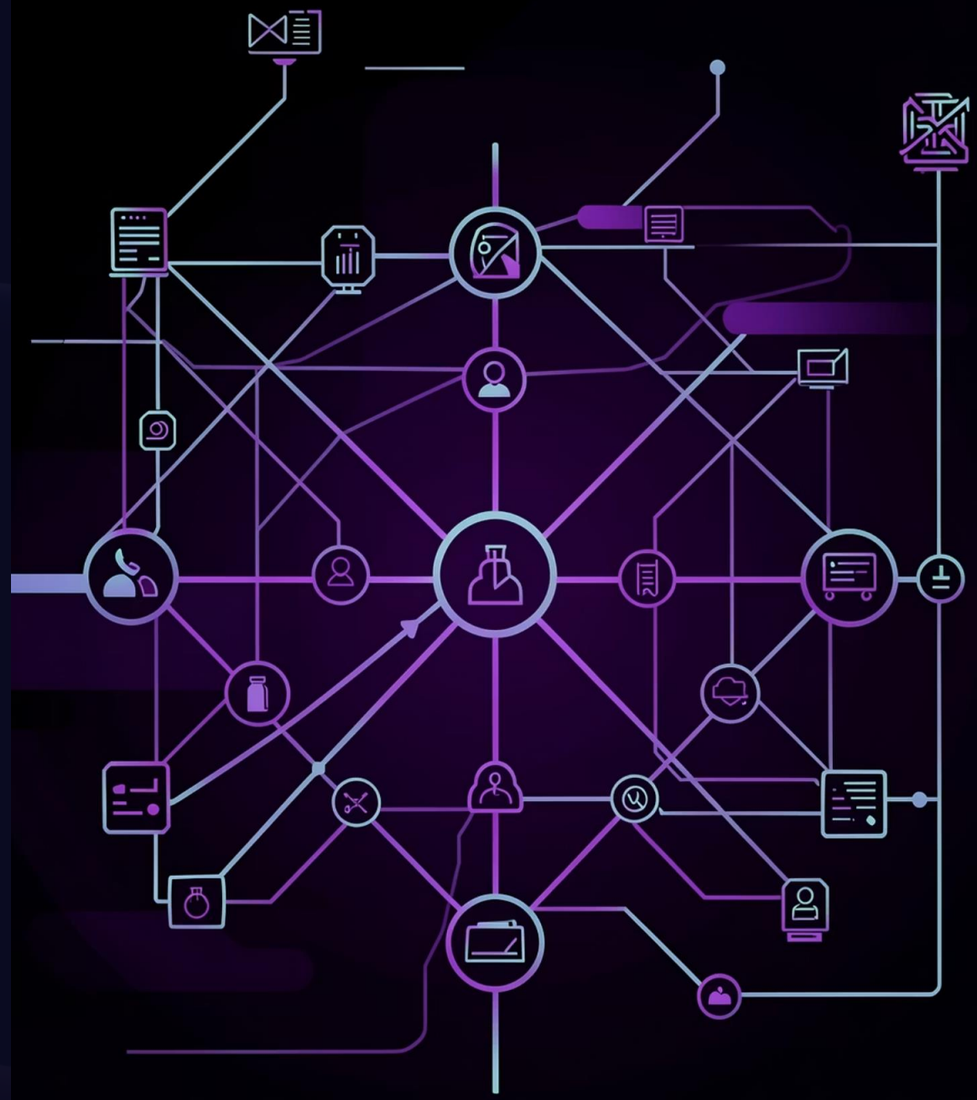
Booking Website Automation Framework

Built using Selenium, TestNG, Cucumber BDD, POM, Allure Reports, Log4j2 with Java + Maven.



Project Objective: Streamlining Booking Site Automation

Our primary goal was to automate key functionalities of a booking website, focusing on a seamless user experience from search to validation. This project aimed to establish a robust and maintainable automation framework using industry-standard tools and practices.

