Software Requirements Specification

Project Name: Augur

Table of Contents

1.) Introduction

a.) Purpose

This document is intended to lay out the details of the open-source development management software known as Augur. This software requirements specification (SRS) will cover the external behavior of the software, as well as the internal systems at work inside of Augur itself. This should also go over the non-functional requirements, design constraints, and other things to provide a complete and comprehensive description of this software.

b.) Scope

Augur is an application to be built to be able to determine the “health” (which is calculated off of a variety of types of information like number of contributors or number of commits) of the different open-source projects that are in the system. Combining several of the different information of the “health” of the different open source projects should fulfill clients’ needs to be able to view an easily viewable and compact summary of the different projects that they are managing or a part of. This should take away some of the work for compiling information that clients want automated.

c.) Assumptions and Dependencies

- Given the AGILE nature of an imaginary development process, it is entirely possible, that the development of this software may change based on a variety of different possibilities, like user feedback change, etc…
- Github, but more specifically, the information provided by Github’s repository information will be used by Augur to be able to compile and display the information about a user’s repository health.
- Augur should be entirely web-based. So, any device should (and must) have some kind of access to the internet, and a web-browser to be able to use this software.
- Upon creation of an account with augur, administrative users should be able (and want) to add in which repositories they want to be able to view information about.
- There is not a messaging system included in this web-app since there are plenty of messaging systems that are already used by employers.

2.) Software Product Overview

This section’s purpose is to provide an overview of Augur as a software product. This provides an overview of how the system works from all perspectives. This section includes the system scope, the system architecture, and an overview of the system features that make up Augur.

a.) System Scope

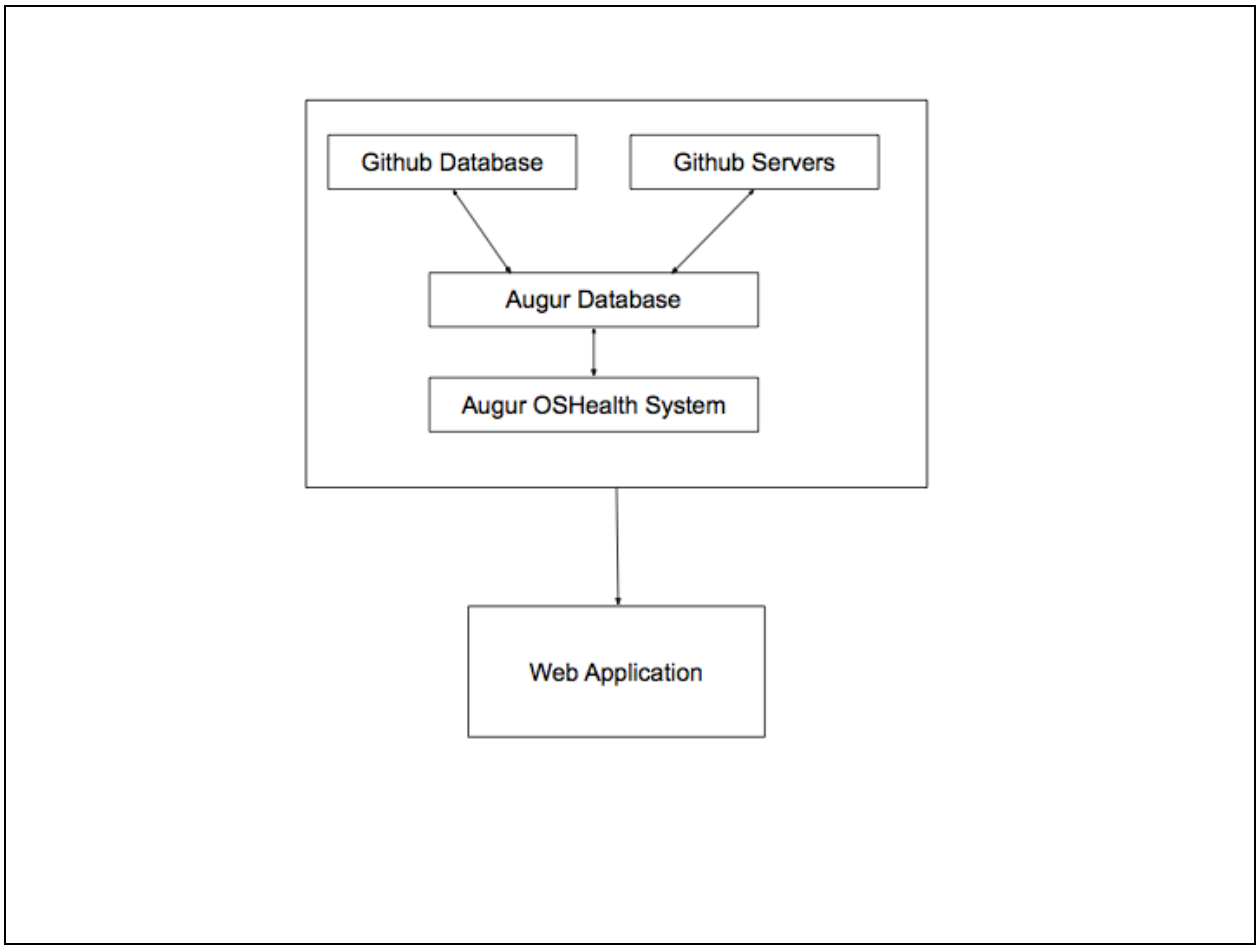
This section is intended to provide information about the physical (architectural) system scope of Augur. Augur is an integrated system with Github to access the information about the health of a persons’ different repositories and present that in an easily manageable format.

