David Hannan

Assignment 9: Requirements Analysis

Outline:

- Introduction
- Software product overview
- System Use
    - Actor Survey
- System requirements
    - Use cases
    - Functional specification
    - Non-functional requirements
- Design Constraints
- Purchased Components
- Interfaces

Introduction:

This document is intended to specify the requirements for the development of a system to analyze open source projects and make sustainability predictions based on this analysis. This document will provide an overview of the software, the system use and use cases, the system requirements, design constraints, a list of components to purchase, and a description of the interfaces required.

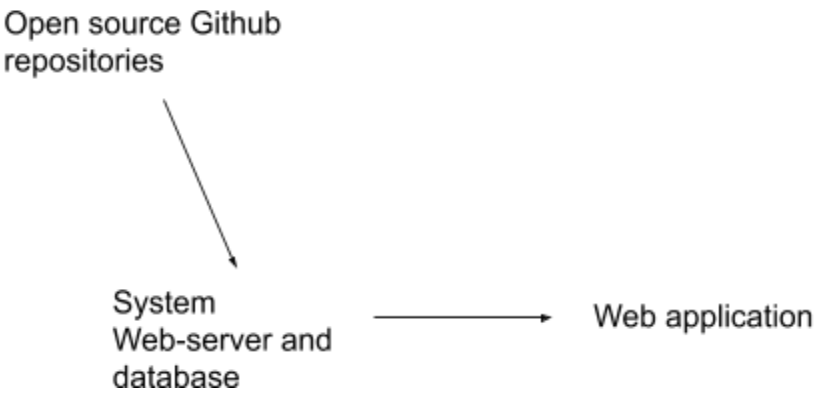
Software Product Overview:

With so many different open source projects being created and developed at varying levels of support and seriousness of intent, it can be difficult to make a decision of whether or not it would be a good idea to implement a given open source project into your project. The goal of this system is to make that decision easier.

Scope:

This system will be deployed as a web application for users that are looking at open source projects and deciding if implementing the projects would be a sustainable decision. The system will utilize information available from repositories on Github to do so.

System Architecture:



System Use:

Actor Survey:

System Administrator

The system administrator is responsible for maintaining the system. This includes maintaining data integrity and responding to problems that arise within the system.

System Features:

- Add/remove repositories
- View raw analysis data
- View system logs

Prospective Open Source User

The prospective open source user is the main target user of the system. This user should be able to search for/provide an open source project that is on Github and view/generate a variety of metrics in order to make the decision of whether to implement the project or not.

System Features:

- Search for repositories
- Provide a repository
- View project metrics

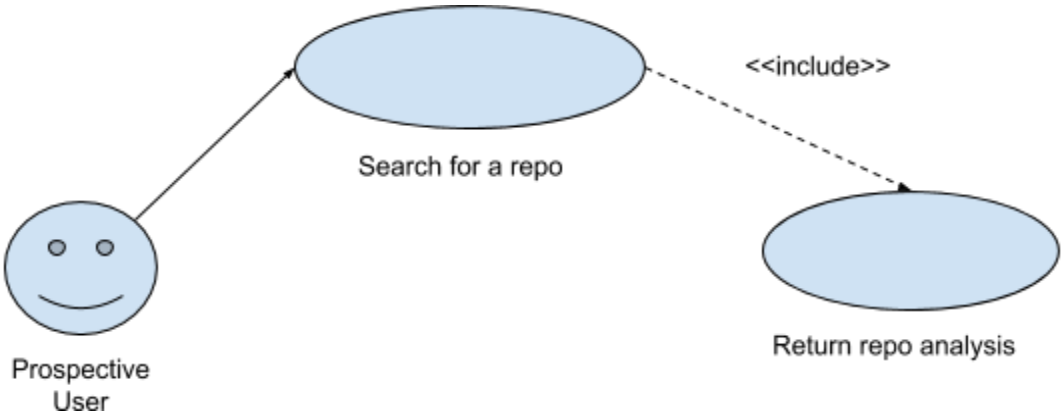
Open Source Project

The open source projects that are being analyzed are not interacting directly with this system, however the system would not have any data to analyze without them.

