# Guidelines:

If you would like to contribute to the project, please contact the DAD-Bod Development team and follow all proposed guidelines.

## General Guidelines:

* All Code should be submitted through a 'pull' request to JacobW1997.
* All variable names should follow the camel case convention HTML elements should follow the same convention.
* Model and Class names should have the first letter of each word capitalized E.g. “ThisClassFunction”.
* ID is used for each primary key.
* Variable names should be short and concise
* Comments should be used to describe functionality

## Code:

* Use external JavaScript files
* Curley braces should be on their own line
* Variables should be declared at the top of each function
* Dioxygen style comments should be used for functions that have vague/complex operations

## Git:

* Use branches
* When creating a pull request make sure you are requesting to pull to the development branch.
* Commit often (don't feel like you must make major, complete, changes or new features before committing)
* Write good commit messages
* Don't commit code that doesn't compile
* Don't add and commit any files that are auto-generated (i.e. html documentation, .o, .tmp, ...)
* Always pull before working on your local repository.
* All pull requests should be made before 5:00 pm on Sunday

## Style:

* TBD

## Software Construction Process:

During this project we will be practicing the disciplined agile delivery framework defined by the Agile methodology. Our construction process includes sprints that are completed every two weeks, following our sprint retrospective meeting. We have also integrated various meeting types defined by the SCRUM framework, some of these meetings include: Daily, Weekly, Stand-Up, Sprint Retrospective, and the end of Sprint review meeting.

## Pull Request Model (Taken from source below)

1. Before doing any work, ensure you have pulled the latest changes from the upstream remote repository to your forked repository in both the master and development branches. Make sure to push these changes to your remote repository.
2. Create your feature branches off the dev branch and do all your development on this branch. Try to commit as often as possible.
3. When you are ready to merge your work into the overall project, **make sure to complete the following instructions First:**
   * 1. Checkout the development branch and pull any changes that have occurred since you last branched.
     2. Checkout your feature branch and merge dev into your branch.
     3. Fix any possible merge conflicts.
     4. Test thoroughly by building and running the project, making sure everything still looks and works the way you intended.
4. Push your feature branch to your remote repository.
5. From your forked repository on GitHub, create a new pull request:
   * 1. Ensure that the branch you are merging from is your feature branch on your forked repository.
     2. Ensure that the branch you are merging into is the **development** branch on the upstream repository (you will likely need to change this from master to dev).
     3. Fill out the title and description fields as necessary.
     4. If you are no longer going to be working on this feature branch, feel free to check the box to close the branch after the pull request is merged.
     5. Create the pull request.
6. If you continue to work on this feature branch before the pull request is accepted, or if the upstream repository suggests any changes before the merge, you can still make changes on your branch and push them to your remote repository. Doing so will automatically update your pull request.
7. Once your pull request is accepted and merged, pull the updates to your local dev branch and push to your remote repository.

Source: <https://bitbucket.org/cstanavige12/startech/src/036ff53819fc4aedee921814e5aa870edde12638/guidelines.md>