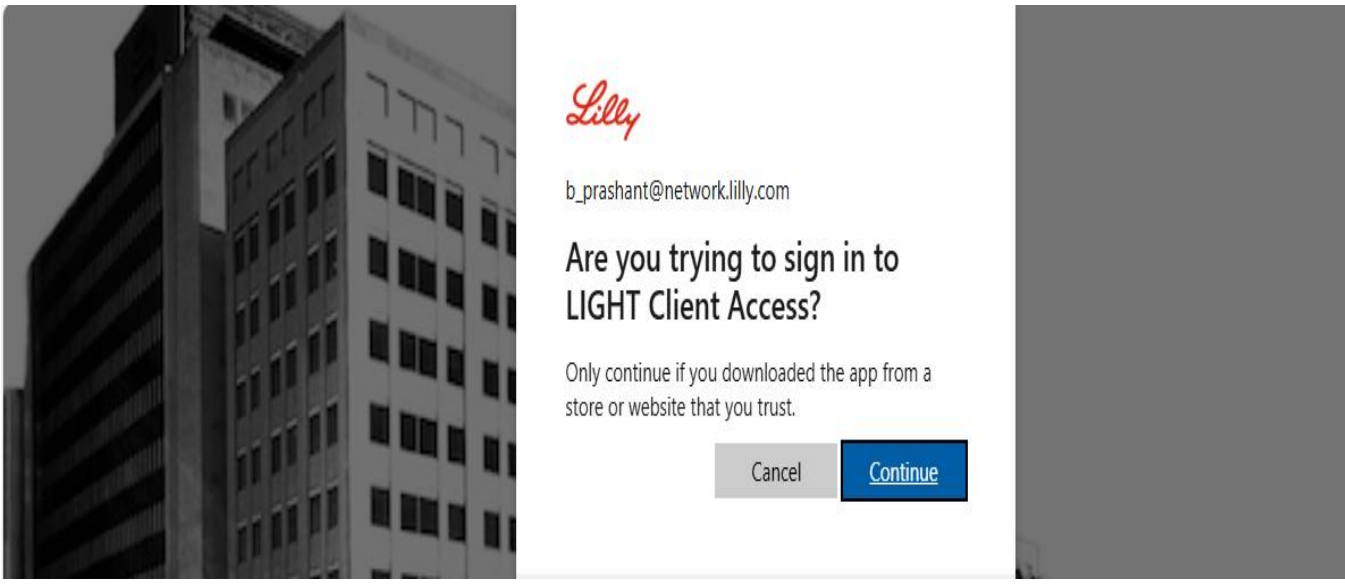
- Once everything is completed in Step 4, click on the Generate Test Cases button. Post clicking on the button, launch URL https://microsoft.com/devicelogin and enter code (one time) and click on Next. User is redirected to Microsoft login page where users need to pick respective Lilly email ID and consequently enter password and click Sign In. Next user will be redirected to **LIGHT Client Access** approval page where they need to click on Continue button. Refer to corresponding screenshots below

