Bug Reporting

Justal Kevin

1 May 2022

Contents

Part I

How to report a bug

Preface

When reporting a bug, there is few points required for easily understand and rapidly been able to fix the problem. The following points are the minimum for a Jira ticket or a bug report to be handle :

1. **Description**
2. **Screenshots**
3. **Replication**

You will find under more explanation about each sections with some examples of what should find in each part.

Feature Information

In the description, the bug should be describe as simple as possible for anyone to be able to understand it. It needs to be exhaustive, the more information the better, the more information the easier it is to understand the problem. The bug should not be only understandable by the one handling the ticket/bug. The reason is simple, what happen if the reporter and the one handling the ticket are not in position for answering (Holiday, hospital...). In this case, the ticket need to be handle by someone else and he will need to understand the problem.

Screenshots

A description should always been accompagned with a screenshoot. Text alone is often not enought for pin pointing where is the problem. Human are visual creature, so a text plus an image made any bug report clearer. It also limits the scope of the error to what the screenshot will show.

- **Deadline/Due Date :** Provided by the Business, this limit give information to the TL/DEV of the time allowed for the new feature. It should be written in full letter for avoiding confusion between different way of writting the date.
  For example : 12 September 2023.
- **Business need summary :** Provided by the Business, this row should indicate in one or two sentances the general goal.
  For example : I want the users to be able to pay.
- **Priority :** Provided by the PO/PM, this row should indicate how important is the feature on a range from 0 to 10.
  For example : 10 (very important) - 0 (optionnal).
- **Jira :** Provided by the TL, this row should contains the link poiting to the Jira story related to this feature.
- **Reporter :** The person who is requesting the feature. This information make it easy to know who to ask information to.
- **Wording :** Provided by the Business/PO/PM, For every feature, a document should contains the sentances for the feature. If connection with Phrase, the link to the Phrase need to be provided.

For example :

Deadline	12 September 2023
Business need summary	I want the users to be able to pay
Priority	7
Jira	http://www.example.com
Reporter	Kevin Justal
Wording	http://www.drive.com

0.1 Scope

This part fill up by the PO/PM should indicate what inside this feature is the minimum required (MVP), the nice to have and the part outside of the scope.

For example :

MVP	The users need to be able to pay with a credit card 3D Secure
Nice to have (optionnal)	The users can pay with paypal
Outside of scope	The users can pay with bitcoin

0.2 Problem Statement

This section will give the necessaries information for understanding the functionnal requirements or modification of the UI. Any person should be able to graps the idea of the feature and the improvement just by reading this section.

Actual state of the system

In the case of a refactoring or improvement of a feature already present in our system, PO/PM/PM should describe the part that gonna be improve. This section highlight the actual way the system is working or should be working and put everybody on the same page. Dont hesitate to put image, graph or any document that could be interesting.

For example :

The users can actually register to the app by entering an username, a password and a confirmation of password. After registration, they become automatically redirected to the platform where they can fully use the system.

Problem/Improvement to do

In this section, PO/PM/PM/Business should described what is the problem or the improvement to make on the actual system described in the previous section.

For example :

With this system, the new users are not customer. We would like to make the new users pay to be able to access the system.

Final state of the system

In this section, PO/PM/PM/Business would described the result needed. With the previous sections, it will become easy to compare the actual and final state of the system by highlighting the differences.

For example :

The new users registering to the app will now have a new screen after registration and before becoming allowed on the system and redirected. In this system, they would be requested to pay an amount of 29$ by entering they credit card information. See the mockup in the other section. Once the paiement is accepted, they will be automatically redirected to the system.
In case, they registered and quit the page before paying, the next time they connect to the app, they will restart from the paiement page. They wont be accepted on the system till paiement has been made.
In case of trouble for paying, a button will be added for requesting assitance by chat to our customer service.

Additionnal notes

- **Algorithms :** Provided by the Business/PO/PM/DT, if any algorithm is needed for the feature. It should include the explained algorithm with a minimum of one example for easy understanding.

  For example :

  I want the system to count the number of occurence for each tags.

