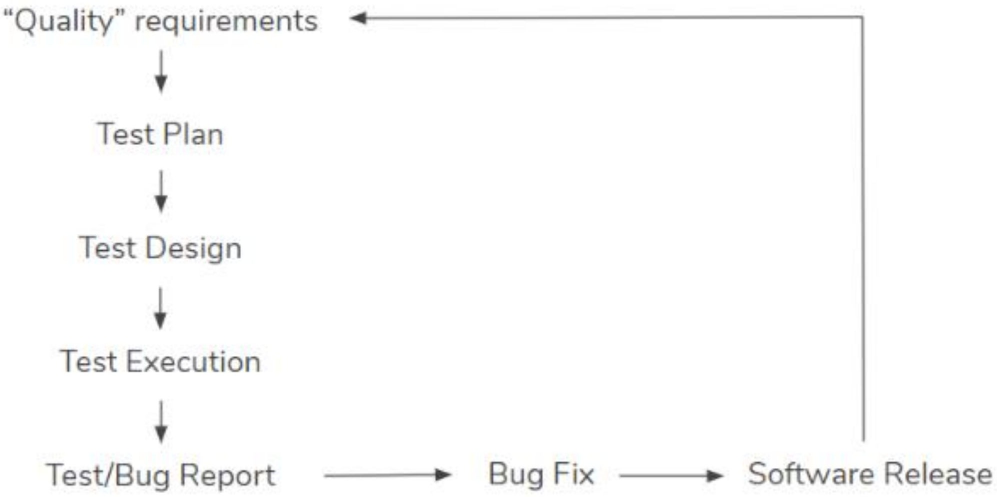
MISIONEROS
STUDIOS

Quality Assurance Plan

PROCESS FOR QUALITY TESTING:

Misionero’s studios will use a cyclic process to test and control the quality of every feature and development of the game Mythologic Parade, this will consist of the following stages:



1. **“Quality” requirements:**

For every task done in our team we have our own system of delivery to ensure all tasks and new features do not break the game and that way avoid further problems.

The way to accomplish this is using the same protocol for every task or feature done before its uploaded to github an added definitively in the game, the member who did the task will have to tag it with a specific state to let know the rest of the team that this task is waiting for approval and it will be revised by specific members of the team to be accepted and finally added in game.

It could seem a slow process but it will speed up the process when working in small teams for every part of the game cause they will ensure their own task does not crush the game and the other teams will focus in their tasks only and at the same time we will save time by avoiding a lot of errors.