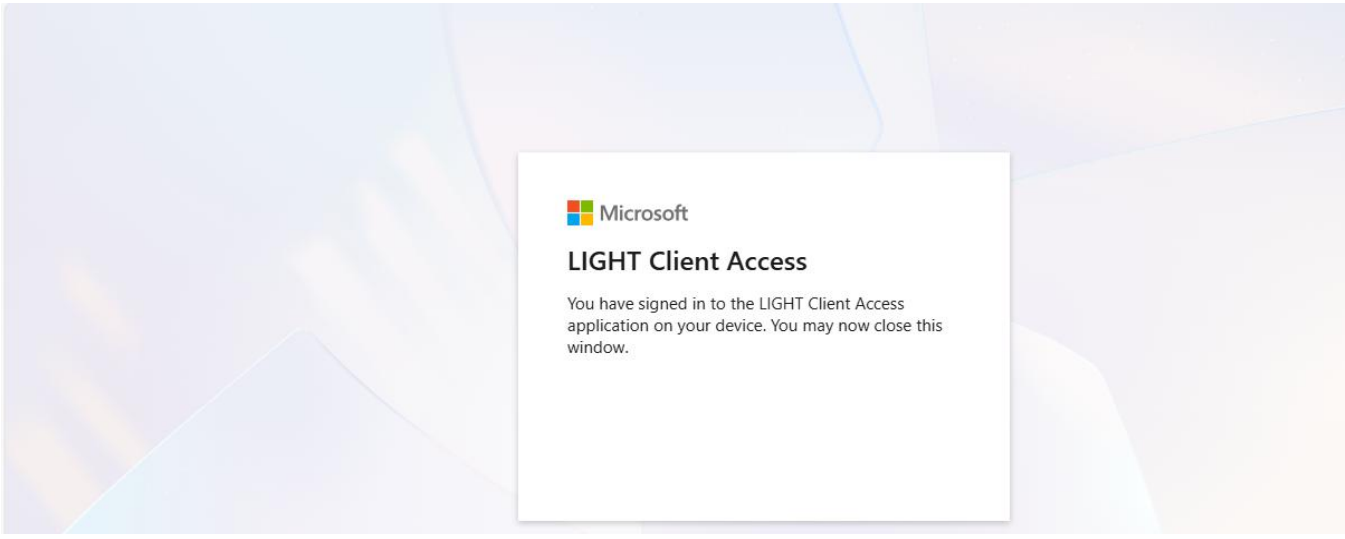




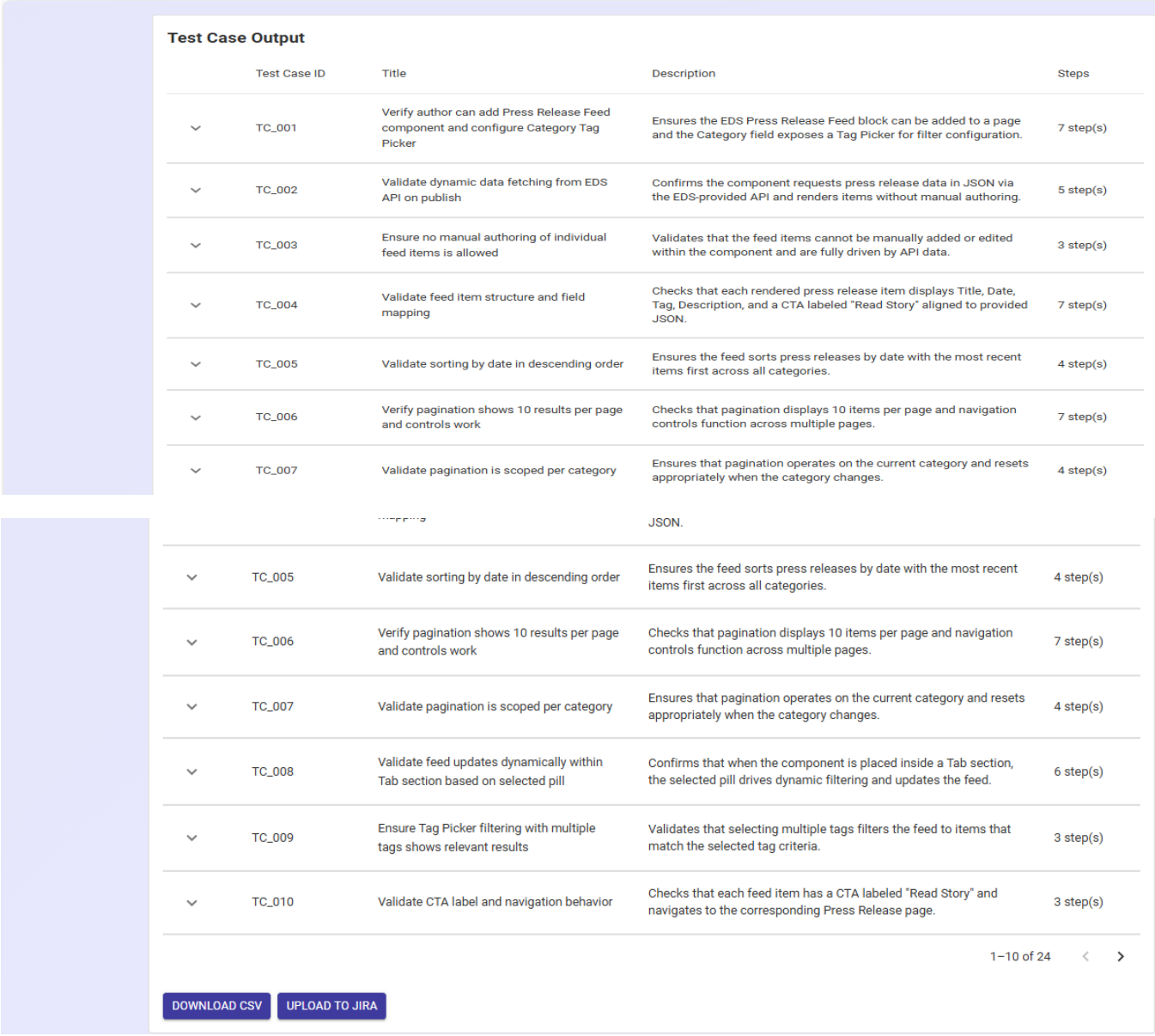

Pick an account
You're signing in to LIGHT Client Access on another device located in United Kingdom. If it's not you, close this page.
Prasant B - Network
b_prashant@network.lilly.com
Connected to Windows


b_prashant@network.lilly.com
Enter password
You will be signed in to LIGHT Client Access. Click Back if this isn't the application you were trying to use on your device.
Password
Forgot my password
Use a certificate or smart card
Sign in





- The tool will generate test cases based on the information provided.
- The generated test cases will be visible under the Test Case Output section.



Test Case Output

	Test Case ID	Title	Description	Steps
⌄	TC_001	Verify author can add Press Release Feed component and configure Category Tag Picker	Ensures the EDS Press Release Feed block can be added to a page and the Category field exposes a Tag Picker for filter configuration.	7 step(s)
⌄	TC_002	Validate dynamic data fetching from EDS API on publish	Confirms the component requests press release data in JSON via the EDS-provided API and renders items without manual authoring.	5 step(s)
⌄	TC_003	Ensure no manual authoring of individual feed items is allowed	Validates that the feed items cannot be manually added or edited within the component and are fully driven by API data.	3 step(s)
⌄	TC_004	Validate feed item structure and field mapping	Checks that each rendered press release item displays Title, Date, Tag, Description, and a CTA labeled "Read Story" aligned to provided JSON.	7 step(s)
