Process Deliverable 4

The provided test cases are designed to ensure that the Onboarding System meets its project scope, functional and non-functional requirements, and adheres to its design decisions. The test cases validate the critical aspects of the system that seek to improve and automate the developer onboarding process for new hires. These include:

1. Automated Repository Setup: Tests ensure that repositories are cloned, development environments are properly configured, and developers can begin working without need for manual intervention.
2. Centralized Access Management: Tests verify that developers are granted access to necessary tools resources, to reduce delays in the onboarding process.
3. Task Progress Tasking: Tests validate that developers can monitor their onboarding progress in real-time.
4. Automated Codebase Documentation: Tests confirm that the system generates, updates, and allows developers to search project documentation seamlessly.
5. Codebase Documentation Search: Tests confirm that developers can properly commit documentation to a specified project.
6. Adding New Development Tool: Tests ensure that developers are able to select and add tools to be integrated into their development environment with minimal effort.
7. Removing Development Tool: Tests ensure that developers can remove tools from their development environment with minimal effort.
8. Onboarding Feedback Collection: Tests validate the system's ability to collect and store feedback from developers to improve the onboarding process.
9. Role-Specific Onboarding Tasks: Tests ensure the onboarding process is customized to the developer's job role for efficiency and relevance
10. System Notifications for Onboarding Milestones: Tests verify that developers and administrators are notified upon achieving key onboarding milestones.

These functionalities directly address user needs identified during the requirements elicitation process and ensure our system meets the stated objectives.

The test cases ensure alignment with both functional and non-functional requirements. The functional requirements include the following:

- Automated Repository Setup (Test ID: 1)
- Centralized Access Management (Test ID: 2)
- Task Progress Tracking (Test ID: 3)
- Automated Codebase Documentation (Test ID: 4)
- Codebase Documentation Search (Test ID: 5)
- Adding New Development Tool (Test ID: 6)
- Removing Development Tool (Test ID: 7)
- Role-Specific Onboarding Tasks (Test ID: 9)

The non-functional requirements include:

- Usability: Ensuring a smooth user experience such as completing repository setup and accessing code documentation with minimal effort.
- Performance: Verifying tasks such as repository setup and documentation updates are within properly defined response time limits.
- Reliability: Tests validate that the core system features such as task tracking and documentation updates function without errors.

By covering functional and non-functional requirements, the test cases ensure that our system is feature-complete, user-friendly, and reliable.

Our test cases support an incremental design model and the model delivers functionality in phases, with each increment building on the previous one. Our test cases are structured in a manner which supports the validation of each increment independently. The first increment (Test IDs 1 and 2) validate centralized access management and repository setup. Increment 2 (Test ID 3) validates task tracking. Increment 3 (Test IDs 4 and 5) validate documentation automation and search functionality. Increment 4 (Test IDs 6 and 7) validate the ability to add and remove development tools from the already configured development environment. Increment 5 (Test IDs 8, 9, and 10) introduces feedback collection, role-specific onboarding tasks, and onboarding milestone notifications. These features focus on improving the onboarding process by tailoring tasks to developer roles, gathering insights through feedback, and ensuring developers and administrators are informed of key milestones during onboarding. By testing each increment in isolation, we ensure that the project progresses steadily, with each feature fully implemented and validated before moving onto the next.

In conclusion, the provided test cases ensure complete traceability from scope to requirements to design, confirming that the system meets its objectives in a structured manner. Thest test cases validate crucial functionalities, ensure compliance with design patterns, and align with our project's scope and requirements, all in all providing confidence in the system’s functionality and usability.

Black Box Test Plan

Test ID: 1	**Name:** Automated Repository Setup **Author:** Jett Morrow **Type:** Functional
Description	**Preconditions:** ● The developer’s profile is created in the system. ● The repository URL is stored in the system. **Steps:** ● Log in as a new developer.

	● Trigger the repository setup process. ● Verify access to the repository. **Test Inputs:** ● Developers id ● Repository url
Expected Results	The repository is cloned, permissions are set, and the developer can access it without errors.
Actual Results	

Test ID: 2	**Name:** Centralized Access Management **Author:** Jett Morrow **Type:** Functional
Description	**Preconditions:** ● The developer is added to the system. ● Project-related documents are uploaded to the system. **Steps:** ● Log in as a new developer. ● Check for document access in the portal. ● Open a document to verify access. **Test Inputs:** ● Developer id ● Document list
Expected Results	All relevant documents are accessible in the developers portal.
Actual Results	