Augur will be available on any device with an internet connection, and a browser. Since this is browser-based, both mobile and desktop versions of Augur should have all of the functions that are available to either one. This should allow open-source software developers to overview the health of their systems either on the go, or at home with relative ease.

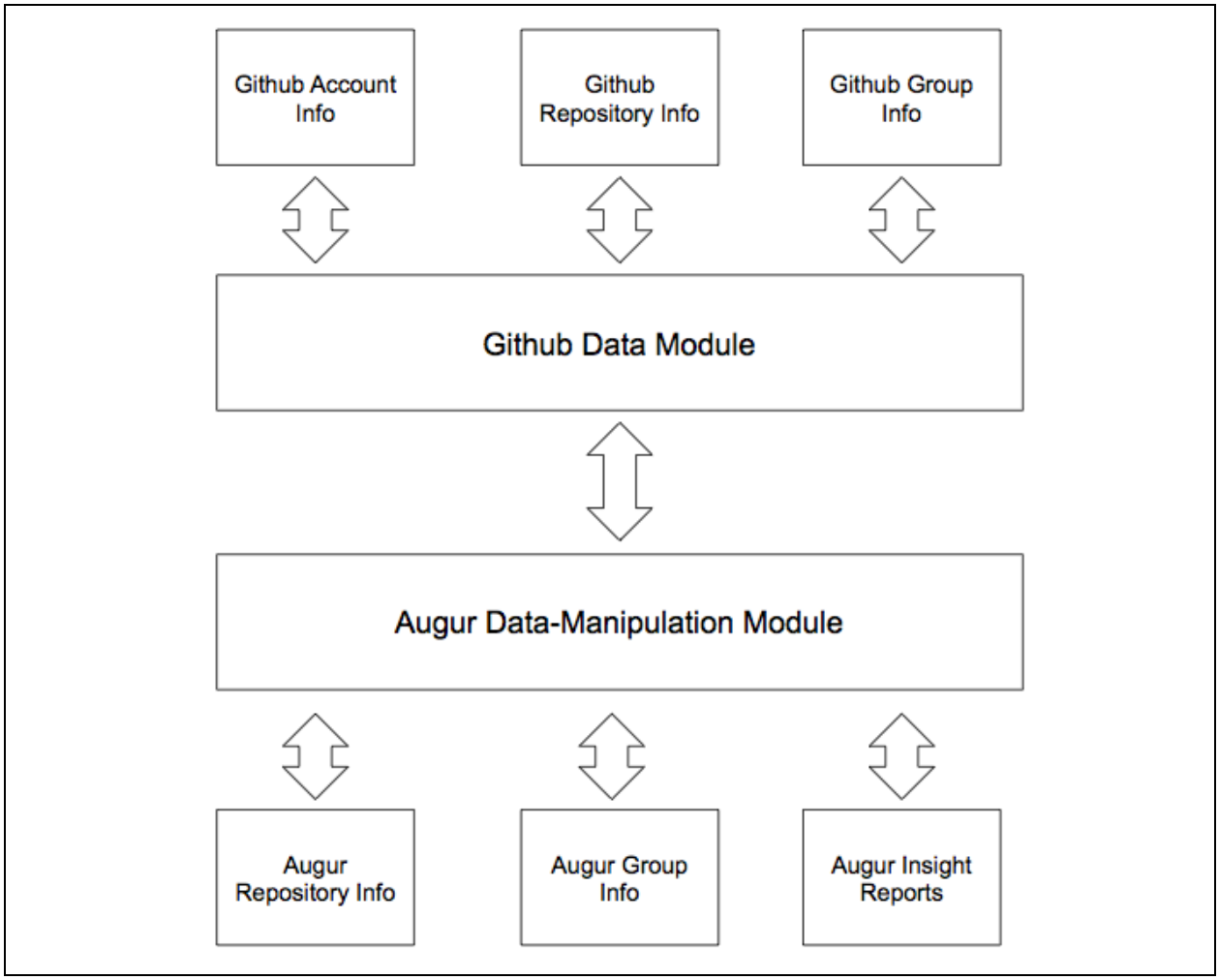
b.) System Architecture

This section defines the internal and external system architecture of Augur.

External-



Internal-



c.) Feature Overview

This section provides a brief overview of the system features available through Augur. They are divided into Overall, Group, Repo, and Insight Functions.

- Overall Function-
    - Create New User- This feature creates a new user for the augur system and allows them to attach their Github account to link their repositories.
    - Delete User Function- This feature deletes a current user in the system in order to give the ability to remove their information from the site.
    - Repository/Repository Group Search- This feature allows the user to look up their repository group/repositories to be able to view them.
    - Store Repo- This feature allows the user to store a repo that they are a part of to be able to use currently stored _____ functions below.
- Group Functions-
    - Create/delete repo group- Creates a repository group, and allows you to add/delete repositories to/from that group.

    - Add repo to group repo- Adds a repository to as a group repository, and allows the people to view them if they are the manager, or a part of the group
    - Currently Stored Groups- Presents information about a user's currently stored repo groups, including the name, description, website, last modification, and type.
