Team Name	Documentation		
	Code of Conduct		
Requirement 1: Quality of Writing	Few grammatical, technical, and semantic errors. Easy to understand for a non-native speaker.	Grammatical, technical, and semantic errors are noticable. May be difficult for non-native speakers to understand.	Grammatical, technical, and semantic errors are made often. Impossible to understand for non-native speakers.
Requirement 2: Clarity of Rules	Rules are understandable. Have a clear reason for being.	Most rules make sense, may not have a clear reasoning.	Rules seem completely absurd and unexplainable. Or they simply don't exist
Requirement 3: Defined Process of Enforcement	Step by step guide for dealing with violations, a clearly defined system of discipline	Process is relatively vague or confusing, but reasonably written.	No process is defined.
Requirement 4: Two People Designated to Enforcement	Two people, at minimum, designated to enforce Code of Conduct and actively enforce it. This reduces bias from a single individual person.	Someone is responsible for enforcement, but may be neglecting its enforcement or hard to reach.	No one is responsible for enforcing the Code of Conduct
	Contributing Guidelines		
Requirement 1: Quality of Writing	Few grammatical, technical, and semantic errors. Easy to understand for a non-native speaker.	Grammatical, technical, and semantic errors are noticable. May be difficult for non-native speakers to understand.	Grammatical, technical, and semantic errors are made often. Impossible to understand for non-native speakers.
Requirement 2: Explainantion of Common Practices	Common practices can be interpreted by someone outside the project, granted they have knowledge of general open source project practices.	Common practices require a level of special background knowledge.	Common practices are exclusively understood by team members or they don't exist.
Requirement 3: Guidelines for Filiing an Issue	Anyone can follow the guide and successfully file an issue that follows a common format.	Guidelines may not fit many issues being filed but overall provide a unitary theme to issues submitted.	No one can follow the guide, or the guide doesn't exist.
Requirement 4: Guidelines for Pull Request	Anyone familiar with Git and the tools needed for the project can follow and successfully submit a pull request	May need project specific knowledge outside of the guidelines to successfully submit a request	Pull requests have no guide, making it difficult for people to submit.
Requirement 5: Timelines and Expectations	Timelines are assigned to tasks, issues, and requests and those responsible can easily understand the expectations therein.	Timelines are assigned, but aren't very specific, or expectations aren't clearly communicated.	Timelines are functionally non-existant, no pretense of expectations.
Requirement 6: Method of Further Contact	Further contact leads to relevant contact information, emails, social media, and possibly phone numbers. This contact information will lead them to someone.	Further contact leads to one email or account that someone attends to once in a while.	Further contact leads to nowhere.
	Developer Documentation		
Requirement 1: Quality of Writing	Few grammatical, technical, and semantic errors. Easy to understand for a non-native speaker.	Grammatical, technical, and semantic errors are noticable. May be difficult for non-native speakers to understand.	Grammatical, technical, and semantic errors are made often. Impossible to understand for non-native speakers.
Requirement 2: Ease of Editing	Anyone with internet access can suggest a change to the documentation.	May have some barrier of entry, but the community can still submit a suggestion to the documentation.	Documentation can only be modified by team members.
Requirement 3: Development Environment Explaination	The developement environment is fully explained, any dependencies shown, and all technical setup clairified. The process of setting up a development environment is simple.	The developer environment has some explaination. may have a few missing details, but gets most of the setup communicated correctly. **OR** Development environment setup is explained in detail but is difficult to manage, has potential of causing issues with other environments on the user's box, etc.	Development environment is not mentioned. No way for the developers to easily find out how to setup the project on their device.
Requirement 4: Project Hierarchy Explaination	The organization of the repositories is explained, typically visually. With each component getting a breif explaination of what it is and how it fits into the architechture.	The organization of the repos is explained, without visuals. Each component may or may not get a proper explaination.	Organization is insufficiently explained, no context for how the components of the project fit together.
