Intelligent Automation Framework

IAF/intelligent Automation Framework is tool developed to assist in automating tasks such as testing.There are 3 main webpages namely the dashboard,projects and reports.
There is a cpu core utilisation showing as a line graph calculated in percentage(x axis - time ,y axis - cpu core utilisation(in percentage))
There is a memory utilisation also shown as a graph (x axis - time ,y axis - memory utilisation(in percentage))

Health Check status:
It checks whether Cache backend,Celery HealthCheck,Celery Ping,PostgreSql,FileStorage,Disk Usage,Glance,Memory Usage,Migrations,Redis is up or has errors or is being down or timeout.
It is up then it will be shown as green else it will be red

We can logout from the website by clicking the “thinkpalm” user and then logout option
We can see “activity logs” ,”profile’ ,”settings” by clicking “thinkpalm” user

On clicking the notification bell icon we can select action,another action in an dropdown

There is a Group run and testbed in the dashboard

There is also a pie chart and histogram in this dashboard

On clicking Reports in the nav bar a drop down appears that gives the option to select Execution result and analysis,we can see the execution result of various testcases here
There is a projects option that goes into project page in the navbar

1.Projects Page:

Here you can add different projects ,click on the ‘+’ icon on the projects section ,to add a new project.
Input the name and summary of the project.It will be added and displayed in that section.Once the project is listed ,different icons corresponding to different functionalities can be accessed against each project.User can see-view,edit,delete ,topology,flow,develop,testplan analyzer icons against each project.By clicking on view ,edit and delete they can perform corresponding operations.So from this projects section user can navigate to the topology ,develop,flow testplan analyzer.Detailed usage of these sections are given below.
1.1 Toplogy Section
When u click on the topology button it navigates you to a page where you can view topology of a project.here ,`#`: Indicates the serial number or ID of the topology.
`Owner`: Represents the entity or user who owns the topology. For example, the owners are "core," "user," and "saas." These might represent different modules or layers within your project.
`Topology Name`: Identifies the name of the topology. It likely describes the specific system, configuration, or environment being referred to.
`Category`: Indicates the category of the topology. It suggests topologies fall under which category, suggesting that these topologies are being used for which type of testing like sanity or regression (basic functionality validation).
`Public`: This column shows whether the topology is public or private.For example, all entries are marked as `False`, meaning these topologies are private and not accessible publicly.

Action Buttons:

- **Clipboard Icon**: used to **duplicate the testbed** implies that clicking it will create a copy of the selected topology or test environment.
- **List Icon**: Navigate to scenario section where users can manage different test scenarios .
- **Link Icon:**Allows you to select multiple scenarios or tests and run them as a batch.
- **Calendar Icon**: used for scheduling tasks or tests related to this topology.
- **Three Dots**: A menu for more actions, such as editing, deleting, or configuring the topology.

1.2Develop

On clicking develop users get navigated to a test case creation or execution interface.**Sections Explanation:**

1. **TestCaseInfo:**
    - **Purpose**: Captures details about the test case.
    - **Fields:**
        - **Functionality**: Indicates the functionality being tested.
        - **Topology**: Links the test case to a specific topology setup.
        - **TestCase Name**: Specifies the name or identifier of the test case.
2. **Topology:**
    - **Purpose**: Selects the testbed topology on which this test case will run.
    - **Dropdown**: Likely lists all available topologies to choose from.
3. **Precondition:**
    - **Purpose**: Lists the conditions that must be satisfied before the test case execution.
    - **Selection**: Allows linking of preconfigured preconditions to the test.
4. **Profiles:**

    - **Purpose**: Enables selection of user profiles or system configurations required during testing.
    - **Usage**: May include roles, permissions, or environmental variables.
5. **TestSteps:**
    - **Purpose**: Defines the sequence of actions or steps executed as part of the test.
    - **Dropdown**: Allows adding, editing, or deleting individual test steps.
6. **SyntheticData:**
    - **Purpose**: Provides test data generated or injected during test execution.
    - **Relevance**: Useful for simulating realistic test conditions without using actual data.
7. **PostCondition:**
    - **Purpose**: Defines actions or checks performed after test execution.
    - **Usage**: Ensures cleanup, validation, or resetting the environment.
8. **Save/Execute/Generate Script:**
    - **Save**: Saves the test case configuration for future use.
    - **Execute**: Runs the test case immediately in the selected topology.
    - **Generate Script**: Converts the test case configuration into a reusable script (likely for automation).

1.3 Flow

1.4 Testplan analyzer

Purpose of Testplan Analyzer

- **CSV Upload**: The uploaded CSV file may contain details like:
    - List of test cases.
    - Test steps or configurations.
    - Expected outcomes or parameters.

    - Execution schedules or dependencies.
- **Validation**: The analyzer probably checks the uploaded data for:
    - Completeness of the test plan.
    - Consistency or alignment with the selected test topology.
    - Identification of any errors, gaps, or duplicates.
- **Optimization**: Provides insights to optimize test coverage, execution order, or configurations

Reports Page:
1. Dashboard Overview

- **Pie Chart or Completion Overview (Top Left)**:
  - Shows a graphical representation of the test execution results, likely indicating the percentage of completed tests or success/failure distribution.
- **Notifications Section (Top Right)**:
  - Contains a notification icon indicating updates or alerts related to the tests.
- **User and Branding (Top Right)**:
  - Includes user details or the organization logo

---

2. Filters Section

- A set of filters is provided to refine the search or view for test case execution results:
  - **Testcase Name**: Allows filtering results by a specific test case.
  - **Execution Status**: Filters based on the status of the test execution (e.g., completed, pending).
  - **Run Result**: Filters by the outcome (e.g., pass, fail).
  - **Date Range**: Filters executions within a specific time range.

---

3. Execution Results Table

The table displays detailed information about executed test cases:

- **Columns**:
  1. **#**: Serial number of the entries.
  2. **Group Run Name**: Indicates the group under which the test was executed (e.g., `default` here).
  3. **Testcase Name**: The specific test case name executed (e.g., `flow_prem`).
  4. **Execution Status**: Status of execution (e.g., `Completed`).

5. **Scheduled At**: Time when the test was scheduled to start.
6. **Started At**: The actual start time of the test.
7. **Finished At**: The time when the test execution ended.
8. **Duration**: Total time taken for execution.
9. **Execution Result**: The outcome of the test (e.g., `Fail` in this case).

---

4. Actions

- On the right-hand side of the row (last column):
  - **Download Icon**: Likely allows downloading logs or execution details.
  - **Eye Icon**: To view more detailed results or logs.
  - **Retry Icon**: To rerun the test case.

Overall Purpose

This interface is designed for managing and monitoring test case executions in an automated system. It provides both a high-level overview and detailed insights into each test case run, including execution timing, status, and results.