System Requirements:

Use Cases:

Use Case 1: User repository search

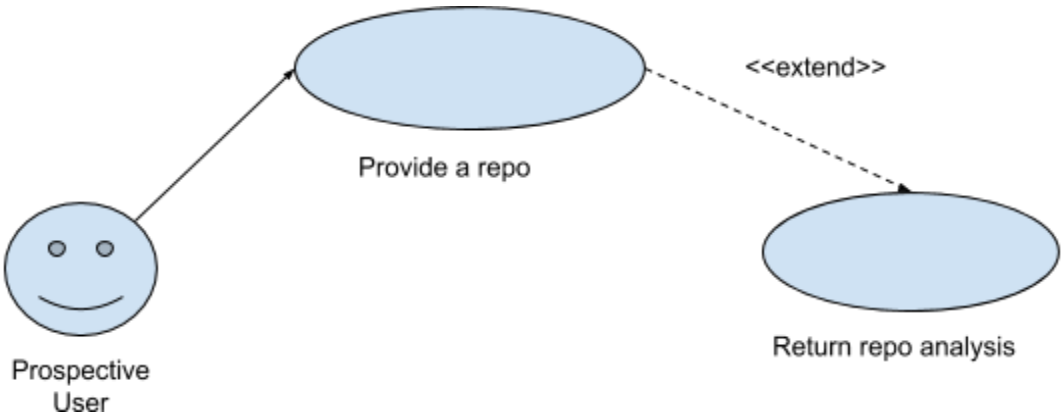


Use-Case name	UC1 User Repository Search
System	This system
Actors	Prospective User
Brief Description	This use case explains how a prospective user searches for an open source project repository and is returned with the analysis of said repository.
Basic Flow of Events	Basic flow begins when the prospective user

	accesses the web application and is prompted to search for the project they may want to use. The user then enters the name of the repository and is returned with a list of matching repositories. The user can then select the correct repository and view the analysis.
Alternative Event Flow	If the repository that the user searches for is not found, the user is prompted to utilize UC2, in which the user can provide the repository specifics to perform the analysis.
Special Requirements	<u>Usability:</u> The system should be usable by anyone with knowledge of open source projects. It should be intuitive to use and the analysis should be easily interpreted by the user. <u>Performance:</u> The system should have a responsive search in which results begin to show up as soon as the user begins typing. The analysis of the repository should take no longer than 30

	seconds. <u>Reliability:</u> The system will build a database of repository analyses that will be provided to the user when they select the respective repo. The data in this database will be updated on a regular basis. The database will be built upon under UC2.
Pre-conditions	The user must be on the website and the repository that they want to analyze must be present in the database.
Post-condition(s)	The user will be able to view the analysis.
Extension points	None.

Use case 2: User Provided Repository



Use-Case name	UC2 User Provided Repository
System	This system
Actors	Prospective User
Brief Description	This use case explains how a prospective user can provide an open source project repository and is returned with the analysis of said repository.
Basic Flow of Events	Basic flow begins when the prospective user has already searched for the project they may want to use and is told that it is not already in the database. The user is then prompted to provide a URL of the Github repo that the

	project is located at. The repository is then added to the database and an analysis is performed and returned to the user.
Alternative Event Flow	If the URL that the user provided is not valid they should be notified and re-prompted to enter it.
Special Requirements	Usability: The system should be usable by anyone with knowledge of open source projects. It should be intuitive to use and the analysis should be easily interpreted by the user. Performance: The analysis of the repository should take no longer than 30 seconds. Reliability: The provided URL should be checked for validity as soon as the user enters it and before an analysis is attempted. The repo can then be added to the database and the analysis can be performed and returned to the user.
Pre-conditions	The user must be on the website and the

	repository that they want to analyze must not be present in the database. The user must have a URL for the Github repository.
Post-condition(s)	The user will be able to view the analysis.
Extension points	Add repository from URL

System Functional Specification

This section provides an overview of the functional processes that are required for this system.

User Interface Functions

The system architecture implemented will be client-server since the system is a web application.

All requests will be handled by the server and results will be passed back to the user.