⌄	TC_005	Validate sorting by date in descending order	Ensures the feed sorts press releases by date with the most recent items first across all categories.	4 step(s)
⌄	TC_006	Verify pagination shows 10 results per page and controls work	Checks that pagination displays 10 items per page and navigation controls function across multiple pages.	7 step(s)
⌄	TC_007	Validate pagination is scoped per category	Ensures that pagination operates on the current category and resets appropriately when the category changes.	4 step(s)
			JSON.	
⌄	TC_005	Validate sorting by date in descending order	Ensures the feed sorts press releases by date with the most recent items first across all categories.	4 step(s)
⌄	TC_006	Verify pagination shows 10 results per page and controls work	Checks that pagination displays 10 items per page and navigation controls function across multiple pages.	7 step(s)
⌄	TC_007	Validate pagination is scoped per category	Ensures that pagination operates on the current category and resets appropriately when the category changes.	4 step(s)
⌄	TC_008	Validate feed updates dynamically within Tab section based on selected pill	Confirms that when the component is placed inside a Tab section, the selected pill drives dynamic filtering and updates the feed.	6 step(s)
⌄	TC_009	Ensure Tag Picker filtering with multiple tags shows relevant results	Validates that selecting multiple tags filters the feed to items that match the selected tag criteria.	3 step(s)
⌄	TC_010	Validate CTA label and navigation behavior	Checks that each feed item has a CTA labeled "Read Story" and navigates to the corresponding Press Release page.	3 step(s)

