Task:
Imagine you are joining a newly built team (6 developers - 4 BE 2 FE) which has taken over an **external** developed legacy product.
There is almost no documentation of the product and only a very view unit-tests which covers only main critical functions.
The PO already planned the next 2 sprints because new high prio features need to be delivered within 6 weeks.
Deadline cannot be postponed.
The product is splitted into frontend and backend applications.
The frontend application is only used in chrome browser.
The backend application provides additional API for other connected services.

1. Review existing documents. The priority goal should be to gather all the scattered documents, Emails, Recordings, Defects and connect the dots.
2. Review existing Unit tests, they should help as a starting point for documenting tests.
3. Prepare business process flow documents for the existing business functions
4. Check for the existing defects, and derive the complexities and risk areas in the existing system
5. Prepare presentations with business flow charts to help others understand the system
6. Understand the impact of new enhancements and derive the best fit within existing functionalities
7. Maintain a library of Knowledge Transfer sessions and functional queries to reduce the iterative work
8. Identify the go-to person for each module within the application and make sure that all your scenarios are thoroughly reviewed
9. List all dependencies and make sure every dependency should be planned to cover in the Regression Testing. (The backend application provides additional API for other connected services)
10. 1-2 BE developers should start working on unit test coverage for existing legacy code(as we have 4 BE developers, part is working on legacy test coverage and part is working on new feature/bug fixes)
11. 1 FE developer should start working on unit test coverage for existing legacy code(as we have 2 FE and only one browser to cover)
12. Static test analyzers could help (SonarQube, FindBugs, PMD, checkstyle)
13. Need to create a priority list of components for which testing makes the most sense. It’s important to map out the various components according to their number of dependencies, their amount of logic, and the project priority.
14. For developers(who work on legacy code): In a separate source tree build regression or 'smoke' tests, using a tool to generate them, which might get close to 80% coverage.
15. For each bug fix, or feature we add from now on, use a TDD approach to ensure new code is designed to be testable and place these tests in a normal test source tree.

16. Existing code will also likely need to be changed, or refactored to make it testable as part of adding new features.
17. When making changes (bug fixes or features) via TDD, when complete it's likely the companion smoke test is failing. Verify the failures are as expected due to the changes made and remove the less readable smoke test, as your handwritten unit test has full coverage of that improved component. Ensure that test coverage does not decline, only stay the same or increase.
18. When fixing bugs write a failing unit test that exposes the bug first.