    - Stored Groups Sort- This allows the user the ability to sort by the previously explained statistics from the currently stored groups function.
    - Add/remove user to/from group- This allows administrators of repository groups to add or delete other users to their groups.
- Repo Functions-
    - Currently Stored Repos- Presents information about a user's currently stored repo, including the URL, Repo Group Name, Total Commit Count, and Total Issue Count.
    - Stored Repos Sort- This allows the user the ability to sort by the previously explained statistics from the currently stored repos function.
- Insights-
    - Data Projection- This function allows the user to view information about their repos/groups with easy-to-view projects (graphs, summaries, etc…).
    - Switch Projection Function- This function allows the user to swap the projection that is currently being used for a specific projection (e.g- switching from a line graph to a bar graph).

3.) System use

This section is used to give an overview of what features are available to the user, as well as the different "roles" the system provides.

a.) Actor Survey-

Manager-
The manager is responsible for creation/deletion of repository groups. Their job is to make it easier for them, and the people that are working within their groups to easily view the health of the repositories/projects that they are working on.

System Features-
- Create/Delete Repo Group
- Add/Remove User from Group
- Add Repo to Group Repo
- (+ any function available to Users)

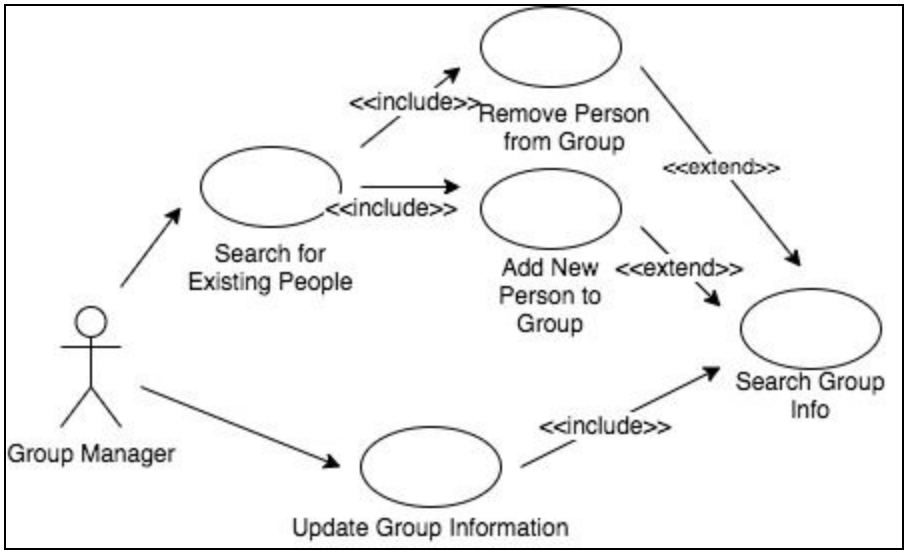
User-
User is the default status of a user for the Augur system. The user status allows them to be able to view the info of their repositories they have created, as well as the repositories of groups that an administrator has associated them with.

