Sprint 3 - User story document

Sprint Document_RE-124- Implementation of CI/CD pipeline and Front-End Development

Document History

Author	Version	Date	Description	Comments
Giridharan Sankaran	**0.1**	**27-Feb-2024**	**Functionality and Technical document update**	
Yogesharvind Nedumaran	**0.3**	**29-Feb-2024**	**Test results document to be updated**	
Ashwin Krishna Mani	**1.0**	**01-Mar-2024**	**To check the documentation**	

1. **Background**

In the 'Implementation of CI/CD pipeline and Front-End Development' sprint, our team is currently focusing on enhancing the efficiency and reliability of our software development process. We are primarily concentrating on the implementation of Continuous Integration (CI) practices, streamlining the build and deployment pipelines to ensure seamless integration of code changes. Significant progress is being made in setting up the CI infrastructure, automating build processes, and integrating comprehensive testing suites. Additionally, we are dedicating efforts to advancing the Front-End development, particularly focusing on the Manager screen interface. While the primary emphasis is on the CI aspect, our team is also making strides in frontend design, completing a portion of the Manager screen layout and functionality. Alongside these development tasks, considerable time is being allocated to addressing bug fixes identified during testing phases, ensuring the stability and reliability of our software. Rigorous unit and integration testing are being conducted to validate the implemented features and ensure compatibility with existing functionalities, laying a solid foundation for future iterations.

2. **Proposed Functional Solution**

In our sprint titled 'Implementation of CI/CD pipeline and Front-End Development', we will make significant strides towards enhancing our development infrastructure and expanding frontend capabilities. We will implement a CI pipeline within GitLab, configuring unit integration and deploy test jobs to streamline our development workflow. On the frontend, we will leverage React Native to develop Supplier and Inventory pages for the Manager screen, providing essential functionalities for inventory management. While login screens have been created, authentication implementation will remain pending, which we plan to address using Spring Boot in subsequent iterations. Additionally, our team will conduct thorough unit and integration tests using JUnit and mockMvc frameworks, focusing on MongoDB collections such as 'Inventory', 'Expiry', 'Discount', and 'Supplier', to ensure robustness and reliability. This proposed solution will lay the groundwork for a cohesive and efficient development process, aligning with our project objectives.

1. Design Analysis and Solution:

3.1 Technical changes:

3.1.1 Design Analysis:

As part of this sprint, we need to implement CI pipeline to include unit, line, integration and deploy jobs for the tests and check its working flow. Also, we will be implementing unit and integration tests using Junit framework. Using React Native, we need to design the pages 'Inventory', 'Supplier', 'Expiry' and 'Discount' collection.

3.1.2 Building CI Pipeline:

In the phase of 'Building CI Pipeline', our focus lies on establishing a robust Continuous Integration (CI) infrastructure using GitLab. This entails creating comprehensive unit, integration, line, and deploy test jobs within the CI pipeline, meticulously designed to validate code changes at various stages of development. Integrating these tests seamlessly with our Git repository ensures that each code commit undergoes rigorous scrutiny, maintaining code quality and reliability. By verifying the functionality and integrity of our codebase through automated testing, we aim to enhance development efficiency and minimize the risk of introducing errors into our software. Our meticulous design analysis ensures that the CI pipeline aligns with our project requirements and standards, enabling smooth integration with our development workflow.

3.1.3 Unit and Integration tests:

Under the heading of 'Unit and Integration tests', our design analysis revolves around the meticulous creation of test cases using JUnit framework to validate the functionality and integration of key components within our software. By meticulously crafting unit tests, we ensure that individual units of code perform as expected, verifying their correctness in isolation. Additionally, integration tests are meticulously designed to assess the interactions between various components, ensuring seamless collaboration and functionality across the entire system. Through this comprehensive testing approach, we aim to detect and rectify any discrepancies or bugs early in the development process, thus bolstering the overall quality and reliability of our software. Our design analysis ensures that the test suite aligns closely with the project requirements and objectives, facilitating thorough validation of the implemented features.

3.1.4 React Native Front End:

Our design analysis encompasses the development of user interfaces for crucial collections including 'Supplier,' 'Inventory,' 'Expiry,' and 'Discount.' Leveraging the capabilities of React Native, we are crafting intuitive and responsive interfaces that enable seamless interaction with the underlying data. Our focus extends beyond mere display to encompass essential functionalities such as displaying all records, implementing filters to fetch data based on specific criteria, and facilitating operations like deletion and editing of records. Each component of the front end undergoes meticulous design consideration to ensure a cohesive and user-friendly experience. Through thoughtful design choices and adherence to React Native best practices, we aim to deliver a frontend solution that not only meets but exceeds user expectations, enhancing usability and productivity within our application.

4. **Use Test Cases:**

S.NO	Test Scenario	Expected Results
1.	To check if CI pipeline is working with unit test jobs.	All the unit test cases should pass in the pipeline without any issues
2.	To check if CI pipeline is working with line test jobs.	All the line job test cases should pass in the pipeline without any issues

3.	To check if CI pipeline is working with integration test jobs.	All the integration job test cases should pass in the pipeline without any issues
4.	To check if CI pipeline is working with deploy job.	All the deploy job test cases should pass in the pipeline without any issues
5.	To check if we are able to create multiple products in Inventory using POST API using unit and integration junit tests.	The testcase should get passed without any errors in both the tests
6.	To check if we are able to list all the products in Inventory collection using GET API	The testcase should get passed without any errors in both the tests
7.	To check if we are able to list the products in Inventory collection using UPCID with GET API.	The testcase should get passed without any errors in both the tests
8.	To check if we are able to list the products in Inventory collection using category with GET API.	The testcase should get passed without any errors in both the tests
9.	To check if we are able to list the products in Inventory collection using brand with GET API.	The testcase should get passed without any errors in both the tests
10.	To check if we are able to list the documents in Inventory collection using procurement date with GET API.	The testcase should get passed without any errors in both the tests
11.	To check if we are able to list the documents in Inventory collection using expiry date with GET API.	The testcase should get passed without any errors in both the tests
12.	To check if we are able to create multiple documents in Supplier using POST API.	The testcase should get passed without any errors in both the tests
13.	To check if we are able to list all the documents in Supplier collection using GET API	The testcase should get passed without any errors in both the tests
14.	To check if we are able to list all the documents in Supplier collection based on Brand_name using GET API	The testcase should get passed without any errors in both the tests