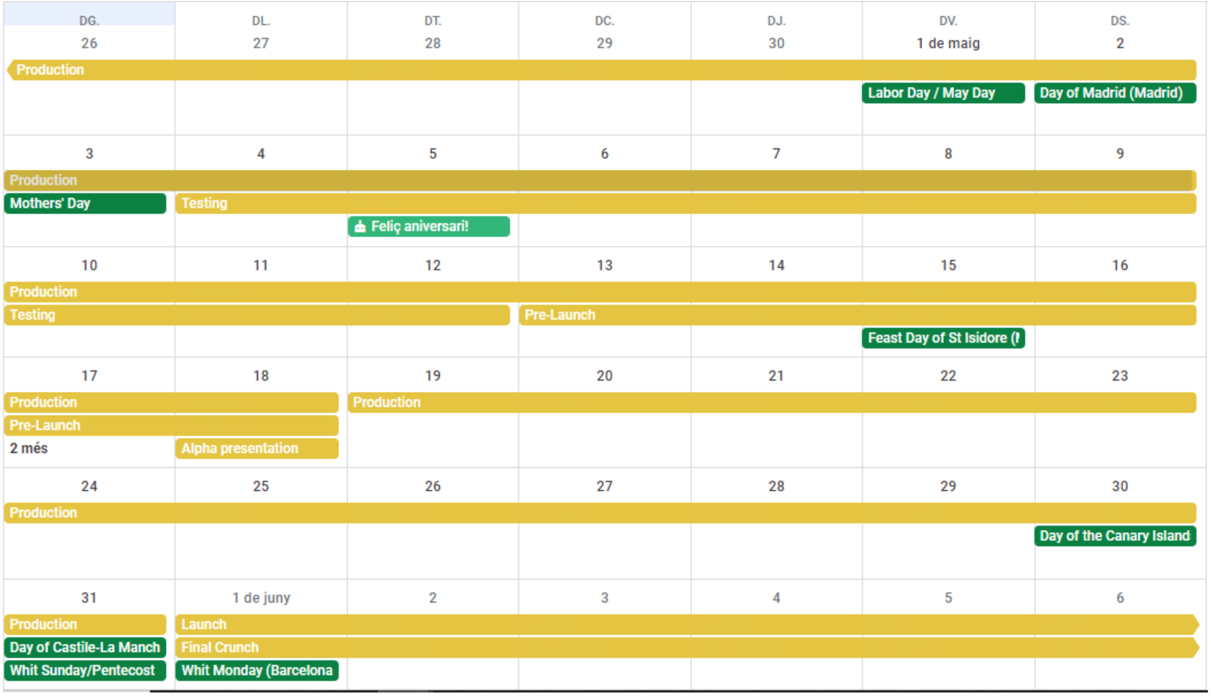
This quality requirements are explained in the following section: *Milestone Delivery Protocol.*

2. Test Plan:

The plan created to testing the game will try to give us the more time as possible to develop the game and at the same time correct the possibles errors and bugs. So this workflow plan is defined the same for all the team to all be conscious of the progress of the game and the others teams work.

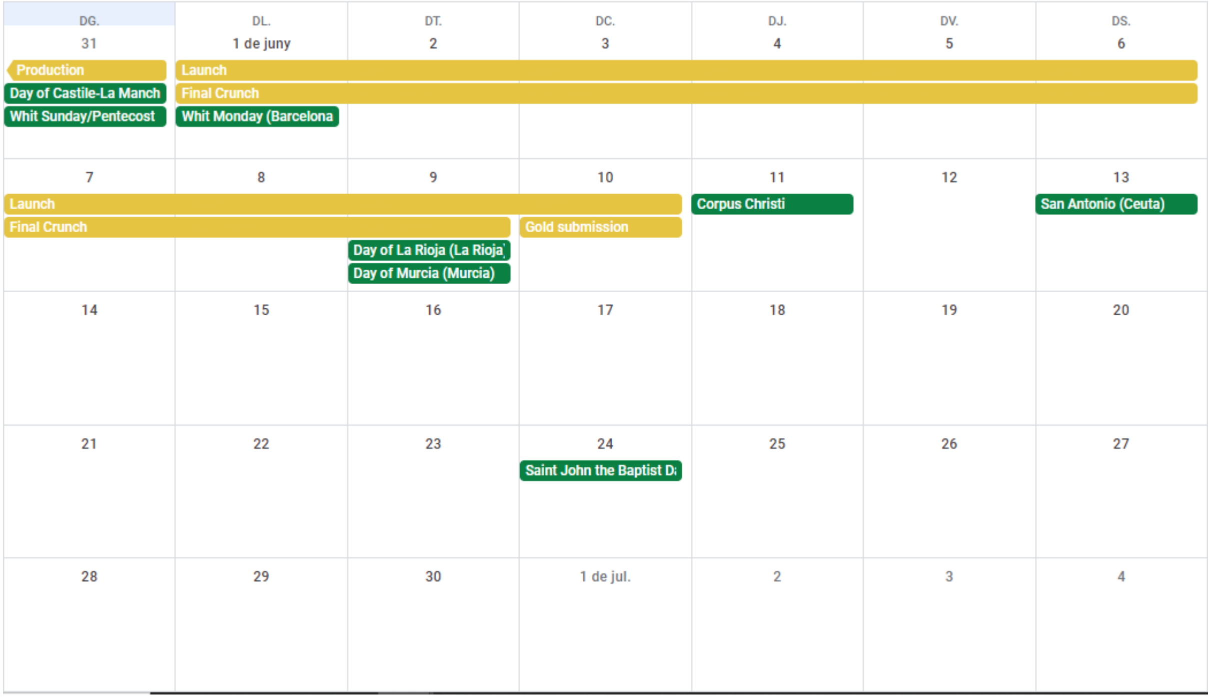
This test plan is shown in the following images:

