Sprint Test Plan

Where to start, what to include and how to keep it straightforward

By Heitor Augusto Mecelis

Sprint Test Plan

Before we head to the template itself, first let's talk about what is necessary on a Test Plan and how we want to organize this information on the file. Let's begin organizing our train of thought:

- **The Scope: User Stories, Tickets, Defects**
  What are the user stories that we are going to test through this sprint? Are there any defects that we are planning to have fixed in this period? Is there a work order, or any sort of ticket we should work on that will need tests? Well, this is where it's going to be located.

- **Test Environments: Devices, Versions and Browsers**
  Here is where we are going to keep track of where we intend to have the stories mentioned on the scope tested. Specify versions - Windows 11 - Firefox 22.3.2, Mac OS X Monterrey - Safari 13.5.4, etc. Same goes if any mobile tests will be done: Samsung A35 with Android 12.5, iPhone 13 with iOS 17.2.4, Bluestacks Emulator with Android 13. It's always important to keep track of versions as the behavior of the applications can be different either on purpose or by accident. And, depending on the application tested, the database type and version may need to be mentioned as well (sometimes more than one type of database is used, then more than one has to be tested).

- **Estimated Test Time: How long will the tests take**
  Yes, it's obvious that the tests will take the whole sprint, the planning is about that, though, why not estimate how long it's going to take to finish testing on each of the stories/ tickets/ defects so, if anything is not ready with enough time to be tested, we already have it registered to avoid complications? Example: User Story US-12345 - ETT = 3 days or 24 hours. Also, remember to include in your time count the time taken to set up the environment so there's enough time to do all the testing.

- **Test Scenarios: Structured test cases, Exploratory approach and more**
  Even if you want to create test cases while the developers are working on the stories, it's always good to have at least a placeholder list and your approach mentioned, such as an estimation of the positive and negative scenarios you may want to use to test those changes. Remember: the test cases are born from the scenarios, which is really helpful on a review where the developer shows what they have done and the QAs have the opportunity to ask questions and draw the

test cases or at least confirm if the scenarios estimated are correct or if new ones are needed, if any has to be deleted and so on.

- **Test Types: Types of testing necessary through the sprint and tickets, their levels**
  It doesn't matter if we're talking about Manual or Automated testing, when we are planning the sprint tests, we have to foresee at least part of which techniques and technologies we will need to use, for example, if we are up to testing an API response, which approach will be taken? Is there a Swagger option? Will it be better to use Postman? And inside this part, understand if there's a library or if one needs to be created.

All of the details above are pretty necessary. Besides that, we cannot forget to consider the Risks, References, Requirements and Variables.

- Risks: when thinking about everything that can go wrong, a QA should also think of solutions for those possible issues. What if the developers don't deliver the code on time to get tested? Then we have to register that and discuss with the team the possibility of getting it tested early during the next sprint before new stories get ready for testing. What if the testing environment has errors and cannot be used? Then we have to prepare a new one and embrace the fact that sometimes environments don't work and we need to build new ones. What if one of the QAs is out of office unexpectedly? Then we should distribute the remaining work between the remaining QAs and understand that this sort of thing can happen to anyone. What if critical bugs appear and make the stories take double the time estimated to them? Embrace that delay, discuss with the team because it's better to have a critical bug during a sprint than having it found by an end user.

- References: which references do we have to make sure the new code is behaving as expected? Acceptance criteria should be built along with the Product Owner and developers, if there's UI work done, Figma or any image reference and have the business rules in mind is always good, and if we have links to this sort of information, that's even better.

- Requirements: what do we need to begin testing? Developers should deliver the code, DevOps should control the environments, testers should have the licenses to the needed programs or access to other solutions in order to perform their jobs, such as Jenkins, TeamCity, IntelliJ Idea, AWS, all of the tools that are necessary to make sure a QA can do the QA job.

- Variables: I know unexpected days off are part of the risks, but sometimes it has been previously discussed, so, team member vacations that either start or end by anytime during the sprint should be taken into consideration during the sprint test plan, to make sure all the necessary work gets done and nobody gets overworked.

Now that we have been through all of that needs to be on a test plan, let's build one!

In the below example we are going to be really generic, so, if necessary, adapt to what fits better your team's needs. A good QA doesn't have to stick to structures when these structures are no match to their real life situation.

Sprint 25 - Test Plan

Scope:

US-1234 - Change Send button to Icon
US-1235 - Update colors on headers according to new UI reference file
DEF-0324 - 404 error when opening old messages

Environments:

Windows 10 and 11:
Chrome - 12.5 (or above)
Opera GX - 10.2 (or above)
Firefox - 35.6.3 (or above)

Database:
MySQL 8
MSSQL 2019

Estimated Test Time:

US-1234 - 8 hours / 1 day or 2
US-1235 - 40 hours / 1 week or 2
DEF-0324 - 16 hours / 2 days or 3

Test Types:

US-1234 - Visual comparison, Functional testing of the button
US-1235 - Visual comparison with color pick between reference and result
DEF-0324 - API check on Postman, UI exploratory around messages, force reproduction on fix version

Test Scenarios:

US-1234 - Change Send button to Icon
- Scenario 1: Icon is successfully implemented on button
- Scenario 2: Icon fails on button but button works
- Scenario 3: Button is not usable with icon

US-1235 - Update colors on headers according to new UI reference file
- Scenario 1: Colors are successfully updated
- Scenario 2: Colors differ from a browser to another
- Scenario 3: Colors differ from an OS to another

DEF-0324 - 404 error when opening old messages
- Scenario 1: User can see old messages
- Scenario 2: User can't accidentally click on old messages
- Scenario 3: User can delete old messages

Risks:

Environment won't start - Get a backup ready beforehand
Sick QA - get all tests documented at all moments of the sprint

References:

Figma link
API libs link

Requirements:

Developer work done with a day + ETT before end of sprint, AWS access to all QAs, Figma access to QAs, API docs access to QAs.

Variables:

No PTO at this period.