Requirement 5: Regularly Updated	Documentation is updated in parallel with changes to the code.	Documentation is updated frequently, not as often as the project however.	Documentation is rarely or never updated.
	FAQ		
Requirement 1: Quality of Writing	Few grammatical, technical, and semantic errors. Easy to understand for a non-native speaker.	Grammatical, technical, and semantic errors are noticable. May be difficult for non-native speakers to understand.	Grammatical, technical, and semantic errors are made often. Impossible to understand for non-native speakers.
Requirement 2: Relevant Questions	Questions are common ones many new users and developers ask.	Questions are ones the team thinks users and developers will have but not comprehensive.	Questions are obscure and esoteric.
Requirement 3: Clear Answers	Answers are detailed, well-phrased, and helpful.	Answers are decent, may be lacking in detail or phrased somewhat confusingly, but they get the message across.	Answers are unhelpful.
	READMEs		
Requirement 1: Quality of Writing	Few grammatical, technical, and semantic errors. Easy to understand for a non-native speaker.	Grammatical, technical, and semantic errors are noticable. May be difficult for non-native speakers to understand.	Grammatical, technical, and semantic errors are made often. Impossible to understand for non-native speakers.
Requirement 2: General overveiw of content	Overveiw covers all major aspects of the project or component in a well-written, easy-to-navigate way.	Overveiw covers most aspects of the project or component, fairly well-written and organized.	Overveiw doesn't cover any or very few aspects of the project.
Requirement 3: Installation Instructions	Provides a step-by-step guide for getting the content of the repository installed on a machine.	Provides a guide to installation for most compatible OSes.	Installation guide is insufficient, does not tell the user what they need for installation.

Requirement 4: Leads to Other Sources	Other resources, (i.e. documentation, website, wiki), are linked in the readme for further information.	Other resources are linked, but not as many as their could be.	No resources are linked in the readme.
Requirement 5: Basic Functionality Explained	How this project or component works / fits into the larger project is explained in detail.	Project / component is explained, but may be missing a few key details.	Project / component has no explaination.
Requirement 6: Mission Statement			
	User Documentation		
Requirement 1: Quality of Writing	Few grammatical, technical, and semantic errors. Easy to understand for a non-native speaker.	Grammatical, technical, and semantic errors are noticable. May be difficult for non-native speakers to understand.	Grammatical, technical, and semantic errors are made often. Impossible to understand for non-native speakers.
Requirement 2: Quick Start Guide	Quick Start Guide provides an easy to access way to install, setup, and utilize the project.	Quick Start Guide does not completely cover the starting process, but gives a sufficient start.	Quick Start Guide may give a few tips, but does not cover the starting process in a meaningful way.
Requirement 3: Project Explaination	Project Explaination details the goals of the project, the state the project is in, and current work in progress.	Project Explaination may be lacking in detail, but covers all the topics it needs to.	Project Explaination is lacking any meaningful information.
Requirement 4: Organization	Documentation is easy to navigate, with a table of contents, section headings, and consistent formating.	Documentation is managable to navigate, may be missing a table of contents, section headings, or consistent formatting.	Documentation is difficult to parse. Lacks table of contents, headings, and formatting.
Requirement 5: Regularly Updated	Documentation is updated in parallel with changes to functionality.	Documentation is updated frequently, not as often as the project however.	Documentation is rarely or never updated.
	Project Management		
	Project Board		
Requirement 1: Public Access	It's easy for anyone looking for the project board to find it. Within a web search and 1-2 clicks	The project board is challenging to find, linked from a few places in the project, but requires looking for it.	Project board is difficult to find, may only be linked in one place,
Requirement 2: Public Visibility	Anyone who wants to post a task can and all archives of past tasks are easy to find. All tasks are transparently dealt with.	Project board may require a log in and archives may or may not exist. Some tasks may not be announced pubicly.	Project board is inaccessible to those outside the project. All tasks made and completed internally.
