TECHNICAL GUIDE

**Installation**

Playwright is a NodeJS based package and can be installed using NPM. It can be installed either at system level or project level. For more information, see this: [Installation | Playwright](https://playwright.dev/docs/intro)

**UI Automation**

Project Structure

This document recommends using TypeScript for Playwright UI automation and the project structure like below (in VSCode)

A screenshot of a computer

Description automatically generated

Directory details:

|  |  |  |  |
| --- | --- | --- | --- |
| - | **.vscode**  This contains a “settings.json” file that controls how VSCode works for the particular project. This is included in the project because we need to update settings.json to ensure cucumber works properly. | | |
| - | **config**  This directory is meant to host configuration files for 3rd party packages used in the project. It should not be used for Playwright configurations (playwright.config.ts) and TypeScript configurations (tsconfig.json); those files should be at project root level. | | |
| - | **src**  This is the o | main  have any files but  **src/helpers** This test.  ▪ | directory that contains automation implementation. This directory does not have several sub-directories as explained below:  directory contains general implementation not specific to the application under  **src/helpers/browser**  This directory contains a file called ‘browserManager.ts’.  Playwright, by default, works against a browser engine (like Chromium) rather than a vendor specific browser (like Google Chromium). While invoking a browser, one can specify various options such as which browser type should be used, whether browser should be launched in headless mode, etc., All those options are managed centrally in ‘browserManager.ts’. |
|  |  | ▪ | **src/helpers/env**  This directory contains various environmental files such as staging, etc., |
|  |  | ▪ | **src/helpers/report**  This directory contains implementation related to test reports. |
|  |  | ▪ | **src/helpers/types** this has a file with NodeJS related implementation to manage environment keys. |
|  |  | ▪ | **src/helpers/util**  This directory contains code for remaining utility related tools. As of this writing (i.e., v1.0 of this document), it contains below files: dataBag.ts and logger.ts.  In addition to above files, this also has below sub-directory to host test data related model files.  • **src/helpers/util/test-data -** this directory is specific to application under test. |
|  |  | ▪ | **src/helpers/wrapper**  This is meant for providing implementation for general Playwright approaches. E.g., retrieving a dropdown item (i.e., finding an element by ‘**option**’), waiting for page stability (i.e., waiting for ‘**domcontentloaded**’, etc., |
|  | o | **src/hooks**  In addition to, scenario specific code, Cucumber can run code for following: | |

* execute code before and after running all scenarios (this runs only once for all scenarios in all feature files)
* execute code before and after running every scenario (this runs once for each scenario in all feature files)

Such kind of common execution logic is captured in “hooks”.

Examples: general initializations (such as environment, browser, etc.,), scenario specific initialization (such as Data Bag, logger, etc.,)

* **src/pages**

This directory contains application under test specific code.

* **src/tests**

This directory contains features and step implementations.

* + **src/tests/features**

BDD features must be defined here.

* + **src/tests/steps**

BDD features step implementations must be defined here.