May 2020



DG.	DL.	DT.	DC.	DJ.	DV.	DS.
26 Production	27	28	29	30	1 de maig Labor Day / May Day	2 Day of Madrid (Madrid)
3 Production Mothers' Day	4 Testing	5 🎂 Feliç aniversari!	6	7	8	9
10 Production Testing	11	12	13 Pre-Launch	14	15 Feast Day of St Isidore (I	16
17 Production Pre-Launch 2 més	18 Alpha presentation	19 Production	20	21	22	23
24 Production	25	26	27	28	29	30 Day of the Canary Island
31 Production Day of Castile-La Manch Whit Sunday/Pentecost	1 de juny Launch Final Crunch Whit Monday (Barcelona	2	3	4	5	6

June 2020



DG.	DL.	DT.	DC.	DJ.	DV.	DS.
31 Production Day of Castile-La Manch Whit Sunday/Pentecost	1 de juny Launch Final Crunch Whit Monday (Barcelona	2	3	4	5	6
7 Launch Final Crunch	8	9 Day of La Rioja (La Rioja) Day of Murcia (Murcia)	10 Gold submission	11 Corpus Christi	12	13 San Antonio (Ceuta)
14	15	16	17	18	19	20
21	22	23	24 Saint John the Baptist D:	25	26	27
28	29	30	1 de jul.	2	3	4

3. Test Design:

For the game developed by Misionero's studios will use a mix of QA Department testing and UX testing in two differents stages of the Quality Testing process.
We will focus on automated tests like Stress tests and Unit tests for code and design tasks mostly. The following stages are some we will be testing:

- Economy of the game balance.
- Displacement of units in game.
- Performance of the game code.
- Memory leaks.
- Mechanics and dynamics errors or bugs.
- IA performance.

Then the manual tests will be focused for art, UI and user experience tasks, tests like Interface tests, Hurístic tests, Cognitive tour tests, Mechanical tests, Sound tests, Smoke tests, Stress tests and Smoke tests.The following stages are some we will be testing:

- The game art looks complete and non-bothering.
- Sounds are correctly delivered to the user and does not desconcentrate the user.
- The UI is understanding and it has the most possible accessibility for the users.
- The aesthetics of the assets in game correspond it use.
- The gameplay presents a learning curve, tutorial or difficulty progression process.

This are some of the tests we will be doing and some features we already know that will need testing, although there are a lot of mechanics, tasks and features that will require a testing in the future of the game and these will be added at these list while their development.

For these tests we are going to use heurístics reports, cognitive tours documents, bug reports, delivery methods files and some software like visual studio, github and tiled. We are using a proactive and based on risks mix, where we're trying to prevent errors with the delivery protocol and the correction of these bugs with the bugs report file. We have in mind to the deficit of automated testing tools so we have designed this previous documents.

4. Test Execution:

Although we we'll keep a constant process for testing the new features included during the development of the game with the Milestone Delivery Protocol we will have some time to implement the previous explained tests techniques. During this sections of time dedicated at testing the game will be the QA team who leads the process and will be divided in two specific plans of test: the Alpha and the Beta.

This will be the principals pillars of Alpha and Beta testing on Misionero's Studios:

- Alpha testing will clear out most of the major bugs in the system while beta testing will try more like polishing the game.
- Alpha testing is won't open to the public while beta testing could involve the public.