UI1: Search for available repositories

UI2: Select from available repositories

UI3: Provide a URL for a repository

UI4: View repository information and analysis

Server Application Modules

The following functionality will be supported by a server module. These functions will be performed as a result of actions made by the user with the exception of database updates which are performed on a regular basis regardless of user input.

Database Management:

DB1: Get data from repository and save it to the database

DB2: Pull new data from repositories every 24 hours

DB3: Perform preliminary analysis to optimise analysis performance and save results to database

Analysis Performance:

AP1: Get relevant data from database and perform necessary computations to produce desired metrics

AP2: Send metrics to user

Non-functional Requirements

Usability:

NFR-1: The systems interface should be highly intuitive to use.

NFR-2: Any user familiar with open source projects should be able to perform the analysis they desire

NFR-3: There should be a small description of each metric and what it says about the health of the project

Reliability:

NFR-4: The system should work as expected with any given valid repository.

NFR-5: The system should not go down for more than a day at a time.

NFR-6: The data projected in the analysis should be an accurate representation of the data that is available from the repository.

Performance:

NFR-7: The system should support 500 simultaneous users at minimum

NFR-8: The system should support a minimum of 10,000 available repositories

NFR-9: The system should support 500 simultaneous analysis performances minimum

NFR-10: The system should be able to perform each analysis in 30 seconds or less

NFR-11: Repository searching should occur as soon as the user begins typing their query

NFR-12: URL validity checking should occur as soon as the URL is entered

Design Constraints:

The system will run as a web application on all major browsers. It will need to be tailored to each browser to ensure proper functionality. The database should be managed with MySQL. Analysis data should be visualized using CanvasJS. Main application functionality should be written in python. Each page of the application should be designed to follow a common theme.

Purchased Components:

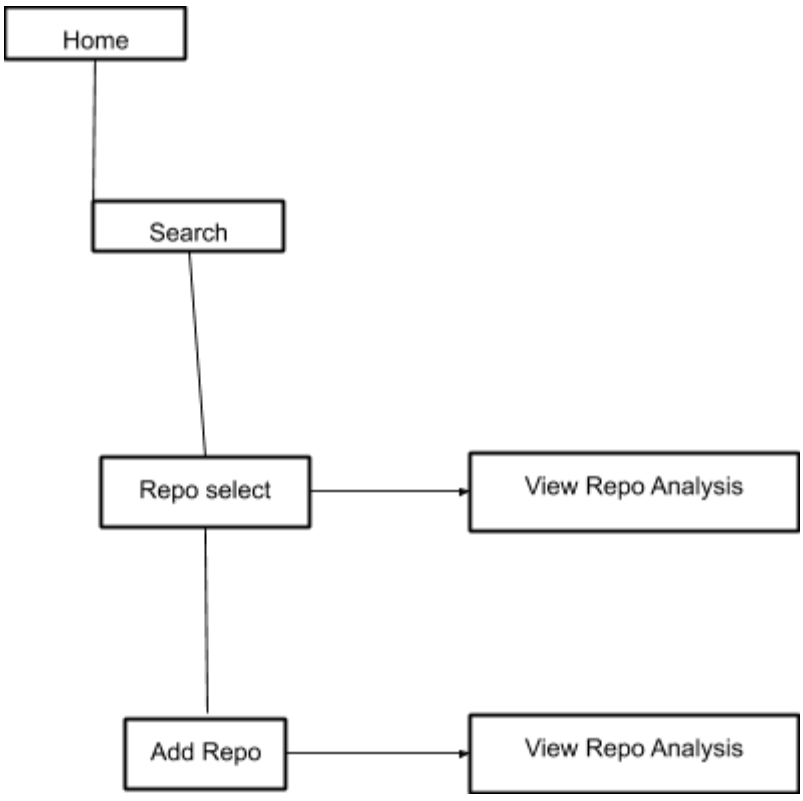
CanvasJS License……..$399

AWS Webserver……..$$$

AWS Database…….$$$

Interface:

Screen Flow:



Hardware Interface:

The webserver will have an IP address that is TBD.

Software Interface:

The web application will have to communicate with the database. There will also need to be a way for the application to access data from Github.