ONTARIO GAME TESTERS

GAME TESTING PRIMER

A simple method

Version 2.1

GOALS OF GAME TESTING

Our goals as testers are to help the development team.

- Know what bugs and issues are present in their game
- See how someone else would play the game
- Understand how and where these defects occur
- Confirm that their fixes are successful
- Launch a better game!

FRESH PLAY

Each tester has a one-time ‘fresh’ impression of a new game that we can use to our advantage. For this first playthrough, we act like any player would and write a log of observations and impressions.

This log can be recorded as a video, transcribed audibly using speech-to-text, or written manually. The important thing is to capture (mostly) unfiltered impressions of the game as we play it for the first time.

We then use this content to prepare a **fresh play document** that describes the playthrough. This helps the developer understand how the game will be perceived by first time players.

FRESH PLAY

Example (Early Game)

I’m launching the game. Cool studio logos. Main menu. The resolution doesn’t seem right so I’m going to check settings. Changing to 1920x1080 and very-high graphics. Apply… save. New game. Cool back story. Love the motion graphics to support the text. This is going on a bit long. Alright, looks like this is a tutorial. No problem. Hope I get to gameplay soon. Oh, more dialog. Who is this guy? Okay now I can run around. Let’s talk to this person.

Example (Mid Game)

Okay, died to this boss last time because I didn’t know they were going to shoot fire everywhere in phase 2. Going to equip some fire resist gear and then try to keep my distance. Doing the boss walk. Battle start. Whew that swipe is hard to see coming. I think phase 2 will happen soon, going to back up. Yep, there it is. Okay sweet, still alive. I think they’re almost dead. Woo, dead. Death animation was a bit odd, like, they just fell through the floor.

REGULAR TESTING

The backbone of our work is regular testing. This is where we,

- Play the game, one **build** or **version** at a time
- Pay close attention to **bugs** (defects) and **feedback** (possibly undesired effects)
- Report on **bugs** and **feedback** by making **tickets** in a **bug tracker** like Trello
- Add information to each ticket that makes the bug or issue easily **reproducible**
- Share **additional information** to help the development team understand the bug, such as **written details**, a **player log, save file, screenshot,** or **video recording**

SPECIFIC TESTING

This is triggered by the development team telling us about something new.

- **New game features** usually as patch notes or design documents that describe expected behaviour.
- **Problem areas** like systems or features that are unfinished, that the developers have noticed some problems with, or areas that have simply undergone a lot of changes.

From this, we test more rigorously and specifically than in a regular play-through using strategies like **Matrix Testing**. We also compare features against the designs (if provided) to ensure the feature is behaving according to the intended design. This is called **Design Verification**. This is important because sometimes features can fail to function properly without exactly looking like a bug.

SPECIFIC TESTING

Matrix Testing is a great way to thoroughly test specific features together. In Matrix testing, we try to encounter every possible combination of certain features.

For example, in a game with 3 playable characters and 3 weapon choices, we have 9 possible combinations. Add three different levels, and we now have 27 combination scenarios.

At a certain point, matrix testing can become cumbersome for humans to test, and the developer may want to consider automated testing.

SPECIFIC TESTING

Design Verification is another item that could be in your testing toolbox - if the developer has provided you with documentation.

(Note that we don't always have access to game design documents or feature descriptions, in which case we cannot conduct Design Verification).

This testing requires matching up of the described feature to the behaviour of the game. It's important to not only consider the mechanics of the feature, but also the intention. Systems and code can often behave like a 'monkey's paw' and follow the rules given to them to the detriment of the overall intention. These cases are important to highlight.

