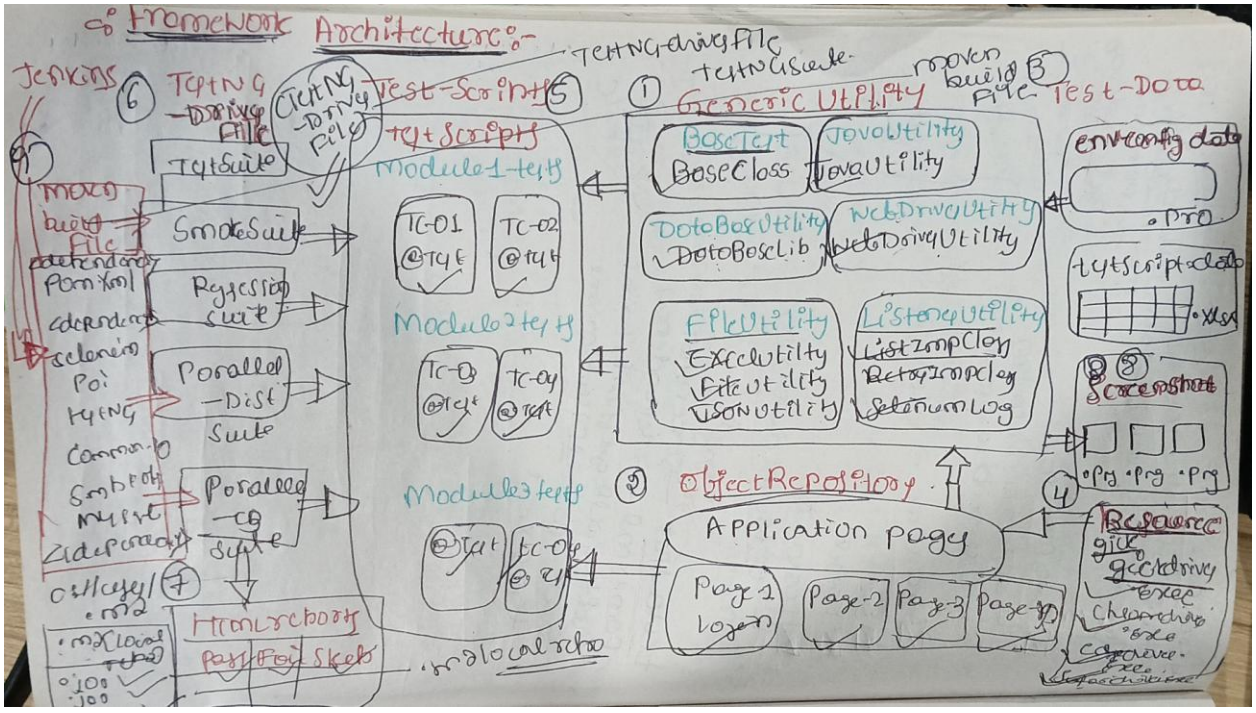
FRAMEWORK

In our project, we utilized a hybrid framework, which is a combination of Modular Driven Framework, Data Driven Framework, and Keyword Driven Framework. To create this hybrid framework, we structured our project as a Maven project with four main folders:

FRAMEWORK ARCHITECTUR



1. src/main/java

This folder contains the generic utility packages, which include methods that are applicable to all applications.

- **BaseClass**: Implements all basic configurations such as report configuration, database connection, launching and closing the browser, login, logout, closing the database connection, and report backup.
- **ListenerImplementationClass**: Implements overridden methods from `ISuiteListener` and `ITestListener` (e.g., `onStart`, `onFinish`, `onTestStart`, `onTestSuccess`, `onTestFailure`).
- **RetryListenerImplementationClass**: Implements the overridden methods of `IRetryListener` (e.g., `retry` method).
- **FileUtilityClass**: Contains generic methods to read data from properties files and JSON files.
- **ExcelUtilityClass**: Contains methods for reading and writing data to/from Excel files (e.g., `readDataFromExcel`, `setDataToExcel`).

- **DatabaseUtilityClass**: Implements methods for database connection, closing the database connection, executing select and non-select queries.
- **WebDriverUtilityClass**: Provides generic WebDriver-related methods such as maximizing the browser, implicit and explicit waits, switching to windows and frames, and performing mouse actions (e.g., `moveToElement`).
- **JavaUtilityClass**: Contains generic Java methods, such as generating random numbers and system dates.

After creating the generic utilities, we moved on to the **Page Object Model (POM)**.

Page Object Model (POM)

POM is a design pattern used to store web elements and their associated business methods on a per-page basis. For each module, we created a separate package. Inside each package, a class was created for every page, where we identified all the web elements using `@FindBy` and `@FindAll` annotations with private access specifiers.

- We initialized the web elements using the `PageFactory.initElements(driver, this)` method inside the constructor.
- Getters were used to encapsulate the elements, and business methods were created to interact with them.

The creation of these generic utilities and the POM happened during **Sprint 1**, which was the framework design and implementation stage.

---

2. src/main/resources

This folder stores all the data related to the generic utilities, such as common environment configuration data (properties files, JSON files, etc.).

---

3. src/test/java

Starting from **Sprint 2**, we began developing test scripts. In this folder, we created separate packages for each module. For each module, we implemented the test cases using the generic libraries, POM, common data, and test script data.

- All the dependencies required for developing the scripts were added to the `pom.xml` file, which allowed Maven to automatically download the necessary JAR files and attach them to the project.

---

4. src/test/resources

This folder stores all the test-script related data, such as Excel files. It contains sub-folders like:

- **Manual Test Cases**: Stores manual test cases.
- **Screenshots**: Stores screenshots of failed test cases.
- **Advanced Reports**: Stores Extent reports.

Post Script Development

After the test scripts were developed, the code was pushed to **GitHub**. GitHub was integrated with **Jenkins**, which automatically executed the scripts whenever a new build was triggered. Jenkins created a suite file, executed the tests, and generated a report.

Once we received the reports:

- We analyzed the results, identified any failed test cases, and investigated the root cause.
- If it was a product issue, a bug was raised in **JIRA**.
- If it was a synchronization issue, we reran the tests, found the root cause, and fixed the issue.

This structured approach helped ensure efficient test automation and streamlined reporting and bug-tracking processes.