* **test-data-store**

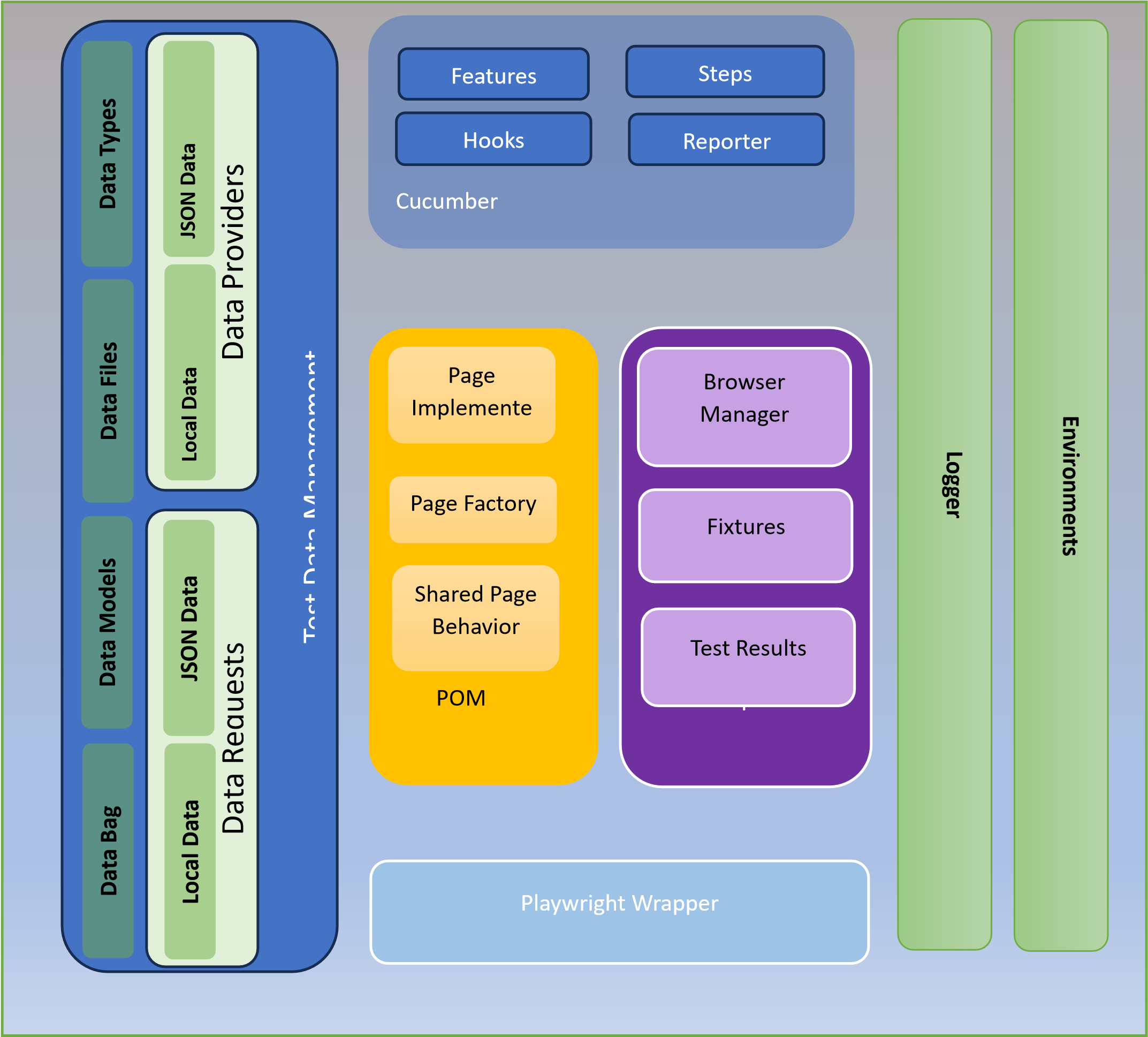
This directory contains the test data that needs to be used for test execution.

* **test-results**

Once tests are executed, results will be stored in this directory. Results contain: log files (in logs folder), video recordings (in videos folder). The complete Cucumber report is also stored here.

Building Blocks

Below diagram depicts different conceptual blocks involved in writing automation code. These are explained in detail in further sections.



**Installation Commands**

**1. Install Node.js**

Ensure Node.js (version 14 or later) is installed on your system. If not, download and install it from[**Node.js**](https://nodejs.org/)**.**

**2. Set Up Project Dependencies**

Run the following commands in the terminal to install all the required dependencies:

**Step 1: Install Playwright**

Install Playwright with the required browsers:

**npm install playwright**

This will automatically download the necessary browsers (Chromium, Firefox, and WebKit).

**Step 3: Install Cucumber.js**

Install Cucumber.js for writing BDD scenarios:

**npm install @cucumber/cucumber**

**Step 4: Install Additional Libraries**

Install other necessary libraries:

**npm install dotenv**

**Step 5: Install a Test Runner (Optional)**

You can also use Playwright’s test runner for managing test execution:

**npm install @playwright/test**

**Step 6: Install Development Dependencies**

Install optional packages like TypeScript or linters for better development practices (if applicable):

**npm install typescript ts-node eslint --save-dev**

**Step 7: Install Browsers for Playwright**

Run this command to download and install the required browsers (Chromium, Firefox, and WebKit):

**npx playwright install**

**Running the Tests**

**1. Run All Tests**

To execute all tests in the project, use this command:

**npm test**

**Run Tests by Tag**

To run tests associated with specific tags defined in your .feature files:

**npm test --tags= "@tagname"**

Example:

npm test --tags="@Home"