Automate Core Flows

Automate search, date selection, price filter, and results validation on a booking website.

Build a Comprehensive Framework

Integrate professional tools: Cucumber BDD, Page Object Model (POM), TestNG, Allure Reporting, and Log4j2 Logging.

Ensure Quality Outputs

Deliver readable tests, reusable components, detailed screenshots, and comprehensive logs for enhanced debugging and reporting.

Key Tools and Technologies Utilised

This project leveraged a carefully selected suite of languages, build tools, and automation frameworks to ensure a robust, scalable, and maintainable solution. Our choices reflect modern best practices in software testing and development.

Languages & Build

- **Java:** The core programming language for framework development.
- **Maven:** For project build automation and dependency management.

Automation & Testing Tools

- **Selenium WebDriver:** For browser automation and interaction.
- **Cucumber BDD (Gherkin):** For behaviour-driven development and readable test scenarios.
- **TestNG:** As the test runner, providing assertion capabilities and suite control.
- **Page Object Model (POM):** For structuring element locators and reusable actions.

Reporting & Monitoring

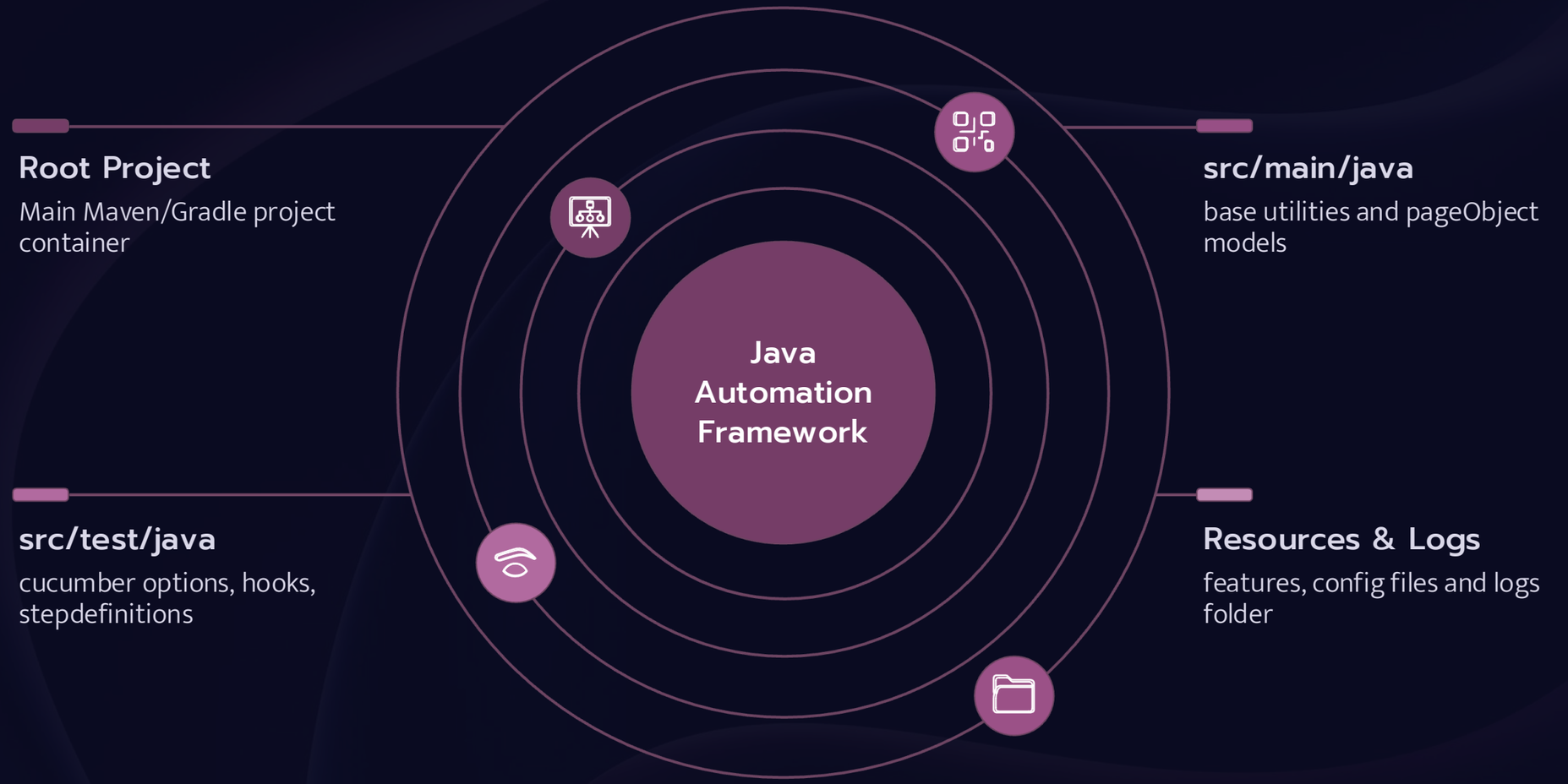
- **Allure Report:** For generating comprehensive, interactive test reports.
- **Log4j2 Logs:** For detailed logging of automation activities and errors.