This to releases will be intended to test the game in differents ways before the gold submission, you can see the Alpha and the Beta processes explained more detailed in the following section: *Alpha and Beta testing*.

5. Test/Bug report:

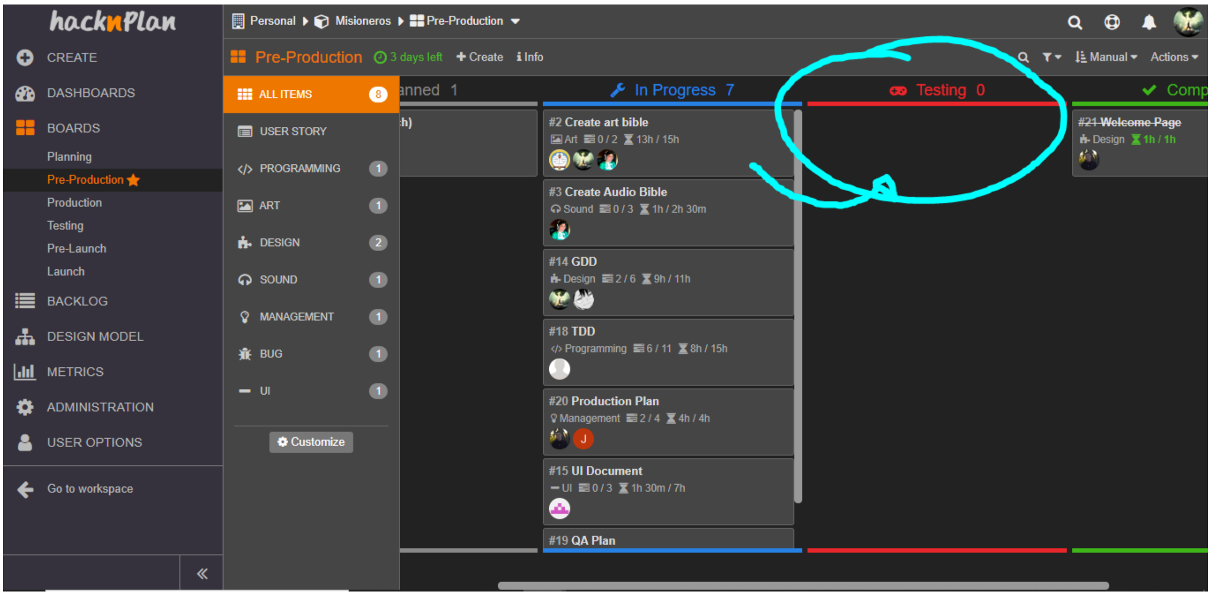
We have created an specific document to report possible bugs that we find in game during the testing processes. This document contains the following features: reviewer information, bug information, bug description, administrative information and added notes.

The Bug Report will ensure all bugs are well descriptive and avoid chaos at the time to correct them, this way will know rapidly who found the bug, where is the bug, what's about the bug and which team will be the one who is in charge to correct it.

This bug report is explained in the following section: *Bug properties and workflow.*

MILESTONE DELIVERY PROTOCOL

Once a determined task is completed the developer will must inform of it via the TESTING board in HacknPlan, there it will be needed the requirements established by the QA team and each leads of other areas to be approved.



#2 Create art bible
Art 0 / 2 13h / 15h

#3 Create Audio Bible
Sound 0 / 3 1h / 2h 30m

#14 GDD
Design 2 / 6 9h / 11h

#18 TDD
Programming 6 / 11 8h / 15h

#20 Production Plan
Management 2 / 4 4h / 4h

#15 UI Document
UI 0 / 3 1h 30m / 7h

#19 QA Plan

~~#21 Welcome Page~~
Design 1h / 1h

Design task requirements:

Requirement	Performs/Doesn't performs
It maintains a cohesive performance with the actual design of the game.	
It doesn’t requires a large new system in-game.	
Doesn’t breaks any other system implemented before.	
Users will realize (understand) that the action is linked to the active object, that is, that the object serves to execute the action you want to execute.	