System Features-

- Create New User
- Delete User Function
- Repository/Repository Group Search
- Store Repo
- Currently Stored Repos
- Stored Repos Sort
- Data Projection
- Switch Projection Function

4.) System Requirements-

a.) Use Case 1- Input and Manage Repository Groups



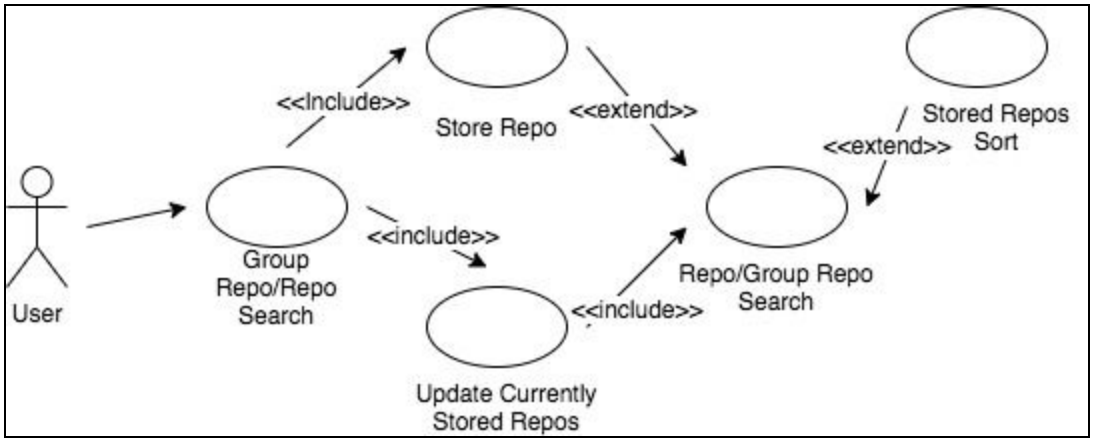
(Note: Actor = Group Manager)

Use-case name	UC1 Add People to Group Repository
System/Subsystem	Augur (repo groups)
Actors	Manager
Brief Description	This use case describes how a group-repository manager adds/deletes people to/from a group, and is able to view them.
Basic Flow of Events	Basic flow begins when a manager needs to add a user that wishes to work on a group repo. The manager selects add/remove users on the group repo screen: 1.) The system displays a user-search screen to be able to easily

	find/select the people/person that the manager wants to add to a group by asking for the desired username. 2.) As the manager types, the screen is updated to show the potential matches to their search, and when the user is found, the manager has the ability to select them. 3.) When the user is selected, the manager is prompted to verify that it is indeed the correct user. 4.) The manager is then prompted to select the group-repository that the new user is to be added to. (if the manager has a lot of repositories, they can search through their repositories in a similar way to the user search mentioned earlier) 5.) When the group is selected, the system will prompt the manager to confirm that the correct user is being added to the proper group repo. 6.) The system will then return to the group-repo screen.
Alternative Flow of Events	● searched person does not exist -> the system will tell the manager so, and return to the group repo screen ● failure to add/delete person -> the system will store and revert to its previous state to before the new person was searched ● update group information fails -> the system would revert itself to before when the person was searched
Special Requirements	● Usability- The system will present a user search and group search that is easy to use for anyone of a high-school education. It would require very little training to use Augur. Also, any errors will be displayed to the manager in a clear, explanative way ● Performance- The system should not take more than 5 seconds to search the user/groups that the manager enters when they have found the desired user/group ● Reliability- The system will store its state before a user is searched for. If some error with the system occurs, the system will revert to that state.
Pre-conditions	The manager must be identified, authenticated, and logged into the system and at the repo group screen. Also, the manager must be a part of at least one group.
Post-Conditions	- If the system fails, then the proper error message will be displayed, and the proper error protocol (mentioned earlier) will be followed - Augur's database must store the user as a part of the correct group now - Confirmation message of addition to new group will be sent to the added User's email

