# Contributing to Google Play Music Desktop Player

? [Google Play Music Desktop Player](https://github.com/MarshallOfSound/Google-Play-Music-Desktop-Player-UNOFFICIAL-) and want to get involved?

Thanks! There are plenty of ways you can help!

Please take a moment to review this document in order to make the contribution

process easy and effective for everyone involved.

Following these guidelines helps to communicate that you respect the time of

the developers managing and developing this open source project. In return,

they should reciprocate that respect in addressing your issue or assessing

patches and features.

## Using the issue tracker

The [issue tracker](https://github.com/MarshallOfSound/Google-Play-Music-Desktop-Player-UNOFFICIAL-/issues) is

the preferred channel for [bug reports](#bugs), [features requests](#features)

and [submitting pull requests](#pull-requests), but please respect the following

restrictions:

\* Please \*\*do not\*\* use the issue tracker for personal support requests (use

[Gitter](https://gitter.im/MarshallOfSound/Google-Play-Music-Desktop-Player-UNOFFICIAL-)).

\* Please \*\*do not\*\* derail or troll issues. Keep the discussion on topic and

respect the opinions of others.

<a name="bugs"></a>

## Bug reports

A bug is a \_demonstrable problem\_ that is caused by the code in the repository.

Good bug reports are extremely helpful - thank you!

Guidelines for bug reports:

1. \*\*Use the GitHub issue search\*\* &mdash; check if the issue has already been

reported

2. \*\*Check if the issue has been fixed\*\* &mdash; try to reproduce it using the

latest commits on `master` in the repository.

3. \*\*Isolate the problem\*\* &mdash; try determine the cause or probable cause

of the bug and maybe even include a reference to a line in the code.

A good bug report shouldn't leave others needing to chase you up for more

information. Please try to be as detailed as possible in your report. What is

your environment? What steps will reproduce the issue? What browser(s) and OS

experience the problem? What would you expect to be the outcome? All these

details will help people to fix any potential bugs.

You should always include your system information in a bug report so developers

can easily reproduce the bug in a similar environment

Example:

> Short and descriptive example bug report title

>

> OS: [Your OS here]

> Application Version: [The version of GPMDP you are running]

>

> A summary of the issue. If

> suitable, include the steps required to reproduce the bug.

>

> 1. This is the first step

> 2. This is the second step

> 3. Further steps, etc.

>

> Any other information you want to share that is relevant to the issue being

> reported. This might include the lines of code that you have identified as

> causing the bug, and potential solutions (and your opinions on their

> merits).

<a name="features"></a>

## Feature requests

Feature requests are welcome. But take a moment to find out whether your idea

fits with the scope and aims of the project. It's up to \*you\* to make a strong

case to convince the project's developers of the merits of this feature. Please

provide as much detail and context as possible.

Any Feature Request for "Chromecast Integration" will be immediately closed as

a duplicate unless it brings something new to the discussion such as a code

example or theory of how to implement such a feature. This does not mean

Chromecast Integeration won't happen, but for now it is too complex for me to

handle.

<a name="pull-requests"></a>

## Pull requests

Good pull requests - patches, improvements, new features - are a fantastic

help. They should remain focused in scope and avoid containing unrelated

commits.

\*\*Please ask first\*\* before embarking on any significant pull request (e.g.

implementing features, refactoring code, porting to a different language),

otherwise you risk spending a lot of time working on something that the

project's developers might not want to merge into the project.

Please adhere to the coding conventions used throughout a project (indentation,

accurate comments, etc.) and any other requirements.

Adhering to the following process is the best way to get your work

included in the project:

1. [Fork](https://help.github.com/articles/fork-a-repo/) the project, clone your

fork, and configure the remotes:

```bash

# Clone your fork of the repo into the current directory

git clone https://github.com/<your-username>/Google-Play-Music-Desktop-Player-UNOFFICIAL-.git

# Navigate to the newly cloned directory

cd Google-Play-Music-Desktop-Player-UNOFFICIAL-

# Assign the original repo to a remote called "upstream"

git remote add upstream https://github.com/MarshallOfSound/Google-Play-Music-Desktop-Player-UNOFFICIAL-.git

```

2. If you cloned a while ago, get the latest changes from upstream:

```bash

git checkout master

git pull upstream master

```

3. Create a new topic branch (off the main project development branch) to

contain your feature, change, or fix:

This branch should be in the style

\* `feature/feature-info`

\* `bugfix/bugfix-info`

\* `dev/generic-info`

\* `upgrade/[dependency]-upgrade-[version]` \*only use this when upgrading a major internal dependency such as Electron\*

```bash

git checkout -b <branch-name>

```

4. Commit your changes in logical chunks. If you are making a single feature or single bugfix there should only be

one commit in the PR, if your commit has to include multiple features then you can have one commit per feature.

Before making a PR please squash your commits into these logical chunks

5. Locally rebase the upstream development branch into your topic branch:

\*\*merging upstream master into your branch will be rejected\*\*

```bash

git pull [--rebase] upstream master

```

6. Push your topic branch up to your fork:

```bash

git push origin <topic-branch-name>

```

7. [Open a Pull Request](https://help.github.com/articles/using-pull-requests/)

with a clear title and description. Pull Requests are reviewed on [http://reviewable.io](http://reviewable.io)

\*\*IMPORTANT\*\*: By submitting a patch, you agree to allow the project

owners to license your work under the terms of the [MIT License](LICENSE.txt).