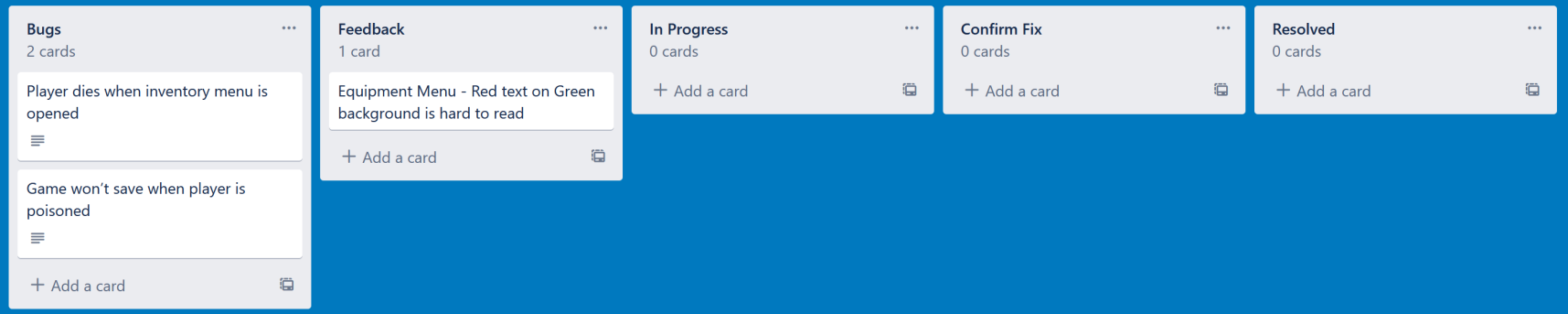
CONFIRMATION TESTING

Confirmation testing is where we validate the development team's fixes.

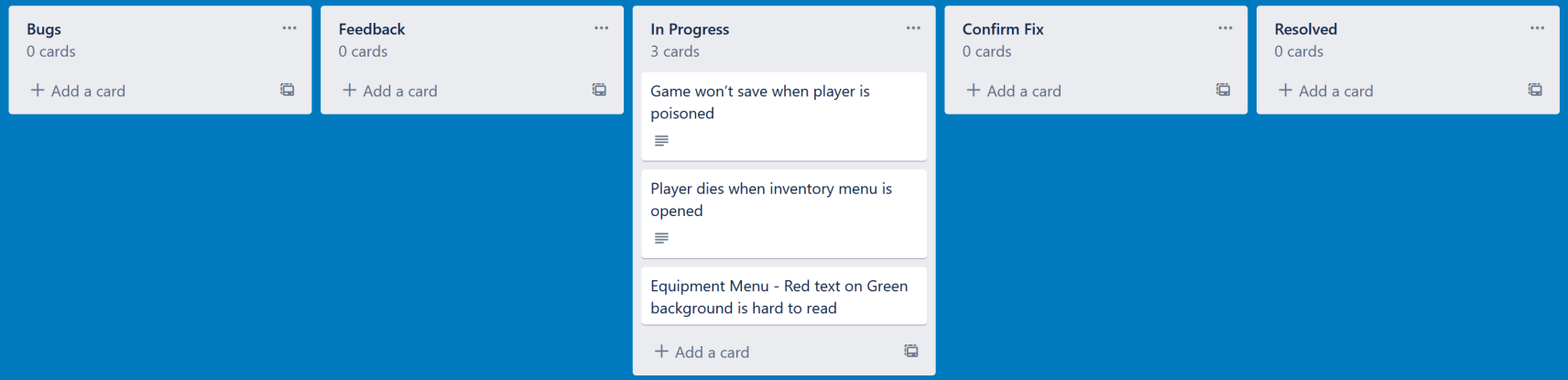
- Play the game's next **build** or **version**
- Work through the list of **confirmed fixes**, marked by the development team
- Try to make each **allegedly resolved** bug or issue occur once again
- If the bug or issue no longer occurs, mark it as **resolved**
- If the bug still occurs, mark it as a **bug** and offer more information if you can
- If the bug described in the ticket is technically resolved, but something similar is happening, close the ticket and open a new one.