  Say you have a list of integers representing the tags from 0 to 5:

  input = [2, 5, 3, 1, 4, 2]

  First, you need to create a list of counts for each unique value in the ‘input‘ list. Since you know the range of the ‘input‘ is from 0 to 5, you can create a list with five placeholders for the values 0 to 5, respectively:

  count = [0, 0, 0, 0, 0, 0] # val: 0 1 2 3 4 5

  Then, you go through the input list and iterate the index for each value by one.

  For example, the first value in the ‘input‘ list is 2, so you add one to the value at the second index of the ‘count‘ list, which represents the value 2:

  count = [0, 0, 1, 0, 0, 0] # val: 0 1 2 3 4 5

  The next value in the ‘input‘ list is 5, so you add one to the value at the last index of the ‘count‘ list, which represents the value 5:

  count = [0, 0, 1, 0, 0, 1] # val: 0 1 2 3 4 5

  Continue until you have the total count for each value in the ‘input‘ list:

  count = [0, 1, 2, 1, 1, 1] # val: 0 1 2 3 4 5

  Finally, since you know how many times each value in the ‘input‘ list appears, you can easily create a sorted ‘output‘ list. Loop through the ‘count‘ list, and for each count, add the corresponding value (0 - 5) to the ‘output‘ array that many times.

For example, there were no 0's in the 'input' list, but there was one occurrence of the value 1, so you add that value to the 'output' array one time:

output = [1]

Then there were two occurrences of the value 2, so you add those to the 'output' list:

output = [1, 2, 2]

And so on until you have the final sorted 'output' list:

output = [1, 2, 2, 3, 4, 5]

1 Limitation and impacts

1.1 Product consideration

In this section, PM/PO will described the resulting effect of this new features with the actual system. The part of the system that will be impacted by such new feature.

For example :

If the goal of this feature was to remove a plan. The PM/PO should let everybody know that the current users using this deleted plan will be impacted and tell the TL and DEV what to do about those users.

1.2 Technical consideration

In this section, TL will described the limitation and the effect on other parts of the system of the new features.

For example :

If the goal of this feature was to remove a plan. The TL will let the PO/Business know that such feature is impossible without managing the user under this plan. Stripe does not allow that.

2 Additionnal information

2.1 Wireframe

Depending of the feature, a front UI might be needed. In this case, FT should provided wireframes before the implementation for creating endpoint close of UI. This requirement helps the frontend and backend connect easily their implementation.

2.2 Milestones/KPI

In this section, the PO/PM/DEV/TL will describe the milestones that the developer of the feature has to achieve to validate the KPI of the entire feature. By exploding the feature into milestones, it becomes easy to see the agile step of the development and what part could be split into multiple developers.

For example :

12 September 2023	The users can register a card
23 September 2023	The information of the user got transmitted to stripe
30 September 2023	The users can pay with the card

2.3 Validation Process

In this section, Business/PO/PM/QA should provide the explicit steps that will be executed for testing the features. Items does not need to be enumerated and can also does not have any relation between each others.

For example :

1. Make sure on first login the prompt is shown.
2. Do not accept, logout and login and make sure it is still shown.
3. Try to click the accept button without checking the box.
4. Check the box and click the accept button now.
5. Logout and make sure the prompt is not shown, make sure in DB the version of the last accepted version was saved.
6. Upgrade the version of the text, make sure the prompt is now shown again.

2.4 Deployment

In this section reserved for the DEV, you will find any information relative to the deployment of the feature on production such as the environment variables to add in the .env file of the production server, any command to run on the production server when the feature has to be merged...

For example :

- Add in the .env file: KEY=z4e54az65e46az4e6
- Run the command: pm2 reload all

2.5 Additionnal document

In this section, anyone can add document or write additionnal information that can be usefull for understanding the feature.

Glossary

Business is any person from the sales teams.

DEV is any person responsible for the development of the features.

DT is the data team.

FT is the frontend team.

PM is a Product Manager.

PO Is a Product Owner.

QA is any person from the quality assurance teams.

TL is the team leader of the backend or frontend team.