**High-Level Design (HLD) for BDD Lab**

**1. Overview**

The High-Level Design (HLD) provides an architectural overview of the BDD Lab. The design will highlight the architecture and components needed to execute BDD scenarios and testing for a web application. The use of BDD in software development emphasizes writing executable specifications that describe and define the features of an application.

**2. System Architecture  
2.1. Modular Architecture**

* **Feature Module:** BDD scenarios are written in Gherkin syntax, which uses plain language steps to describe the features. In this module, all feature files are together in its own directory. The significance of this module is that the behavior of the features is described. The feature files outline the BDD scenarios and expected outcomes for the modules.
* **Step Definitions Module:** Defines the features outlined in the previous module by providing functions that execute the steps given from the feature files and/or feature module.
* **Utilities Module:** Common functionalities are used for different modules to prevent duplicating code and promoting reusability throughout the application. For instance, the use of Scenario context in Behave can be used in the utilities module. Scenario context helps the functions interact with each other. The *context* variable in Behave will hold data specific to the scenario given. Title and tags are accessible, but arbitrary values can be added as well. This is an efficient way of passing data from one step to another.
* **Testing Module:** This module is used to define and execute tests to validate that the aspects of the application functions correctly. This includes BDD scenarios, features, and step definitions glued together to describe how the software conducts itself. The expected behavior of how different functionalities or modules interact with an application.

**3. Components   
3.1. Web Application**

* Represents the fruit stand application under testing circumstances. The web application is a motivating factor for testing the behavior of the software through BDD.
* Features such as product selection, product removal, or even clearing items from the cart can be tested using BDD.

**3.2. BDD Framework (Behave)**

* Behave will extract information from the Gherkin based feature files and execute the step definitions (functions) that are written in Python.
* Behave will manage test execution flow by abiding by the principles of BDD. The feature files that are used will describe the behavior of an application. Scenarios that are implemented will represent specific instances from the application. As mentioned earlier, a good example of this would be product selection. Lastly, step definitions to take action and verify expected results.

**3.3. WebDriver (Selenium)**

* Interacts with the web application by simulating the user’s actions. For instance, clicking buttons or selecting dropdown options.
* Has the ability to use scripts in programming languages such as Python to control the browser. The scripts can interact with the web elements to automate the testing process.

**3.4. Snipping Tool**

* The snipping tool can be useful to take snapshots of test results. Not only is this pleasing to the eye but can be helpful to add for documentation purposes.

**4. Data Flow**

1. **Tester writes scenarios:** Scenarios are written in Gherkin syntax and are documented in the features module.

2. **Step Definitions:** Since the scenario matches up with each step, Python functions are written in the step definitions module. This module executes the actions based on the scenarios and utilizes the web driver to interact with the application.

3. **Execution:** Behave starts the execution of tests by matching the steps in the feature files with the corresponding step definitions. This ensures that the steps are executed in a specified order. Lastly, Behave interacts with the web application through the WebDriver.

4. **Results Generation:** Detailed reports of the test execution are generated to provide an understanding of the test results. Inconsistencies in testing are documented as well. The report of the tests can help one document the challenges or assumptions made during the testing process.

**5. Integrations**

* **Continuous Integration (CI):** The BDD framework can be integrated with CI tools such as GitLab CI/CD, which allows one to define and execute Behave tests as part of the GitLab pipelines.
* **Version Control:** Git or GitHub are version control systems that can be utilized to manage feature files, step definitions, utility scripts, and other code documents related to testing.

**6.** **Scalability and Maintenance**

* The modular architecture and reuse of components is how Behave ensures scalability and maintenance for the software.
* The use of tags in Behave can be beneficial for categorization and selective execution of scenarios. Specific parts of tests can be managed easily when taking advantage of Behave’s tags.

**7. Conclusion**

This HLD delves into the surface area of the architectural structure in the BDD Lab. This design is intended to give insight on BDD, the framework Behave, and ensure understanding of how BDD can be implemented.