First, thank you for contributing to DIM! We're a community-driven project and we appreciate improvements large and small.

Here are some tips to make sure your pull request can be merged smoothly:

1. If you want to add a feature or make some change to DIM, consider [filing an issue](https://github.com/DestinyItemManager/DIM/issues/new) describing your idea first. This will give the DIM community a chance to discuss the idea, offer suggestions and pointers, and make sure what you're thinking of fits with the style and direction of DIM. If you want a more free-form chat, [join our Discord](https://discordapp.com/invite/UK2GWC7).

1. Resist the temptation to change more than one thing in your PR. Keeping PRs focused on a single change makes them much easier to review and accept. If you want to change multiple things, or clean up/refactor the code, make a new branch and submit those changes as a separate PR.

1. All of our code is written in [TypeScript](https://typescriptlang.org) and uses React to build UI components..

1. Take advantage of the [native JavaScript Array methods](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array) and the [Lodash](https://lodash.com/) library to write compact, easy-to-understand code.

1. Be sure to run `yarn run lint` before submitting your PR - it'll catch most style problems and make things much easier to merge.

1. Don't forget to add a description of your change to `CHANGELOG.md` so it'll be included in the release notes!

## Developer Quick start

1. [Install Pre-requisites](#pre-requisites)

1. [Clone](#clone-the-repo)

1. [Get your own API key](#get-your-own-api-key)

1. [Start Dev Server](#start-dev-server)

1. [Enter API credentials](#enter-api-credentials)

### Pre-requisites

\* Install [NodeJS](https://nodejs.org/).

\* Install [Yarn](https://yarnpkg.com/en/docs/install). If you're used to NPM, see "[Migrating from NPM](https://yarnpkg.com/lang/en/docs/migrating-from-npm/)".

\* Windows-based developers will need to install `windows-build-tools` (`yarn global add windows-build-tools`) globally prior to running `yarn install`. Refer to issue #1439 for [details](https://github.com/DestinyItemManager/DIM/issues/1439).

\* It is highly recommended to use [VSCode](https://code.visualstudio.com/) to work on DIM. When you open DIM in VSCode, accept the recommended plugins it suggests.

\* Linux-based developers will need to install `build-essential` (`sudo apt-get install -y build-essential`) prior to running `yarn install`.

\*\*Docker Supported Development\*\*

\* As an alternative to installing the above, we support a docker-based development environment. See: [Docker Development Guide](Docker.md)

### Clone the repo

git clone https://github.com/DestinyItemManager/DIM.git

### Get your own API key:

1. Goto [Bungie](https://www.bungie.net/en/Application)

1. Click `Create New App`

1. Enter any application name, and `https://github.com/YourGithubUsername/DIM`

1. For `Oauth Client type` select `Confidential`

1. Set your redirect url to `https://localhost:8080/return.html` (or whatever the IP or hostname is of your dev server)

1. Select all scopes \_except\_ the Administrate Groups/Clans

1. Enter `https://localhost:8080` as the `Origin Header`

### Start Dev Server

\* Run `yarn install`

\* Run `yarn start`

\*Note:\* on Windows, when running `yarn install` the Git Bash shell may fail to fetch all necessary packages even when run as Admin ([details](https://github.com/DestinyItemManager/DIM/issues/2487)). If that's the case, simply use cmd as Admin instead.

### Enter API Credentials

This step will need to be done each time you clear your browser cache. You will be automatically redirected to a screen to enter these credentials

if the app can't load them from local storage when it starts.

1. Open your browser and navigate to https://localhost:8080

1. Copy your API-key, Oauth Client\_id, and OAuth client\_secret from bungie.net into DIM developer settings panel when it is loaded.

### Development

\*\*Overview\*\*

The `yarn start` step will create a hot-loading webserver, and a TLS cert/key pair. You will access your local development site by visiting https://localhost:8080.

You will likely get a security warning about the certificate not being trusted. This is because it's a self-signed cert generated dynamically for your environment,

and is not signed by a recognized authority. Dismass/advance past these warning to view your local DIM application.

\*\*Check code Style\*\*

\* `yarn lint` will tell you if you're following the DIM code style (and automatically fix what it can).

Check out the [docs]() folder for more tips.

\*\*Translation\*\*

\* We use [i18next](https://github.com/i18next/i18next) for all our translated strings, so if you want to translate something that's currently English-only, take a look at that. Usually it's as simple as replacing some text with `<span>{t('KEY')}</span>` and then defining KEY in the `config\i18n.json` file.

\* `yarn i18n` will add, sort, and prune `src/locale/dim.json`. You should never manually edit `src/locale/dim.json`. Some keys are obfuscated by code and will need to be added as comments into the code such as `// t('LoadoutBuilder.ObfuscatedKey1') t('LoadoutBuilder.ObfuscatedKey1'). If you have any questions ping @delphiactual via GitHub, Slack, or Discord.