Extension Points	Search Group Repository Info

b.) Use Case 2- Repository Creation/Management



Use-case Name	UC2 Repository Creation/Management
System/Subsystem	Augur (repos)
Actors	User
Brief Description	This use-case describes how a user adds/manages their repositories to Augur.
Basic Flow of Events	Basic flow begins when a user has the need to add a repository to augur for whatever purpose. The user selects “add repository” from the stored repositories screen. 1.) The user is prompted with a repository search menu that, as they type, updates the search field with relevant repositories 2.) The user selects the desired repository when it appears on the screen 3.) The user is then prompted to confirm that the repository that they have selected is indeed the correct one. 4.) (optional) The user can sort their repo/group repo search by date added, so that they can easily view the new repository
Alternative Flow of Events	● The repository does not exist -> the system will tell the user so, and then return to the repositories screen. ● Failure to add repository -> the system will tell the user so, and then revert to a saved state from before the new repository was searched for. ● Update Currently Stored Repositories -> the system will tell the user so, and then revert to a saved state from before the new repository was searched for.

Special Requirements	● Usability- The system will present a repository/repository group search that is easy for anyone of a high-school education to use. It would require very little training to use Augur. Also, any error will be displayed to the User in a clear, explanative way ● Performance- The system should not take more than 5 seconds to add the selected repository to the stored repositories, and return to the repositories screen ● Reliability- The system will store its state before a repository is searched for. If some error with the system occurs, the system will revert to that state.
Pre-Conditions	The manager must be identified, authenticated, and logged into the system and at the repo group screen.
Post-Conditions	- If the system fails, then the proper error message will be displayed, and the proper error protocol (mentioned earlier) will be followed - Augur's database must store the repository as a part of the stored repositories now - Confirmation message of addition to new group will be sent to the added User's email
Extension Points	Repo/Group Repo Search

c.) System Functional Specification-

Client Terminals- User Interface Functions

Since Augur is a web application, the system architecture will be client-server. All applications will normally execute on the client's hardware with data requests to Github through Augur facilitated through the server. This will be the same for both mobile, and desktop based versions since both are run from browser.

CT1: User login and authorizations; main menu screen
CT2: Create New User
CT3: Add/Remove user to/from group
CT4: Repo/Repo Group Search that returns search for items
CT5: Insight Screen that shows data projections
CT6: Data Projection of data from repositories/repository groups
CT7: Switch Projection type of a specific projection
CT8: Repos Screen that shows currently stored repos
CT9: Store Repo that stores a new, searched-for repository
CT10: Stored Repos Sort that sorts the repositories by specified statistics
CT11: Groups Screen that shows currently stored group repos

CT12: Create/Delete Repo Group
CT13: Add Repo to Group Repo that adds a repo to a group repo
CT14: Stored Group Repos Sort that sorts the group repositories by specified statistics

d.) Non-functional Requirements

- Users should be required to read the product's manual to understand all features (included in the website).
- Users should be familiar with operating a web browser in order to direct themselves to Augur.
- Augur's aim is to be as reliable as possible, but users should fill out crash reports in the event of a system-crash, which will be taken into account A.S.A.P.
- The system does require a github account, which must be provided on account creation. You can see how to do create a github account here: https://github.com/join
- The system has a minimum requirement of 0 repositories/group repositories, which means you technically do not have to have associated repositories to use this web application.
- All system responses should occur within 30 seconds.
- Augur will be available for desktop and mobile browsers.
- Operating systems:
    - Windows
    - Macintosh
    - Android
    - Blackberry
    - Windows Phone
    - iPhone/iOS

5.) Design Constraints-

- The Augur system will be a web application with access via standard secure login procedures
- Augur will be available on all major web browsers.
- The differences between the different major web browsers should be minimal, and not require external setup from the user.
- Every function that is supported on the desktop version should also be supported on the desktop version.
- Differences between mobile and desktop websites should be minimal, but will be changed slightly for design and layout purposes.