TESTING FLOW

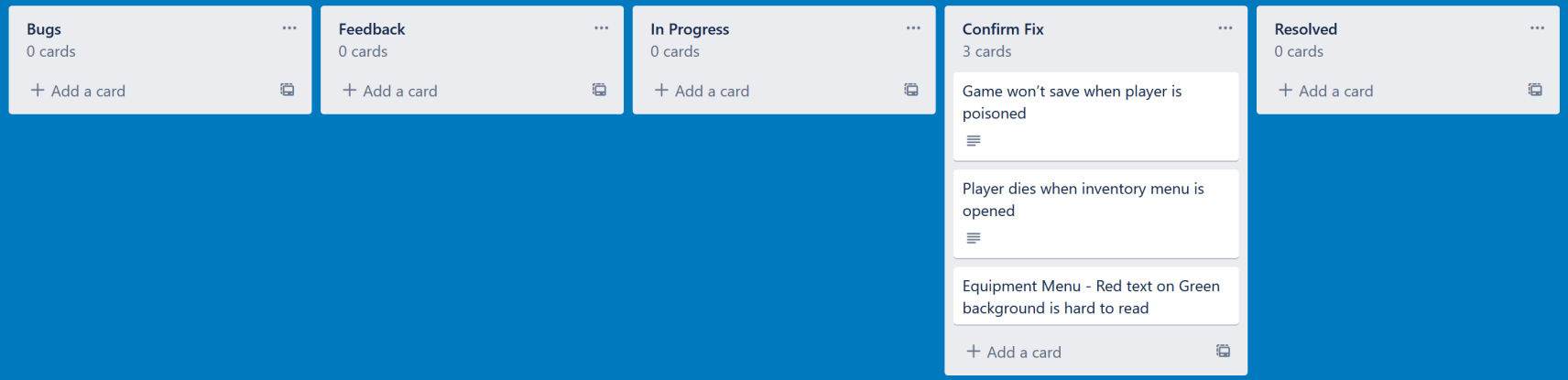
Example on a Trello Board



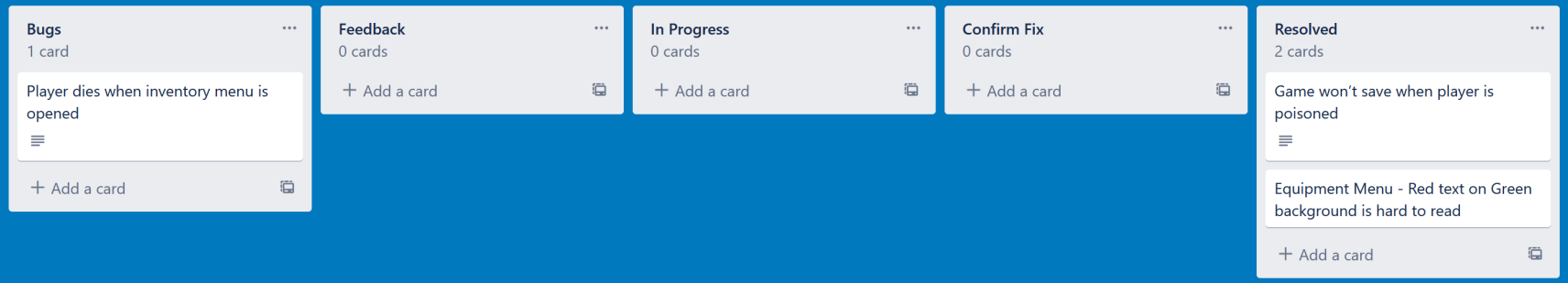
An example Trello board with 2 **Bugs** and 1 **Feedback** tickets.