Other Essentials

- **ChromeDriver:** Browser-specific driver for Chrome automation.
- **Thread waits, XPaths, properties file:** Essential components for robust element interaction and configuration management.

Framework Architecture: A Modular Design

Our framework is structured for clarity, maintainability, and scalability, adhering to best practices in automation development. The modular design separates concerns, making it easier to develop, debug, and extend.



This organised structure ensures that each component serves a specific purpose, contributing to the overall efficiency and readability of the automation suite. For instance, `Base.java` manages WebDriver instances, while `HomePage.java` encapsulates page-specific elements and actions.

Cucumber BDD: Bridging Business and Technical Layers

Cucumber BDD, powered by Gherkin syntax, was instrumental in creating highly readable and business-friendly test scenarios. This approach fosters collaboration between business stakeholders and technical teams, ensuring alignment on expected behaviours.

Gherkin Scenarios Covered

- Open booking page
- Enter destination
- Select check-in & check-out dates
- Apply price filter
- Validate results between min-max price

Key Advantages

- Produces highly readable test cases
- Uses business-friendly language
- Facilitates easy collaboration across teams

Page Object Model (POM): Enhancing Maintainability

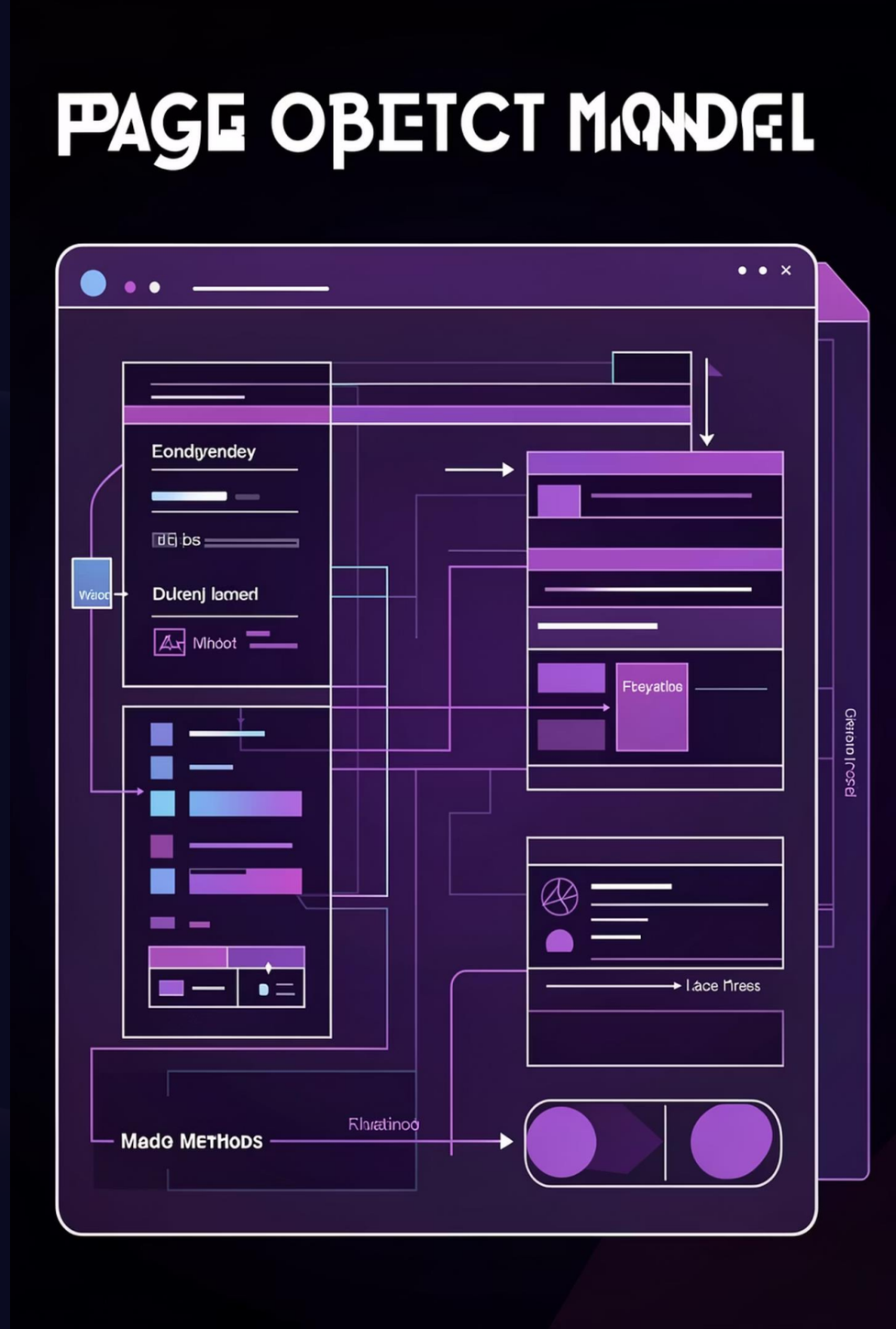
The Page Object Model (POM) design pattern was adopted to enhance the maintainability and readability of our automation code. By abstracting web pages as classes, we centralise element locators and reusable interactions, significantly improving framework robustness.

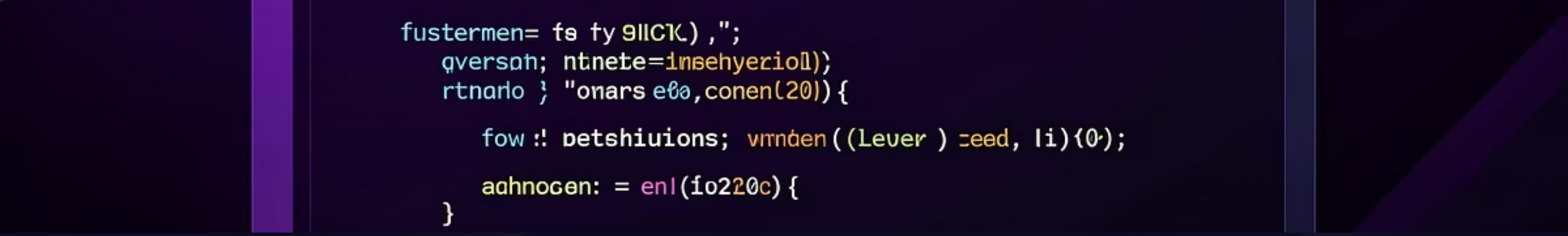
`HomePage.java` Capabilities

- Contains all element locators, primarily XPaths, for easy identification.
- Encapsulates reusable methods such as `openCalendar()`, `clickByXpath()`, `selectDates()`, `enterDestination()`, and `applyFilters()`.
- These methods perform specific actions on the page without exposing the underlying implementation details.