Test ID: 3	**Name:** Task Progress Tracking **Author:** Shuban Sridhar **Type:** Functional
Description	**Preconditions:** ● The developer has at least one task assigned. ● Task tracking is enabled in the system. **Steps:** ● Log in as a developer. ● Navigate to the task tracking page. ● View task progress for onboarding. **Test Inputs:** ● Developer id ● Tasks
Expected	The task tracker displays completed, pending, and in-progress tasks

Results	accurately.
Actual Results	

Test ID: 4	**Name:** Automated Codebase Documentation **Author:** Shuban Sridhar **Type:** Functional
Description	**Preconditions:** ● The developer is working on a project that is in their integrated development environment **Steps:** ● The developer edits code and then commits it to the main branch ● The automated codebase documentation parses through the commit and updates the database to store the documentation that appears in the commit ● The documentation is organized by who made the commit and what project they are working on **Test Inputs:** ● Code file with documentation committed to repository
Expected Results	The documentation from the code file has been copied from the file into the database and can be easily found based on who committed the file and the project they are working on.
Actual Results	

Test ID: 5	**Name:** Codebase Documentation Search **Author:** Richmond Southall **Type:** Functional
Description	**Preconditions:** ● A developer has committed a file that has has documentation and the documentation has been uploaded to the database **Steps:** ● Developer enters the database ● They have to ability to search and filter the database by selecting different projects and documentation submitted by other developers **Test Inputs:** ● Developer name and project name
Expected Results	All of the documentation that has been committed by the specified developer to the specified project.
Actual Results	

Test ID: 6	**Name:** Adding New Development Tool **Author:** Richmond Southall **Type:** Functional
Description	**Preconditions:** ● The developer has their environment setup and configured **Steps:** ● The developer selects a new tool that they want to be integrated into their development environment ● The development environment is automatically reconfigured to incorporate the new tool and ties/connects the tool to other parts of the environment that it might interact with **Test Inputs:** ● New Development Tool
Expected Results	The developer's environment has been correctly configured to incorporate the new tool. They have the ability to use the tool and the tool can interact/communicate with the other tools in the environment if needed.
Actual Results	

Test ID: 7	**Name:** Removing Development Tool **Author:** Richmond Southall **Type:** Functional
Description	**Preconditions:** ● The developer has their environment setup and configured **Steps:** ● The developer selects an existing tool that they want to be removed from their development environment ● The development environment is automatically reconfigured to no longer incorporate the tool and disconnects the tool from the other parts of the environment **Test Inputs:** ● Development tool in the environment that the developer wants to remove
Expected Results	The specified tool is removed from the development environment and the other tools are aware that the tool is no longer incorporated.
Actual Results	

Test ID: 8	**Name:** Onboarding Feedback Collection **Author:** Lucas Kazem

	Type: Non-Functional
Description	**Preconditions:** ● The developer has completed at least of 50% of the onboarding tasks ● The feedback form is accessible through the system portal. **Steps:** ● Log in as a developer who has completed at least 50% of onboarding tasks. ● Navigate to the feedback section in the portal. ● Fill out the feedback form with comments on the onboarding process. ● Submit the feedback form. **Test Inputs:** ● Developer ID ● Feedback text
Expected Results	The system records the feedback. Confirmation of successful submission is displayed to the developer. Feedback is stored and accessible for review by administrators.
Actual Results	

Test ID: 9	**Name:** Role-Specific Onboarding Tasks **Author:** Lucas Kazem **Type:** Functional
Description	**Preconditions:** ● The developer's role is defined in the system. ● A task list for the specified role exists in the database. **Steps:** ● Log in as a developer ● Verify that the system loads task specific to the developer's role. ● Complete a few tasks to check the system updates task progress. **Test Inputs:** ● Developer ID ● Job role ● Role-specific task list
Expected Results	The developer sees only tasks relevant to their role. Task completion updates are reflected in real-time in the system.
Actual Results	

Test ID: 10	**Name:** System Notifications for Onboarding Milestones **Author:** Lucas Kazem **Type:** Non-Functional

Description	**Preconditions:** ● The developer is actively progressing through onboarding tasks. ● Milestone notifications are enabled in the system settings. **Steps:** ● Log in as a developer. ● Complete onboarding tasks that lead to a milestone. ● Verify the receipt of a notification for a milestone. ● Log in as an administrator and verify the corresponding milestone notification. **Test Inputs:** ● Developer ID ● Milestone
Expected Results	The developer receives a notification for the milestone achieved. The administrator is notified of the milestone in the admin dashboard.
Actual Results	