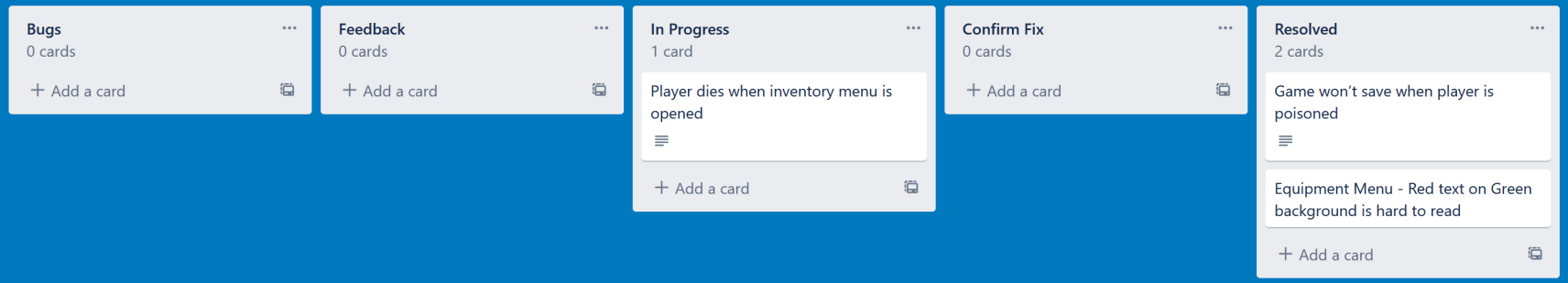
Developers see our logged tickets in **Bugs** and **Feedback**, and move them to **In-Progress** as they work on them.



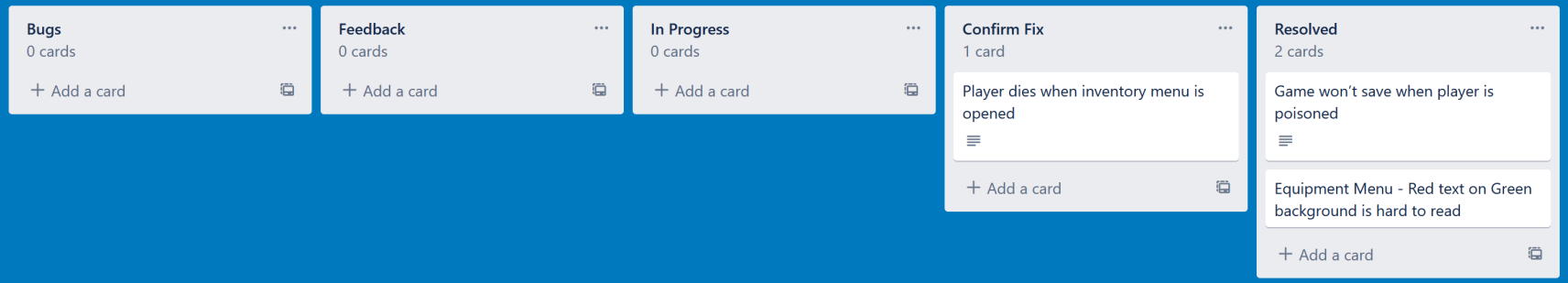
Developers feel they've fixed these tickets, and move them into **Confirm Fix** for us to check.



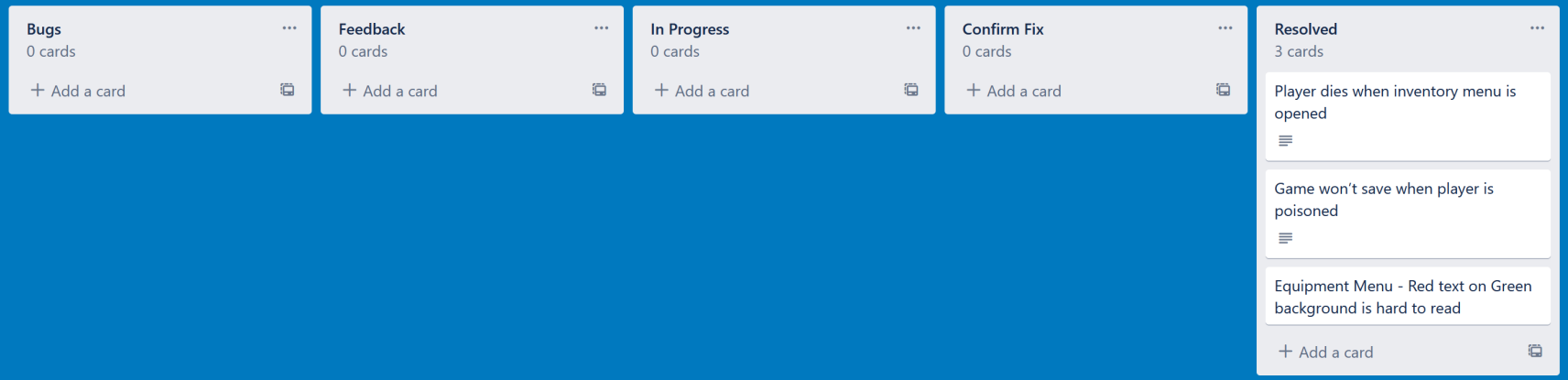
We confirm two tickets are resolved, but one isn't!
That unresolved ticket is moved back to the **Bugs** list.