Core Benefits of POM

- Promotes code reusability across multiple test cases.
- Significantly reduces duplicate code, making the framework lean.
- Simplifies maintenance efforts when UI changes occur, as only page object classes need updates.
- Enhances overall test readability and understanding.





Step Definitions and Test Hooks: Orchestrating Test Execution

Our `BookingSteps.java` file translates human-readable Gherkin steps into executable code, interacting with the POM methods. `TestHooks.java` ensures a clean and controlled test environment, handling essential setup and teardown activities.

`BookingSteps.java` Responsibilities

Manages various actions:

- Interactions with search boxes.
- Date selection logic.
- Applying price filters.
- Validation of test results.

Crucially, it calls POM methods to maintain a clean separation of concerns between test logic and page interactions.

`TestHooks.java` Management

Handles critical lifecycle events:

- WebDriver initialization before scenarios.
- Proper WebDriver termination after scenarios.
- Capturing screenshots on both test success and failure.
- Attaching logs and screenshots directly to Allure reports.
- Logging scenario statuses for clear execution flow tracking.

Allure Reporting: Visualising Test Results

Allure Reporting provides an intuitive and comprehensive overview of test execution, making it easier to analyse results, identify failures, and track the overall quality of the automation suite. We integrated `allure-cucumber7-jvm` for seamless report generation.

Enhanced Reporting Features

- Automatically attaches screenshots for both passing and failing test steps.
- Includes detailed automation logs for every test run.
- Generates the report using the command: `allure serve target/allure-results`.

Key Allure Features Utilised

- Detailed test steps for clear traceability.
- Visual screenshots linked to each step.
- Comprehensive logs for deeper insights.
- Timeline view for understanding execution duration.
- Scenario history to track changes over time.
- Attachments for additional test data.
- Aesthetically pleasing UI with interactive graphs and metrics.

Logging with Log4j2: Unveiling Execution Insights

Effective logging is crucial for debugging, monitoring, and understanding the flow of automation. Log4j2 was implemented to provide detailed and categorised logs, essential for both development and production environments. This also integrated seamlessly with Allure reports.

Logging Implementations

- **automation.log:** Captures general informational messages (INFO), warnings (WARN), and errors (ERROR) to provide a complete picture of test execution.
- **error.log:** Specifically designed to capture only error messages, making it easy to identify and address critical issues quickly.

Logged Actions Include:

- Browser launch events.
- URL navigation to specific pages.
- Clicks on various web elements.
- Date selections within calendars.
- Detailed failure messages and stack traces.

Benefits of Robust Logging

- Simplifies the debugging process by pinpointing exact locations of failures.
- Provides clear visibility into the execution flow of automated tests.
- Invaluable during both development and production phases for issue resolution.
- Seamlessly integrated with Allure, ensuring logs are part of comprehensive test reports.

Final Outcome: A Professional Automation Framework

The culmination of this project is a fully functional, professional-level automation framework designed to handle the complexities of booking website testing efficiently and reliably. It embodies modern best practices and provides a solid foundation for future enhancements.

BDD Scenarios with Gherkin

Test cases are written in human-readable Gherkin, ensuring easy understanding and collaboration.

Robust Page Object Model

A well-structured POM ensures maintainability, reusability, and reduced code duplication.

Comprehensive Logging System

Detailed Log4j2 integration for efficient debugging and tracking of execution flow.

Automated Screenshots

Screenshots captured on both pass and fail scenarios, crucial for visual verification.

Interactive Allure Reports

Visually rich and detailed reports provide quick insights into test results and quality metrics.

Clean and Scalable Architecture

The framework is designed with a clear folder structure, making it easy to extend and adapt.

Real-World Booking Site Automation

Successfully automates a complex booking workflow, demonstrating practical application.