Requirement 3: Frequent Use	The community is active, tasks dealt with as they come and questions are answered quickly and politely.	Community is somewhat active, posts infrequently, questions are eventually answered.	Community is seldom active. Questions are rarely answered.
Requirement 4: Organization	Tasks are organized into categories that make sense and reflect the state the task is in. (i.e. Backlog, in-progress, done)	Tasks are somewhat organized, but the categorization is too general to give a sense of where the task is. (i.e. having only to-do and done)	Tasks aren't organized in a meaningful way.
Requirement 5: Understandable tasks	Tasks have a clear goal and method of completion, written in a clear manner.	Tasks have a goal and method of completion, but there may be a few information gaps.	Tasks have no measurable goal, no guiding methods, and written poorly
Requirement 6: Relevant information available	Any external dependencies and information that can't fit in the task itself is linked to within the task.	External dependencies and information is stated, but may not be linked to.	External dependencies and information is missing.
	Project Discussion		
Requirement 1: Public Visibility	It's easy for anyone looking for the project discussion board to find it. Within a web search and 1-2 clicks	Discussion board is challenging to find, linked from a few places in the project, but requires looking for it.	Discussion board is difficult to find, may only be linked in one place,
Requirement 2: Public Communication	Anyone who wants to post can and all archives of past chats are easy to find. All anouncements and decisions are made in the open.	Discussion board may require a log in and archives may or may not exist. Some decisions may not be announced in the chat.	Discussion board is inaccessible to those outside the project. All decisions made internally.
Requirement 3: Frequent Use	The community is active, posts are made daily and questions are answered quickly and politely.	Community is somewhat active, posts infrequently, questions are eventually answered.	Community is seldom active. Questions are rarely answered.
Requirement 4: Use Cases Addressed	The chat has seperated and clear places for both users and developers. While there is a general chat, there are seperate, specific, and clearly organized channels for both.	Chat has some seperated channels, but mostly done in one channel.	All discussion is done in one channel.
	Compartmentalizing of Tasks		
Requirement 1: Ease of Comprehension	Tasks are organized and worded in a sensible way, anyone can get a good idea what the goal and methods of the task are.	Tasks are organized and worded in a somewhat comprehensible way. Most will understand with a bit of patience.	Tasks are organize and written in an indecipherable way or are missing altogether.
Requirement 2: Actionable	Tasks have a clear directive and method of completion.	Tasks have a general directive and some methods included, but not completely.	Tasks are lacking in directive and methodology.
Requirement 3: Defined Goals	Tasks have a defined endpoint when they can be classified as finished. Something definitive is accomplished by their completion.	Tasks have an endpoint, but it isn't entirely clear or is not stated.	Tasks are open ended and vague. They accomplish nothing upon completion.
Requirement 4: Prioritization Method	There is a way to set how pressing the issue / bug is and how quickly it needs addressed.	Prioritization categories are present, but lacking.	Prioritization methods are missing.
Requirement 5: Introductory Tasks	Have a set of tasks that beginners can do. Clearly label them as good beginner tasks.	Have a set of tasks that beginners can do, may not be clearly labelled.	No tasks sectioned for beginners.
Requirement 6: Ease of Responsibility	It's easy for anyone to take and delegate responsibility for that task and easy for other to see who has claimed that task.	Responsibility is limited, but still somewhat accessible. Delegation may not be possible and it may not be easy to see who has taken the task.	Responsibility is only handled by people with certain priviledges, and can't be assigned or taken on externally.

	Continuous Integration and Health Checks		
	Testing		
Requirement 1: Business Logic	Business logic is always unit tested.	Most of the business logic is unit tested, but not all of it.	Minority or none of the business logic is unit tested.
Requirement 2: Functional Tests	End to end test of functionality included with the unit tests, covers all aspects of user functionality.	End to end test of functionality exists, but doesn't cover every feature and use of the software.	End to end test of functionality is minimal or entirely absent.