1–10 of 24 ‹ ›

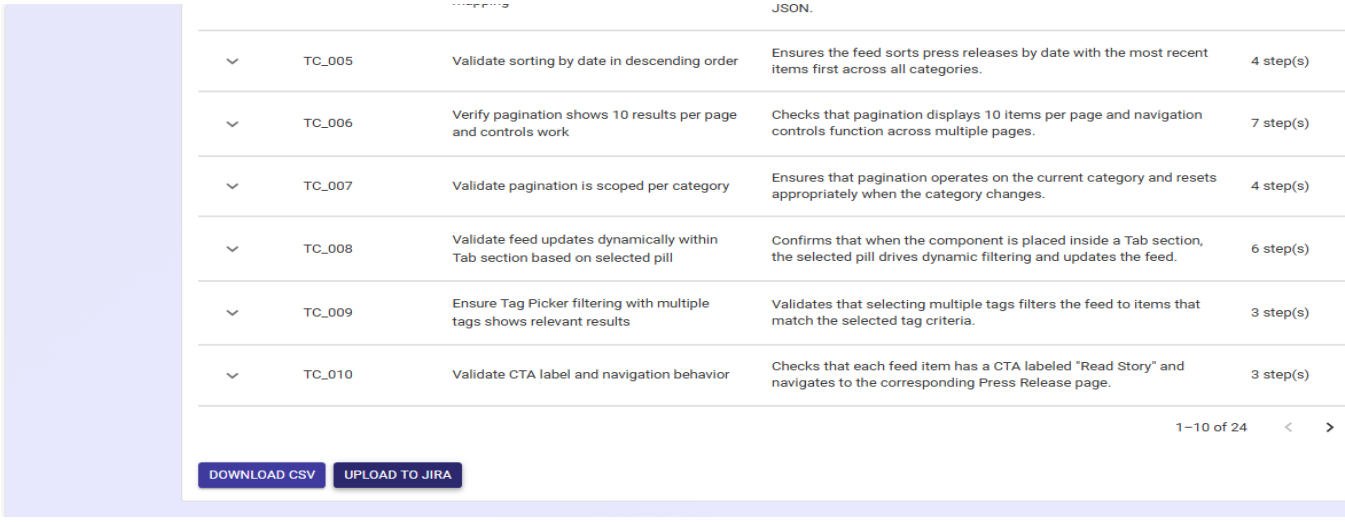
DOWNLOAD CSV UPLOAD TO JIRA

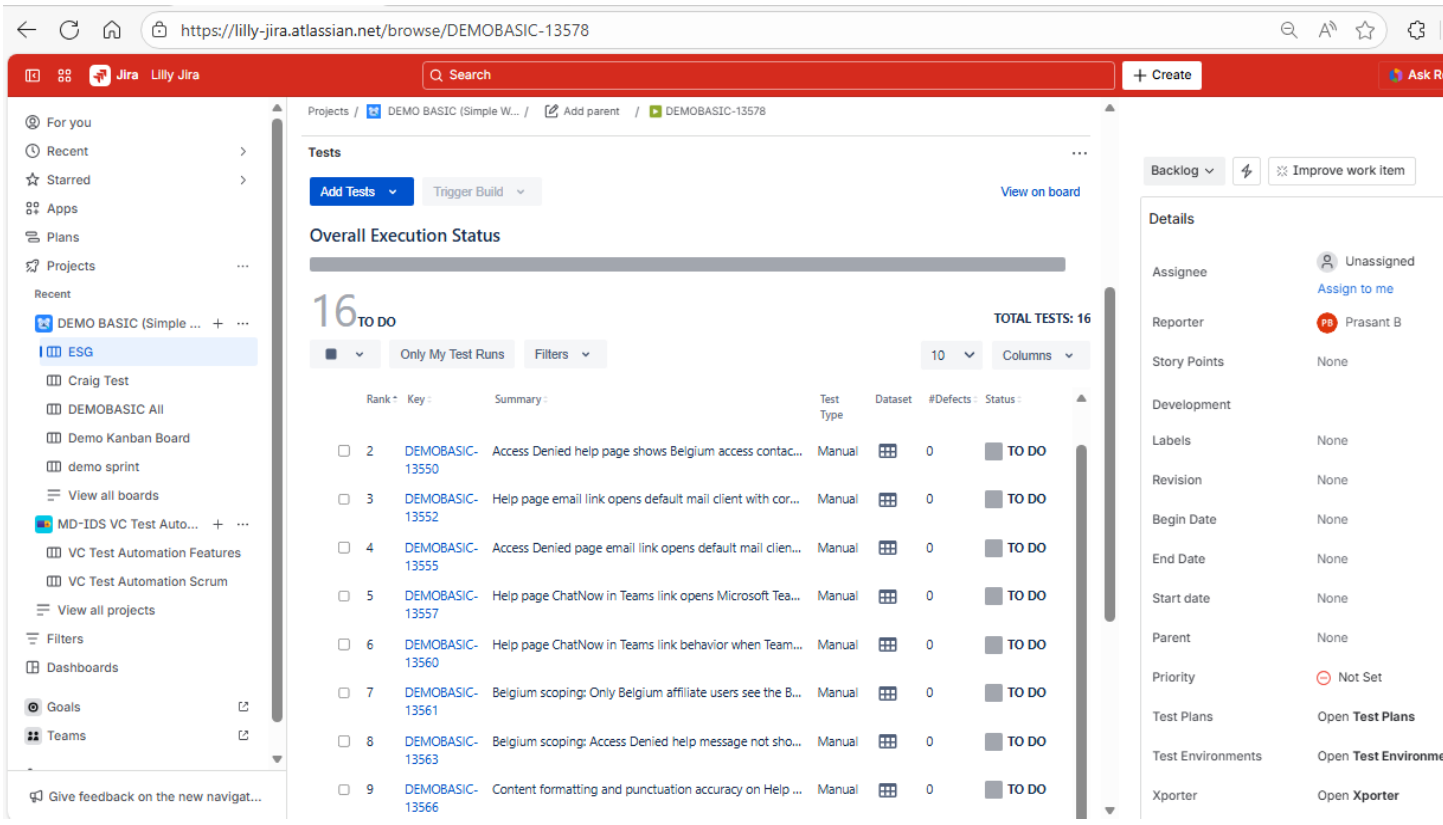
- Users can download the generated test cases in CSV format by clicking on the Download CSV button.

