**TestManagement Exercise**

**Adidas**

**QUESTION BLOCK 1.**

* 1. **Essential points of test strategy.**
* Understanding of scope and objectives.

*\*In Scope:* Functional and non-functional tests. Unit, integration, system testing. Load & spike testing for performance. Start for automation with unit tests to CI/CD. Regression testing. Negative testing. API testing (throttling and burst).

*\*Out of Scope:* Authentication services, testing internal services which are already developed for pricing and media. Logging and reporting.

*\*Assumptions:* Database tests for Amazon RDS already done. Node and react friendly frameworks used for development.

* Defining testing metrices like coverage more than 95% required. Testing goals and KPIs to be defined related to quality testing.
* Choosing the proper methodology. Agile suits best here because of tight deadline.
* Primary skillset of the resources. Knowledge on NodeJS, React, angular, SQL, AWS, automation required.
* Proper risk analysis and how to mitigate them.
* Extra buffer time for unseen roadblocks and issues.
* Tools adopted for testing, automation, and test management. AWS provide a lot of security and inbuilt features where separate developer, and tester group can be made.
* CI/CD implementation. QA won’t need source codes if CI/CD used.
* Roles and responsibilities for the team members.
* Knowledge of retailer and reseller with the technology. Trainings will be provided if not familiar with the technology.
* Automation strategy and tool to be adopted.
* Sensitive data management and usage in testing.
* Proper risk analysis and their mitigation ways.
* Security Testing and penetration testing teams available from start. Cloud based platform and API used. Proper authentication and authorization needed all the time.
* Understanding of quantum of requirements.
* Product catalog size and detailing. How many orders possible, do they have connection with house management system.
* Configuration and change management.
* Environment creation and duplication.
* Test data management and quality.
* Issues tracking and reporting. Test sets for unit tests should be displayed or provided.
* Timelines & Deliverables. Code should be freeze at least 2 weeks before the final deadline of the project and last 2 weeks only for UAT.

**1.2 Main challenges for testing**

* Doing exhaustive and non-automated tests is difficult in building phase because a lot of microservices are involved. Many services used are better understood characteristically when deployed in production and used for long time. Can be tricky to fix then.
* Timelines are challenging.
* Data and variation of data. Data for right components.
* Estimation of testing is hard because of timelines.
* If complexity arises then coverage can increase.
* Functionality changes or modification of user story.
* Unseen changes.
* Environment change.
* Many setup related, captcha, high complexity cases which are not automatable.
* Plans for training end user if not friendly with the technology used.
* Undefined quality standards (no clarity on testing requirements, specifications, guidelines, or characteristics),
* Lack of communication (Development and testing teams must collaborate at regular intervals in the presence of the product head or managers to be on same page).
* Insufficient requirement gathering(leads to inadequate testing).
* Missing document (teams must create comprehensive documentation with specific project requirements in detail. Complete documentation will serve as a baseline for QA teams throughout the project lifecycle).

**1.3 Approach to test automation strategy**

Automation strategy is critical in the project. Automation Feasibility will be checked from the start to identify scope of automation and approach to the strategy will be by considering three major aspects:

Process, Technology and Roles.

More detailed approach will include:

* Defining high business value tests which has critical functionalities.
* Defining repetitive, similar tasks to introduce code reusability, scalability. These are more error prone to do manually.
* Tests requiring multiple data sets.
* Defining roles and responsibilities for the members in automation.
* Capturing results and generating reports to help in decision making related to the software.
* When multiple platforms are involved.
* Identifying risks (to define the priority and business impact it makes)
* Understanding technology, tools & resources. Selection of proper tool is very important.
* Usage of multiple data sets for one requirement. Making sure data is good.
* Define Continuous integration and deployment.
* Considering testing environment.

**1.4 Integration of automation to CI/CD**

Automation will be at core for this project being the central to development process and will go hand in hand from unit test. Right set of unit tests with right techniques will be provided to the CI/CD pipeline. Each time new code is written the test sets will be provided to the testing server in CI/CD. Different developers with different pipelines can push their codes with their test sets and hence when the overall build will be ready test sets also will be ready with all cases from different pipelines.

**QUESTION BLOCK 2.**

**2.1 Recommended methodology for Project development**

Agile development methodology is recommended for this project.

Scrum implementation of agile methodology is suitable.

**2.2 Why?**

* As project is about to start and considering the timelines given there is no room for waterfall approach. It needs fast paced development and testing.
* Scrum provides more customer value because they can verify each requirement or functionality once done.
* Project will get a better time management because each sprint will be planned and divided with functionality and user stories.
* Risk reduction will be more because we will carry out critical and important functionalities at the start.
* Scope related to testing can be modified anytime.
* Roles and responsibilities will be clear to each person involved.
* Daily scrum and retrospective meetings will help in avoiding mistakes or bad practices and improve.
* Flexibility will be provided for a quick change or modification if required.
* Scalability will increase over time with changing requirements, data sets and addition of functionalities.
* Automation is of core importance for this project, team can start automating unit tests from initial sprint and will deploy on testing server.
* We will identify critical functionalities and develop them first.
* Continuous communication is required from client for better requirement gathering and getting signoffs for sprints.
* Burndown chart will help to check the velocity and push more if required.
* Involvement of all stakeholders are important and required for quick feedback and changes.
* CI/CD integration is done and with each sprint it will be easy to integrate and deploy.
* Client review and approval is important for all deliverables for maximum accuracy.
* In case of defects/bugs cost of fixing will be less.
* Easy to do regression and retest.
* Data sets and environment duplication is important which can be verified by client.
* Real time user data can be taken from client for testing.
* Development and testing need to go hand in hand to mitigate risks and improve collaboration and communication.
* Team members are cross trained with each other’s knowledge and experience because they all sit together and work as one scrum team.