Developer tries again, this time chatting with us a bit about how the issue occurs over Discord with screen share.



Developer now believes they've fixed it.



We confirm that the ticket is fixed.

WONT FIX?

Sometimes timelines are tight, and teams need to go into triage mode. This is where the **Won’t Fix** list can show up. This is a list of bugs and issues that the development team has specifically decided not to fix (at least for now). It’s still good to have these tickets on the board, but it’s a helpful tool for developers to focus on high priority fixes.

CLEANUP

Over time, especially on teams with multiple testers, the list of bugs and issues will invariably include duplicates and redundancies. Therefore, it’s useful to assign one person to oversee **cleanup** each cycle. During cleanup, we will delete and/or merge tickets that describe the same issue. When doing this, remember that each separate ticket may include useful information. Descriptions, player logs, screenshots, etc should be copied into the single remaining card in order for the developers to have complete information to support fixing.

CREATING TICKETS

Most of our work product as testers are bundles of information called **tickets**. These are often represented as cards with a **title**, **description**, and **attachments** in a project management tool like Trello or JIRA. The best tickets are clear and concise.

When possible, we use extra software that automates some of this. Our **testing tools** conveniently allow you to log tickets from within the game and automate the capture and attachment of a screenshot and player log.

It’s useful for us and the development team to see these tickets on a high-level view, as a list of to-dos or cards on a screen, where we can see the state of each ticket at a glance.

CREATING TICKETS

Good Title

“Player dies when inventory menu is opened”

Bad Title

“I was playing the game and then I opened up the inventory menu with the I key and then all of a sudden the player died and I didn’t like that”

Remember that title is the **short description**. It’s great to include extra content in the ticket, but that extra information should go into the description. Don’t include flowery language like “all of a sudden” and refrain from sharing personal feelings like ”I didn’t like that.”

CREATING TICKETS

Good Description

"Opening the inventory menu results in the player dying. I noticed this when I closed the menu. It is not clear to me when exactly the player dies between opening and closing the menu, since it obscures the game view. This happened in the Mico's Hangar level."

Bad Description

"I don't know why but the inventory menu kills the player. To reproduce, go to Mico's Hangar and press I and you'll see the menu deals damage to the player (not sure how much damage, but it's a lot because it kills me at full health)"

Here, the bad description places blame on the inventory UI which may be an incorrect assumption that leads the developer astray. Correlation does not equal causation. Just describe what happened and don't leap to conclusions.

CREATING TICKETS

Good Title

“Game won’t save when player is poisoned”

Bad Title

“I got poisoned in the swamp environment and two of my characters died and then I went to go save the game and then I closed and re-launched the game from desktop and my game wasn’t saved!”

Like in the previous example, a long title usually suggests that description and title are being mixed up. Keep the title short. Include reproduction steps in the description. Title tells us the problem. Description tells us the background information.

CREATING TICKETS

Good Description

"In the marsh of sorrow level, just after the collapse of Farrow cutscene, my party was poisoned by a group of Mutant Rats. I saved and quit to desktop, then relaunched the game, noticing that the save game was from before the Farrow cutscene. Repeating this yielded the same results. It appears that save games don't work while party members are poisoned."

Bad Description

"I got poisoned and then the save function didn't work anymore"

The bad description comes to the same conclusion as the good but omits details that would be helpful for developers in isolating where this issue occurs. Noting where the player is progression-wise in the game is often helpful. As well, mentioning the mechanics of the failed save (save, quit, relaunch) is more informative than saying "the save function didn't work anymore" which could suggest to the developer that there is something wrong with the save UI crashing or not showing.

