Assignment 8: Individual Requirements Analysis

Mahmoud Thabit

Pawprint: Mlt446

Project name: Sprint 1

Front-end.

1. Mock Website (could not create a wireframe)

![A screenshot of a social media post

Description automatically generated]()

1. The APIs to be used
   1. Experimental - Top Committers (Repo Group) (/repo-groups/:repo\_group\_id/top-committers)
   2. Average Issue Resolution Time (Repo Group) (/repo-groups/:repo\_group\_id/avgerage-issue-resolution-time)
   3. Reviews (Repo) (/repo-groups/:repo\_group\_id/repos/:repo\_id/reviews)
   4. Aggregate Summary (Repo Group) (/repo-groups/:repo\_group\_id/aggregate-summary)
2. The use case will be the same as the individual ones, the three

* The first feature is to display the top committers per group and the percentage of contributions each made.
* The second feature is to display the average time to resolve an issue per group.
* The third feature is to display the number of pull requester per repo group over the years.

1. AUGUR: augur\_data.repo, augur\_data.repo\_info, augur\_data.repo\_badging

* Introduction:

1. **Purpose**: This document is intended to fully specify the requirements and details of Sprint semester project. This document will describe the external and internal behavers of the application. It will also specify the functional and non-functional requirements, constraints and other factors for the project.
2. **Intended audience**: Programs that are looking to join an open-source project and evaluate how healthy a project is before joining.
3. **Scope**: the scope of the project will be limited to the projects that Augur has, the benefit of this project it will make evaluating the health of a project really simple to evaluate a project based on the number of contribution, how evenly the contributions are, and pull requests acceptance rate per group. The scope of this project will be limited to the APIs that are already available and by the Augur data that you get from the API calls.
4. **Assumptions**:

* All the APIs have been created and work perfectly.
* We will have access to the data that are needed for the project from the APIs that are already in augur.
* The augur project updates and maintained correctly.
* Software product overview

1. **System scope**: the project will be available to all devices through the web application. The web application will cover the mobile app too as the expected mobile users are a small segment of the users.
2. **System architecture**:

* The external architecture will consist of
* A webserver to host the web page, to communicate with the user and the augur server.
* Another webserver to host augur with the database (this already exists in our case).
* A web application for the user to communicate with the web page.

1. **Feature overview**:

* The first feature is to display the top committers per group and the percentage of contributions each made.
* The second feature is to display the average time to resolve an issue per group.
* The third feature is to display the number of pull requester per repo group over the years.
* System use

1. **Actor survey**:

* **Developers**: The developers that are going to maintain and write code to be used on the system.
* **Admin**: The admin is a senior developer that is responsible for deploying any changes or maintenance to the web site.
* **End Users/developer**: the developer that is using the web site to compare and figure out which open source project he/she should join based on the health of the open-source project.

1. **System interface**: there will only be on system interface for all devices which is the web page. The web page should be simple not to overwhelm the use and confuse them. There will only be 3 visualization to keep it simple.

![A screenshot of a social media post

Description automatically generated]()

* System requirement

1. Use Cases:
   1. **Use case 1**: Getting the top committers and the percentage of contribution per repo group

A close up of a map

Description automatically generated

* 1. **Use case 2**: Getting the average time to resolve an issue per group

A close up of a mans face

Description automatically generated

* 1. **Use case 3**: Number of pull requests per repo group over the years

A close up of a mans face

Description automatically generated

1. Functional specification
   1. Web page must only display data only after the data has be fully received
   2. The server shall only communicate with the user web browser and the user web browser shall communicate with the augur server for API calls
   3. User should be only able to choose from three different data representations.
   4. Repository groups should be hard coded in the web page
2. Non-Functional specification
   1. This project should be scalable for future additions. This project should have a clear templet to show how data should be displayed in a simple self-explanatory way.
   2. There should be no more than two API calls to get the data to create one visualization, by keeping the API calls to a minimum the web page will be faster and more data efficient
   3. No loges will be collected on users to keep tack of their activities. No cookies or session will be used in the web page to keep track of users.

* Design constraints:

1. this system should be compatible with the augur open source project
2. this system will use the only APIs that have already been developed within the Augur open source project
3. the web page will be built using JavaScript
4. the system will be hosted on an amazon AWS server
5. the web page will run on all major web browsers (Chrome, Firefox, ... etc.)

* Purchased components

1. AWS server to host the web page

* Interface: the primary interface for the user will be web browser. This means that the traffic will be direct to an application server. The web application will externally call the server with API request to the main augur server to get the data. The API request will be done using JavaScript.

1. User interface: the web applications will be open to any user with out the need for authentication or authorization. All the API calls will be done on the user’s web browser to avoid his network overhead on the AWS server. The speed and latency of the user’s internet speed will greatly impact the speed at which the data will load.