Art task requirements:

Requirement	Performs/Doesn't performs
The object/unit respects the color palette (saturation, brightness and tone).	
It has the right size respect the others tiles/assets/props.	
It has been modified with the photoshop’s “Brush	

Strokes” effect.	
It does not breaks the immersion of the player in-game.	
The objects look like what they are for (affordance).	

Code requirements:

Requirement	Performs/Doesn't performs
It compiles without any relevant error.	
No functions definitions on .h files, definitions MUST be on .cpp files.	
We MUST comment functions in the .h file, at least explain in a line what the function does and what the parameters are.	
The names of functions parameters MUST clearly state what they are.	
Variables MUST be identified with a clear/related name and/or a comment.	
When declaring a class, the code inside must follow this rules, constructor and destructor declarations, variable declarations, function declarations.	

UI requirements:

Requirement	Performs/Doesn't performs
The new interface is consistent in control, colour, typography and dialog design (e.g. large blocks of text are avoided, no abbreviations) and as non-intrusive as possible.	
The added object is intuitive and the meanings are obvious and perceived as a part of the game.	
The visual representation (i.e. the view) allows the user to have a clear, unobstructed view of the area and of all visual information that is tied to the location.	
If relevant information is displayed and if critical information stands out. Irrelevant information is left out.	
Users will realize what they can do in the interface, in	

relation to the objective they want to achieve in this step.	
Users will see the active object of the interface on which they have to act (click, write text ...), to achieve the objective of this step.	

ALPHA AND BETA TESTING

- **Alpha:**

This will be the first full interactable and playable experience of the game, it will contain new features that will be tested.

The core module/mechanic is somewhat usable. A rudimentary basis that can be used only via command line or scripts. Important: the features implemented in the Alpha won’t be locked-up, they will be predisposed to possibles changes in future.
The Alpha should have the necessary assets to make playable the game, it doesn’t need aesthetics props or assets that doesn't have functionalities involved.

During the Alpha release it will be submitted to an internal process of testing, members of the group will be who play and test the game to find dysfunctional functionalities, bugs or possibles errors. This members will try catch majority of the problems by putting the software through all scenarios they can make and try any combination of inputs to coax the software into an error. If a feature fails alpha testing, changes are done and it repeats the tests until the feature passes.

- **Beta:**

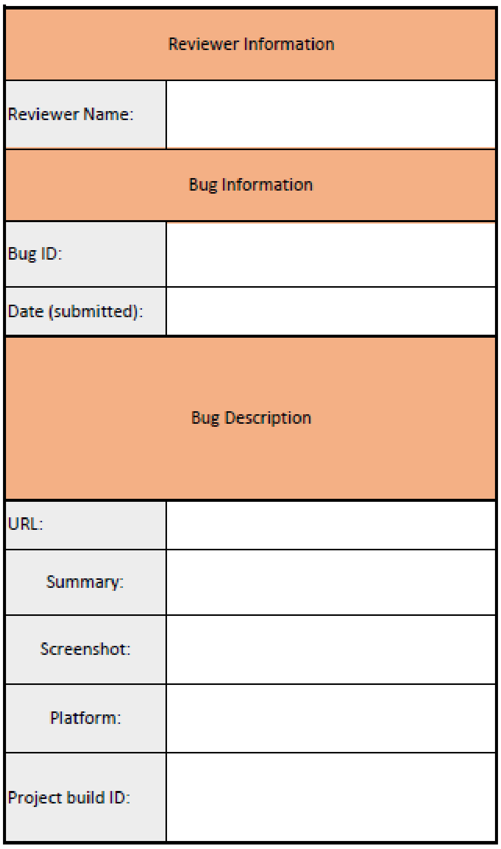
Main features and functionalities will be locked down. All elements, assets, props and functionalities are working and will be tested in the Beta.