Requirement 3: Run in CI and Locally	Unit tests automatically run in CI, but there's documentation for how to run the tests locally.	Unit tests run in CI, but there may not be extensive documentation on how to run those tests locally.	Unit tests are not implementated in CI and no documentation for running locally.
Requirement 4: Utilizes Code Coverage Tool	Have a code coverage tool implemented into the testing.	Code coverage tool is implemented but only for certain parts of the project.	There is no code coverage tool implemented.
Requirement 5: Efficiently Run	The tests run in an acceptable amount of time and in a reasonably optimized way.	Tests run in an average time.	Tests are poorly optimized and take inappropriate amounts of time.
	Code Base Health and Maintainability Level		
Requirement 1: Not a Mono-Repository	Code is seperated into appropriately segmented repositories.	Code base is separated into some separate repositories but repositories are quite large, has a large variety of functionality grouped together in a disorganized way.	All code is shoved into one repository. Repository serves a large variety of functionality which would be better set up as separate projects working together.
Requirement 2: Sensible Architecture	The structure of the code is obvious from first viewing and with explaination.	The code structure may be overbearing at first, but has an explaination that helps developers understand.	The code structure is obtuse and not explained.
Requirement 3: Style Guidelines	Code follows a set of style guidelines that are laid out and enforced by the CI	Code mostly follows a guidelines, but there may be places where it's violated.	Code doesn't follow any sense of guidelines or standards.
Requirement 4: Pass a Truck/Bus/Raptor Test	Codebase passes the git by a truck/bus/raptor/etc test		Codebase doesn't pass the git by a truck/bus/raptor/etc test
Requirement 5: Hacks Kept to a Minimum	Code is self-documented and easily understandble, but code outside of the guidelines is the exception, not the rule. Hacks are marked as so, infrequently used, and explained with inline commenting.	There is a significant amount of hacks but are marked as so and have inline comments marking and explaining them.	Majority of the code is outside the guidelines, no way to measure how much or where this code is.
	CI Testing Integration		
Requirement 1: CI is easy to access independently	CI can run in a simple command.	CI can run in a few complicated commands, but is accessible.	CI is cumbersome to run, taking several steps and a long time to simply set up.
Requirement 2: Matches required formatting	CI makes sure that code follows the required format and guidelines.	CI has a few guidelines implemented, but may not have all of them or may be too lenient on enforcement.	CI has no guidelines implemented.
Requirement 3: Integrated directly with source control	CI is integrated with source control, can immediately do a pull request or commit after a successful test.	CI is somewhat integrated with source control, but may need a few time consuming steps to work properly.	CI is completely divorced from source control.
Requirement 4: Runs efficiently	CI runs in a reasonable amount of time.	CI runs in an average amount of time, but not optimized.	CI takes way too long to execute.
Requirement 5: Quality of Output	CI gives the developer useful feedback, any issues encountered are explained and they can see where they made a mistake.	CI gives the developer some feedback, some issues explained, some just stated without giving the developer a guide to how to resolve.	CI gives very basic feedback, maybe only a letter grade.
	Workflow		
	Pull Request Workflow		
Requirement 1: Clear format	A clear format is defined, frequently used, and easy to follow for any outsider to make a pull request.	A format is defined, used occassionally, and is easy to follow for a pull request.	No format is defined, or it's rarely used.
Requirement 2: Peer Reviews	Every pull request is reviewed by a substantial number of people before it is merged with the project.	Most pull requests get reviewed, but may not have only a single reviewer or lazy reviewing process.	Nothing is reviewed.
Requirement 3: Regular Use in Development	Pull requests are used by developers except in the case of an emergency hotfix.	Pull requests are the most common method, but some developers still push straight to master.	Pull requests are not used often.
	Community Outreach		
	Dev Blog		
Requirement 1: Detailed Announcements	All major announcements and releases are on the blog along with regular updates about progress on the project.	Most major announcements and releases are on the blog, updates are semi-frequent.	Most announcements are entirely ignored and no updates are posted in between.