**QUESTION BLOCK 3.**

**3.1 Configuring test team**

QA team will be a mix and match of 3-4 resources along with test lead. As team members get cross trained within scrum. Members with knowledge in Node, react, APIs, DB will be preferred.

Security testing team will be deployed from the start because of cloud architecture used.

Dynamic team members will be used in all SIT, UAT (if UAT is done in house).

To form a strong test team, a good rapport amongst the members will be focused. To achieve this, some interactive sessions will be carried out where the team will interact with each other or with new team members regarding defects or automation tools, or project-related discussions, or brainstorming valuable innovative ideas.

**3.2 Essential skills required**

* Knowledge base is important. Primary skillset of QA with knowledge and skills in technologies used in project is must. For ex. Node, React, Api, Database, AWS etc.
* Learning curve of the QA member should be high to learn quickly and implement if needed.
* Eye to detailing is important, so that nothing is missed, and test coverage and accuracy will increase.
* Adaptive nature is preferred because there will change in datasets, scenarios, approaches which will lead to changes in project.
* Interpersonal skills are important to work better within a team.
* Individual contributor is highly preferred who is motivated and can take ownership of their own work and drive it.
* Good Retail understanding will be key.
* Knowledge of automation tools and framework. Respective plugins and integrations for reporting, logs etc.
* Communication skills are important to effectively get along with stakeholders.

**3.3 Mindset and attitude required**

* Self-oriented and motivated will keep the individual and team boosted which will help in better productivity.
* Accountable attitude is preferred which will help in delivering on time and taking responsibility of the given work.
* Quick learner to learn required new things and adapt to them.
* Good Team player. It will help in achieving the overall team’s goal better.
* Being Proactive will help to analyze the roadblocks in work and efficiency can be increased.

**QUESTION BLOCK 4.**

**4.1 Identifying Test scenario and test cases.**

Project will be in TDD which is a core functionality in agile practice, so our development will be test driven.

Initially scenarios will be defined based upon the technologies being used. Further moving ahead based upon the user stories test cases will be formed specific to each user story. Purpose is to ensure complete test coverage.

* End-to-end testing scenarios, optimal testing, integration point scenarios all these are critical in designing scenarios and test cases.
* Scenario for API and test cases related to basic functionalities like Get, Put, Post, Delete, Patch etc which is a general practice while working with APIs.
* Particular functionalities will be separated, and their scenarios will be made.
* Each scenario will be tied to minimum of one functionality/ user story.
* Scenario for performance testing like testing for maximum number of user (200 in this case), penetration testing.
* Scenarios for database connections and authentication.
* Scenarios related to security and authentications and authorization.
* Any scenario having multiple requirements at once is to be tested separately for all requirements individually.
* Testing metrices defined for maximum coverage.
* Identifying scenarios for automation.

**4.2 Keeping traceability between requirement and scenario.**

Requirement Traceability Matrix will help to map each user story to the test case.

Bi-Directional Traceability matrix will be implemented which works forward and backword both. By this way, each user stories will be mapped to their test cases and similarly each test cases will be mapped to their respective user stories.

RTM helps in achieving more valid and accurate test cases to each use cases.

**4.3 Splitting Testing Responsibility between developers and testers.**

*Developer’s part:*

Unit testing, which is very important, Basic functionality at the time of developing, code reviews among the team to find errors and having quality, static code analysis to find weaknesses in code and security testing part are among some important testing responsibility given to developer.

Automation will start from unit test cases as soon as develop will develop it can be deployed to test servers in CI/CD and automation can be achieved gradually.

*Tester’s part:*

Integration level testing, system testing, end to end functionality.

SIT related all kinds of testing.

UAT is coordinated effort. If client provide UAT data or asks for in house UAT.

Performance related testing.

**5.1 Tool for test management**

*JIRA:* due to security reasons. Highly configurable and flexible for different environments. Issue tracking and bug reporting is very well managed. Workflows and different kind of customized reporting available. RTM implemented.

**5.2 Tools for performance testing**

*LoadRunner*: very good for determining root cause issues related to performance. Allow performance testing for existing legacy applications also.

*JMeter:* open source for load and performance testing. Can be used in command line interface. Support per thread cookies.

**5.3 API testing & automation**

Rest Assured best suited for java domain specific language. It does not need expertise in HTTP to test web APIs. It has lot of inbuilt functions to test. Also supports BDD.

Postman offers both web and desktop version. GUI based more user friendly. Easy to use, easy to call methods, easy to set parameters. Has ability to store commands.

**5.4 FE testing and automation**

*Selenium:* is preferred. Open source, compatible with multiple OS and browser. Can be written in different languages. Integrated with large number of tools for different purposes.

**5.5 Other tools**

*Cucumber:* BDD (Behavior Driven Development)

*Log4j:* log analysis.

*Elastic search and Kibana:* reporting.

*Confluence(integrate with JIRA) word/office:* Documentation & Strategy

*Microsoft Teams:* Meetings and Communication

*Google Chrome:* preferred for browser

*Sharepoint:* Repository document if required.

*Jenkin:* CI/CD

*Github*: Repository code

*Eclipse/IntelliJ:* Writing codes

*TestNG/jUnit:* Frameworks for automation and reporting