This process will mainly consist in a broken game: the game at the polished version with all mechanics and dynamics implemented to approach all types of bugs and dysfunctionalities.In beta testing, the task is more of polishing the program so that it works nicely for everyone rather than ensuring that it actually works. Problems are then patched prior to the release of the final version of the software.

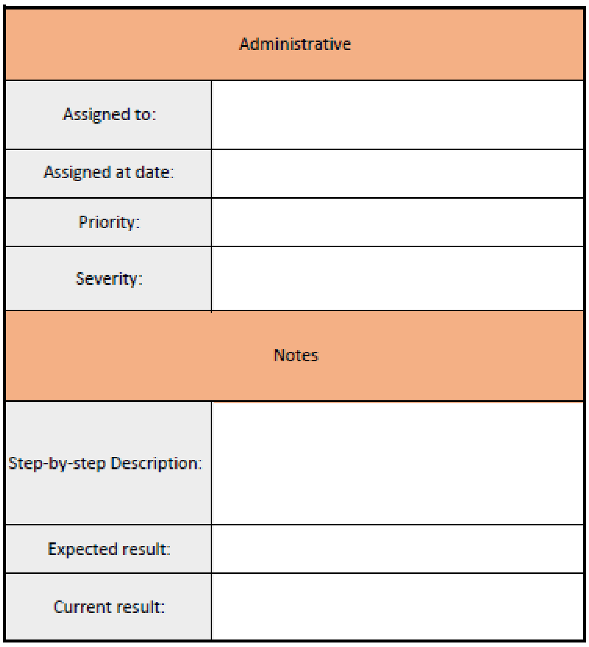
The public of the game will get to play the game and it will be submitted to a stress testing process. The QA department will take control in the possible differents techniques of testing with possible target users.

BUG PROPERTIES AND WORKFLOW

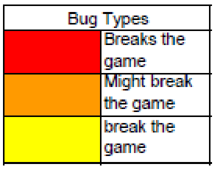
This will be the document that contains the process to determine and inform about a bug or error in the development of the game.



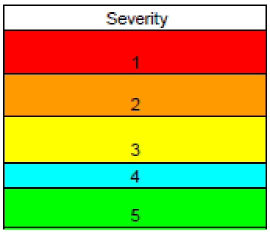
Reviewer Information	
Reviewer Name:	
Bug Information	
Bug ID:	
Date (submitted):	
Bug Description	
URL:	
Summary:	
Screenshot:	
Platform:	
Project build ID:	



Administrative	
Assigned to:	
Assigned at date:	
Priority:	
Severity:	
Notes	
Step-by-step Description:	
Expected result:	
Current result:	



Bug Types	
🟥	Breaks the game
🟧	Might break the game
🟨	break the game



Severity
🟥 1
🟧 2
🟨 3
🟦 4
🟩 5

SOFTWARE USED IN MYTHOLOGY PARADE

The software the team needs heavily depends on their role in the team, the software will be structured as following:

Code:

- Visual Studio
- Draw.io

Art:

- Photoshop
- Procreate
- Tiled
- Adobe Illustrator

- Audacity

QA:

- Google Docs
- Visual Studio

UI:

- Photoshop
- Draw.io
- Adobe Illustrator

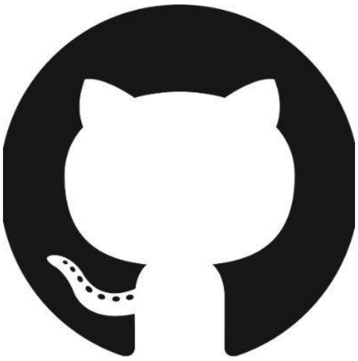
Management:

- HacknPlan
- Slack
- Google Calendar

Design:

- Google Docs
- Photoshop

Ai



h

31