Requirement 2: Archive of Posts	It's easy for anyone to check the post history to find the old posts and read about announcements and releases of the past.	The archive may be flushed after a certain point or certain posts are never archived, but the majority of the information is available.	No archive exists, or the one that does doesn't have any posts in it.
Requirement 3: Demonstrates General Direction of the Project	The announcements, updates, and releases all breifly detail why they add what they add, giving an overall impression of direction.	The detail of why may be missing from a few posts, but the blog still gives enough information to form a direction of the project.	No direction or reasoning behind decisions is given.

Requirement 4: Explains the Current Goals of the Project	The most recent posts detail the overarching goals of the project and which ones have been met since last announcement.	Some goals may be listed in the most recent posts, but the mention of the goals already met may be breif or unclear.	Goals aren't listed or mentioned in the blog.
	Social Media		
Requirement 1: Announcements Posted	Either short form of the blog posts, or linking to the blog post announcements, all anouncements are posted on the social media platform.	Most major announcements are mentioned on the social media account.	No major announcements are posted on social media.
Requirement 2: Communicates with Users	When people engage with the account, someone operating the account responds when necessary.	Replies to users are infrequent, but they happen.	The account is silent on communicating with users.
Requirement 3: Regularly Updated	Posts are regular and communicate that the project is active.	Posts aren't everyday, but still enough to tell the account is active.	Posts are not made regularly.
Requirement 4: Uses a Large Scale Social Media Website	The social media platform is a large one, (i.e. twitter)	The social media platform of choice may be a bit more niche.	The social media platform is entirely obscure.
	Upstream*		
Requirement 1: Offer Support of Upstream Development	Upstream development is supplemented by developers on the team.	Some upstream development is done but it isn't actively encouraged by the organization.	No upstream development is attempted.
Requirement 2: Contributes Feedback and Bugs to Upstream	Any bugs, usability problems, and issues encountered with the upstream software are reported by the team.	Bugs, usability problems, and issues are mostly reported, but some are simply dealt with internally.	No problems encountered are reported to the upstream project.
Requirement 3: Feedback loops between groups	Upstream gives feedback, implementation advice, and development assistance to the project as th porject gives those resources in turn.	Upstream is somewhat involved in the project, and the project is somewhat involved in the upstream, but the relationship isn't developed.	There is no identifiable loop between the upstream and the project.
Requirement 4: Identifiable Pathway for Contribution	A clear way to contribute to the Upstream and project exists.	Contributing to the upstream and project has a guide of some kind, but it's fairly barebone.	No way to identify contributions to the upstream and project.
	Website		
Requirement 1: Explains the Project	Gives a detailed, easy to understand and well written explaination of the project and states that it is open source.	Gives a satisfactory summary of the project and states that it is open source.	Doesn't detail the project in a significant way and does not state that it is open source.
Requirement 2: Leads Developers to Get Involved	Website has a clear section or link labelled for developers (i.e. a navigation tab that says "Get Involved")	Website has a section for developers, but it may not be immediately apparent where it is.	Website doesn't have a section for developers or it is near impossible to find.
Requirement 3: Leads Users to Installation and Guides	Website has a clear path to installation and a section for users.	Website has an installation section, but may not be upfront about it.	Website has no installation resources.
Requirement 4: Links to Resources (github, documentation, social media)	Links to other resources that can be used by both users and developers are immediately available and apparent	Links to other resources exist, but there aren't many listed.	No links to any external resources exist on the website.
Requirement 5: Presentation	The website is professionally designed and isn't prone to navigation or design pitfalls.	The website has an okay sense of design and navigation, not perfect, but it works.	The design of this website is comparable to 1998 standards.
Requirement 6: State the License	The website mentions the license explicitly on the front page.	Website has the license but isn't on the front page.	Website is lacking detail of the project's open source license