	Test Case	Summary	Description	Step Numbe	Action	Data	Expected Result
2	TC_001	Verify author can add Press Release Feed component and configure Category Tag Picker	Ensures the EDS Press Release Feed block can be added to a page and the Category field exposes a Tag Picker for filter configuration.	1	Open the target AEM author page in edit mode	NA	The page should be displayed in author mode with component editing controls available.
3				2	Open the component browser and locate the EDS block named Press Release Feed	NA	The component list should include an entry labeled Press Release Feed.
4				3	Drag and drop the Press Release Feed component into the desired region on the page	NA	The Press Release Feed component should appear on the page with default configuration.
5				4	Open the Press Release Feed component configuration dialog	NA	The configuration dialog should display a configurable field named Category with input type Tag Picker.
6				5	Hover or focus the Category field description	NA	The description should state Select tag(s) for feed filtering.
7				6	Select one or more tags using the Tag Picker	TagA, TagB	The selected tags should be displayed in the Category field and should be saved with the component upon closing.
8				7	Save the component configuration and reload the author page	NA	The component should retain the configured Category Tag Picker selections after reload.
9	TC_002	Validate dynamic data fetching from EDS API on publish	Confirms the component requests press release data in JSON via the EDS-provided API and renders items without manual	1	Publish the page containing the Press Release Feed component	NA	The published page should be accessible on the publish environment.
							The component should initiate a network request to the configured EDS API

Step 6

- To upload the generated test cases to the JIRA project workspace, click on the Upload to JIRA button.




https://lilly-jira.atlassian.net/browse/DEMOBASIC-13578
Jira Lilly Jira
Search
Create
For you
Recent
Starred
Apps
Plans
Projects
Recent
DEMO BASIC (Simple ...
ESG
Craig Test
DEMOBASIC All
Demo Kanban Board
demo sprint
View all boards
MD-IDS VC Test Auto...
VC Test Automation Features
VC Test Automation Scrum
View all projects
Filters
Dashboards
Goals
Teams
Give feedback on the new navigat...
Projects / DEMO BASIC (Simple W... / Add parent / DEMOBASIC-13578
Tests
Add Tests
Trigger Build
View on board
Overall Execution Status
16 TO DO
TOTAL TESTS: 16
Only My Test Runs
Filters
10
Columns
Rank Key Summary Test Type Dataset #Defects Status
2 DEMOBASIC-13550 Access Denied help page shows Belgium access contac... Manual 0 TO DO
3 DEMOBASIC-13552 Help page email link opens default mail client with cor... Manual 0 TO DO
4 DEMOBASIC-13555 Access Denied page email link opens default mail clien... Manual 0 TO DO
5 DEMOBASIC-13557 Help page ChatNow in Teams link opens Microsoft Tea... Manual 0 TO DO
6 DEMOBASIC-13560 Help page ChatNow in Teams link behavior when Team... Manual 0 TO DO
7 DEMOBASIC-13561 Belgium scoping: Only Belgium affiliate users see the B... Manual 0 TO DO
8 DEMOBASIC-13563 Belgium scoping: Access Denied help message not sho... Manual 0 TO DO
9 DEMOBASIC-13566 Content formatting and punctuation accuracy on Help ... Manual 0 TO DO
Backlog
Improve work item
Details
Assignee Unassigned
Assign to me
Reporter Prasant B
Story Points None
Development
Labels None
Revision None
Begin Date None
End Date None
Start date None
Parent None
Priority Not Set
Test Plans Open Test Plans
Test Environments Open Test Environme
Xporter Open Xporter