CREATING TICKETS

There is more than one way to write a good description. In some cases, it may make sense to walk through the steps in point form rather than as a written summary. This can offer more information and is sometimes easier to write. For example,

1. Walked around the marsh or sorrow area
2. Encountered a mutant rat
3. Rat poisoned the player character
4. Defeated rat
5. Returned to world map
6. Opened up the menu and selected 'save’
7. Quit game (to desktop)
8. Opened game
9. Clicked continue
10. Saw that the most recent save was not the one from previous steps
11. Note the player isn't poisoned in this older save
12. Make a new save, quit to desktop again, re-open game and click continue
13. Save game loads as expected

CREATING TICKETS

Every project is a little different, but some other things that are usually helpful to include in each ticket are,

- **Reporter** - the name of the person who logged the issue. This helps the development team know who to reach out to for clarification or help with reproducing.
- **Severity** – while some teams operate with the belief that all bugs are high priority, and some teams prefer for the development team or project manager to set priority, it can be useful to include the severity of the bug in the report. Common severity levels are *"High"* for bugs that crash the game or cause data loss, *"Medium"* for bugs that impede or ignore progress, and *"Low"* for bugs that are technically incorrect but don't affect overall functionality.
- **Version** – most integrated trackers will include this, otherwise it's important to mention what version (or build executable date) the issue was encountered on.

CONDUCTING YOURSELF

Soft Skills and Strategies

ATTITUDE

You are a Game Developer

Everyone who contributes to the completion of the game is considered a game developer. That includes testers! While you may not have a classically creative or engineering contribution, you’re helping the game come to life and be a success. Testing is a stepping-stone for some, and a whole illustrious career path for others.

ATTITUDE

Supporting and Non-Judgmental Attitude

We encourage studios to start testing as early as possible, since bugs and other issues tend to fester and get worse the longer they are left. However, the reality is that most studios don’t formally test with anyone outside the internal development team until the game is nearly complete. We’re always happy to help and won’t shame anyone for what we think are suboptimal practices. Everyone is just trying their best, after all.

ATTITUDE

Your contributions are valuable

Developers, especially Programmers, can sometimes come off as a prickly or annoyed, especially when dealing with bugs. Remember that this is difficult work, and that our common enemy are the bugs, not each other. Your contributions are valuable, even if they appear to cause frustration, and you will ultimately be appreciated for your help. You can always lean on your supervisor for a pep-talk if you’re feeling down.

ATTITUDE

Your role is tester

While you may have some great suggestions to make about improvements to the game, remember that we usually come in near the **end** of development, where game design is largely complete. Think twice before suggesting new features or mechanics and consider that the developer has likely already considered these, and that late changes to mechanics usually come with considerable effort. Our job is not to design the game, it is to be an **intelligent mirror** who can show the developers objective details.