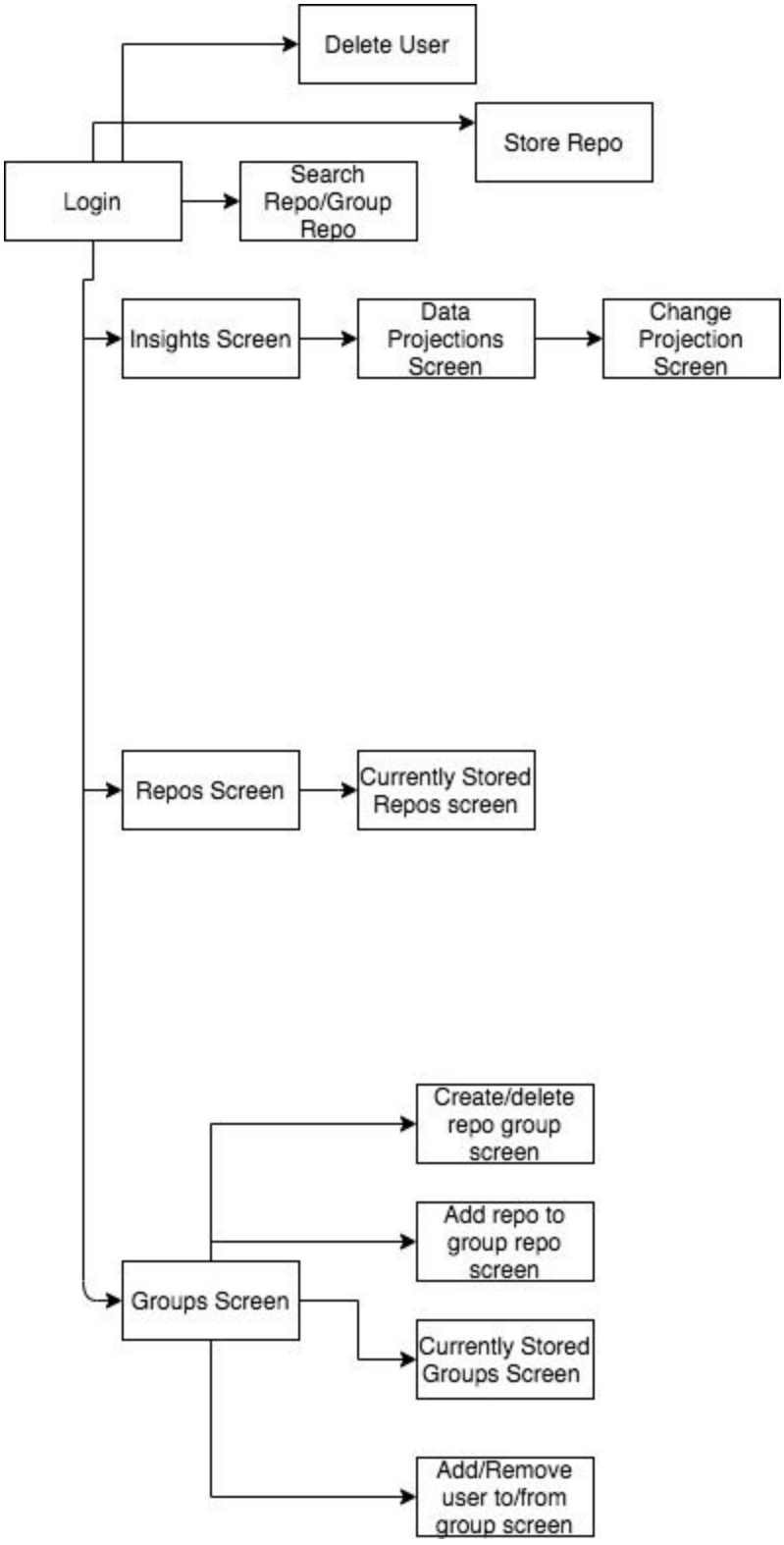
6.) Purchased Components-

Amazon AWS server space - $20-$30/month

Domain Name Auger© - $10-$20/year

7.) Purchased Components-

a.) User Interface-
Screen Flow Diagram: ↓↓↓↓


Delete User
Store Repo
Login
Search Repo/Group Repo
Insights Screen
Data Projections Screen
Change Projection Screen
Repos Screen
Currently Stored Repos screen
Create/delete repo group screen
Add repo to group repo screen
Groups Screen
Currently Stored Groups Screen
Add/Remove user to/from group screen