Individual Requirements Analysis

ASSIGNMENT 9

Lauren Cochran

- ## Introduction:
  - ### Purpose
    - CHAOSS's goals are to:
      - Establish standard implementation-agnostic metrics for measuring community activity, contributions, and health
      - Produce integrated open source software for analyzing software community development
      - Build reproducible project health reports/containers
    - CHAOSS implements different software like Augur. Augur aims to build human centered open source software health metrics. This software focuses on making sense of the data through: comparisons, a time fundamental dimension in all metrics from the start, all data driving visualization are downloadable as .csv, and all visualizations downloadable as .svgs. It is a prototyped implementation of CHAOSS.
  - ### Scope
    - This software is free to use to use and download, so anyone could use it, and anyone could potentially contribute to the software. However, it is geared towards individuals who want to look at the health of open source software.
- ## Software product overview
  - ### System Architecture:
    - Augur allows users to look at the different repo groups they are in and see the data behind the health and sustainability of that repo in easy, human readable formats, which can be displayed on Vue. For the backend, you can access Augur through the command-line with Python and GitHub. Overall, Augur pulls information from a GitHub repository, stores that information in a database, and then displays that information to the user in an easy to read format.
  - ### Feature Overview:
    - Link GitHub
      This allows a user to link their specific repository to the software, so that analytics can be generated to the frontend while storing the content in the backend.
    - Look at the Overview
      This lets user look at a specific repo group's analytics, displaying them in charts and graphs so it is easy to read. With each graph and chart, you can download it to save.
    - Change Repo Group
      This allows you to have a dropdown option to change the repo you're focused on, so then new analytics will show up.

- ## System Use, including an actor survey
  - ### Actor Survey:
    - Workers:
      The workers are there to maintain the system since it's not guaranteed that anyone else would with it being open source. They are also there to provide help for users who are having trouble understanding the software.
    - User:
      The users are the one dealing with the front end, so they can acess all the functionality of Augur, but they wouldn't be able to change the content of it unless their role changed to Contributor. They are accessing the software for the purpose of seeing the analytics.
    - Contributors:
      Contributors are the ones who can add to the project if they see something in the software to fix. Of course, their changes would have to be accepted by the Administrator because the changes were implemented.
    - Administrator:
      The administrator is the one in charge of pushing all the commits to the master branch. The administrator would be in charge of making sure the code works properly and that every worker is pulling their weight in work.
- ## System Requirements (including 2 use cases, a system functional specification, and a list of non-functional requirements)
  - ### Use- Case: Viewing the Analytics (health and stability)
    - Actors: Users
    - Description: Here, a user has selected a specific repo from GitHub that they would like to see the data on. By clicking a button that says Overview, it will generate a general html page of the analytics.
    - Flow of events: A user wants to see the health of a specific repo, so they open up Augur and navigate to the repo they want to see. Once clicking on it, the analytics will be loaded and there will be an option to then see the data present in Risk Metrics instead.
    - Exception: if no data is in the backend, then nothing will result from this.
  - ### Use-Case: Update information in database
    - Actors: Workers
    - Description: there will be a piece of software that will go to the repo and pull the information needed to output in order for the health to be current

        - Flow of events: The worker will set a specific time frame of which the information will be pulled from GitHub. Once the info if pulled, it will update the database which will then trigger the front end to be updated with the current information as well.
        - Exception: if a repo no longer is being committed to, it would not have to be updated
    - System Function Specification:
        - The information pulled from GitHub should match what is stored in the database
        - When web browsers update, everything should still function correctly or else be taken down for maintenance.
        - Information should be updated
    - Non-functional Requirements:
        - Should be able to handle multiple users on it at a time
        - It should be able to handle many repos being stored in the database
        - Not all of the data should be stored on the same server
        - Interface must be responsive

- ## Design Constraints (at least 5)
    - It should be able to run on every web browser
    - The software should be supported by all Operating Systems, so that you're not limited by the Operating System.
    - It should have error checking built in so that if the database is down, for example, the website should display that information.
    - It should be responsive, so that if it is having trouble loading or reaching a GitHub repo, it would display a loading message.
    - It should follow along with the Model View Controller method, so that if you access the front-end, it should not directly manipulate or touch the data in the backend.

- ## Purchased Components (at least 1)
    - A webserver to host and display the information, which can be around $20 a month, starting conservatively.
    - A domain for the website, which can be around $20 a year
    - A computer with internet access, which can cost thousands of dollars.

- ## Interfaces (at least 1)
    - The interface would include:
        - Home page: this would hold a list of all the repo groups that Augur has on file; this would be responsive so that you could click on the groups and it would take you to the group page.
        - Group Page: will display the groups of repos in the database. Once you click on the group, it will take you to the Repo page.

- Repo Page: this displays the specific repos in the group, showing the total commits and issues along with a URL link to GitHub.
- Analytic Page: once you click on the specific GitHub link, it will display the health and sustainability of that repo group in a chart or graph that can be saved.