TIME MANAGEMENT

Set yourself up for success

Most of our testing projects are calculated with the following measurements,

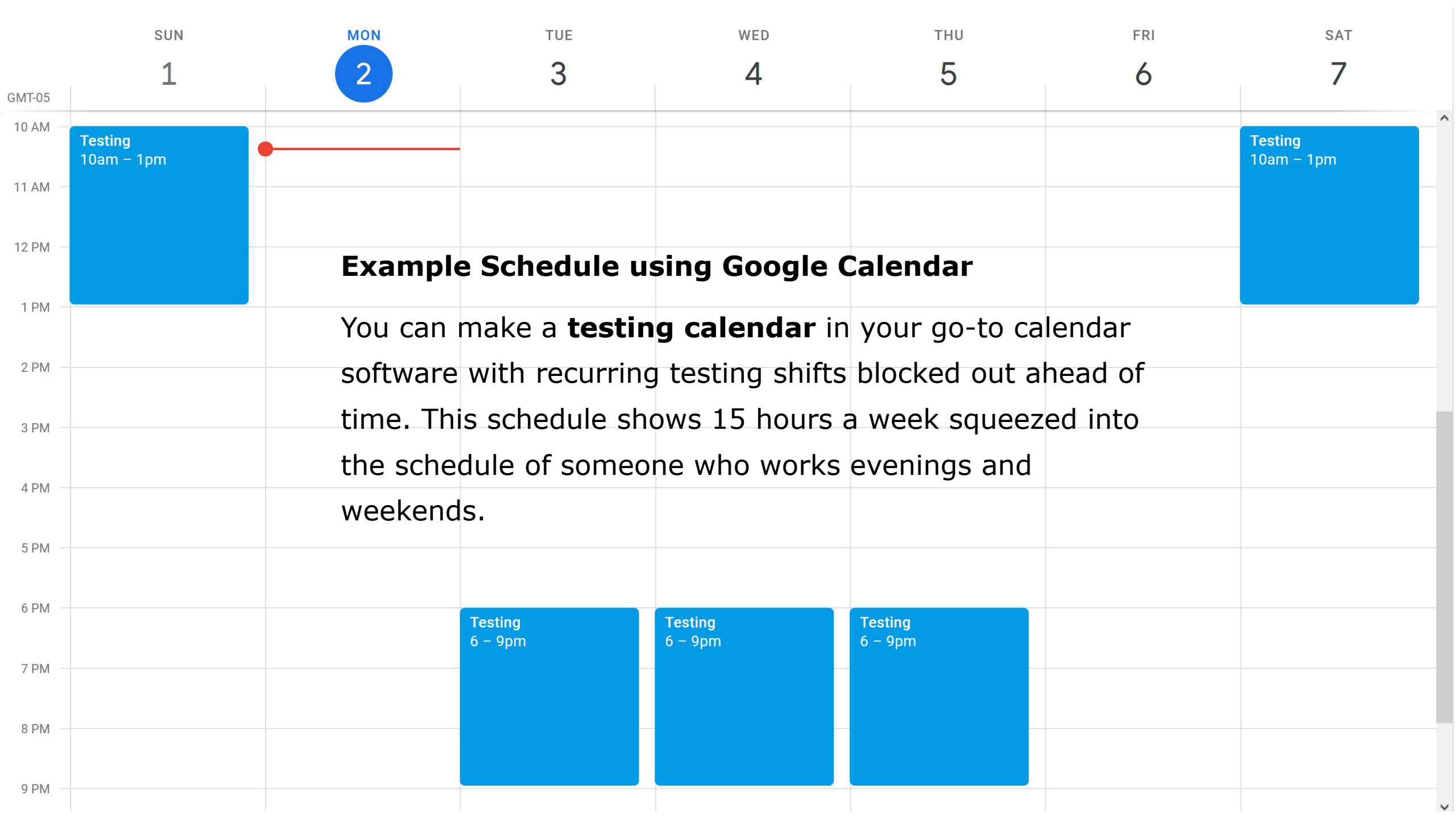
Number of testers
X
Hours per week
X
Number of weeks

TIME MANAGEMENT

Set yourself up for success

You’re being counted on to work a certain number of hours per week. It can be challenging to spend these hours **as you go** wherever you find time. Therefore, it is strongly recommended that you **block out time** pre-emptively during the week, so you know when you will work. This allows you to protect that time against other appointments, and work with more focus and less stress.

This not only allows you to know when you’re working, it makes it easier to plan your playtesting around the releases of new builds. We strongly suggest keeping your testing blocks to a minimum 3 hours in order to gain the focus needed for good work (note: even a 3 hours shift should include short breaks to keep yourself healthy and sharp).



Example Schedule using Google Calendar

You can make a **testing calendar** in your go-to calendar software with recurring testing shifts blocked out ahead of time. This schedule shows 15 hours a week squeezed into the schedule of someone who works evenings and weekends.

THANK YOU!

Thanks for reading the Ontario Game Testers **Game Testing Primer**.

You can find us online at **www.ontariogametesters.com**

We provide testing services and consultation.

ONTARIO GAME TESTERS