Prototype

This document describes the prototype of a GitHub repository visualization tool developed to help developers better understand their code repositories. The tool utilizes diagrams, to provide users with insights into their repositories' structure, complexity, and activity. The tool is designed to be highly customizable, allowing users to save the visualizations to their preferences and needs. The prototype was developed using open-source technologies and tested with several GitHub repositories to evaluate its effectiveness. The results of the testing show that the tool can provide valuable insights into code repositories and help developers improve their workflow and productivity. The document concludes with a discussion of the potential applications and limitations of the tool, as well as recommendations for future development and research.

Problem to solve:

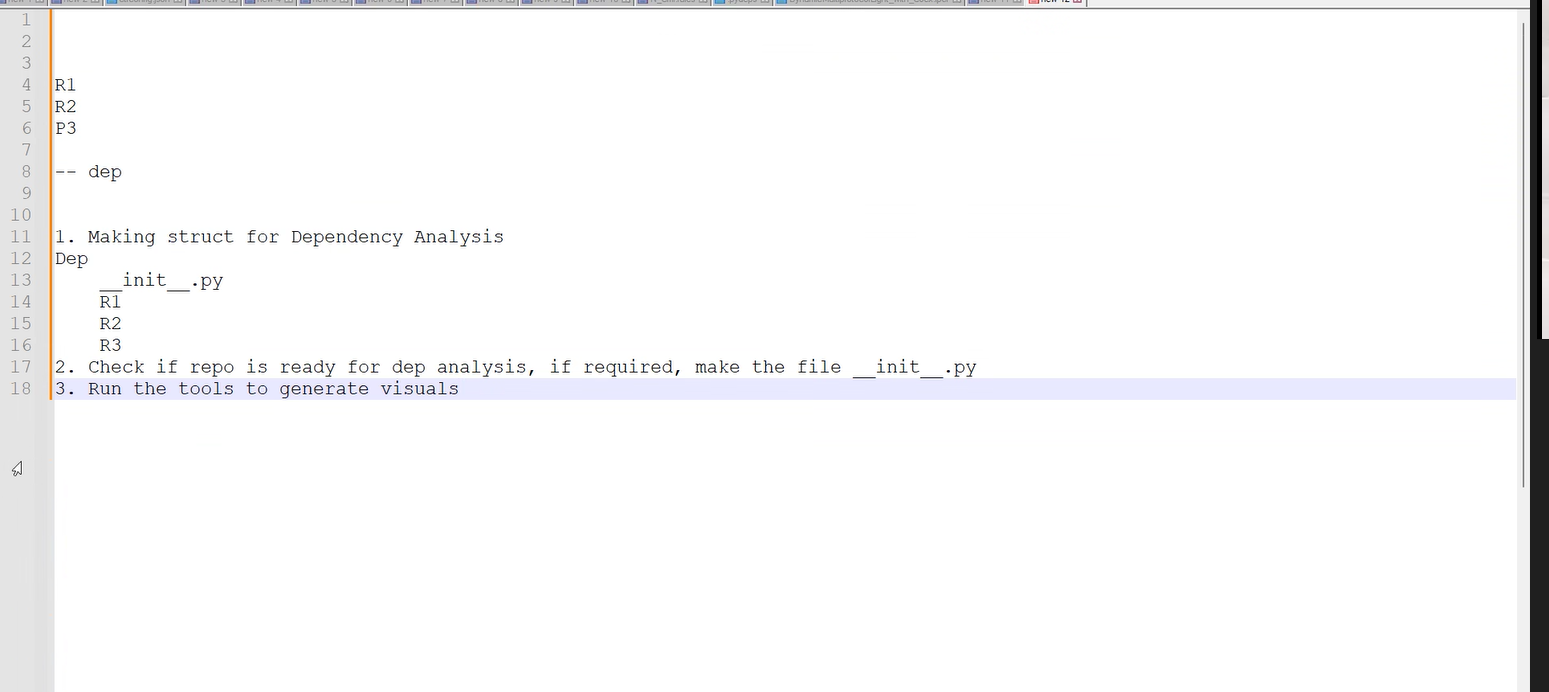
The developers face a problem that they must update their dependencies on the project multiple times and sometimes there is vulnerability in the dependency that can result in the threat for the project.

Approach:

To solve this issue, I started with creating a visualization to show the dependencies, but it was not useful as it does not show the repositories to which the certain dependencies are connected, clearly.

Below you can see an image showing the dependency graph and its limitations. As one must look into the graph to see which repository it is visualizing. Which was not very user friendly, so I had to change my approach.

The process followed to create the visualization is also mentioned below where R1, R2, R3 are short names for repository one two and three.



Performing the analysis, we get the following output:



To get to this approach I test many tools and technologies like GitHub Kraken and glow boards, I also investigated the GitHub dependency visualization. I also checked many languages like C and C# and python and came to conclusion that python will be the best for the project. Since python has many libraries and can be used and maintained much easily than the other technologies.

Next I wanted to get some more information on what troubles other developers face and what should be the solution to their problems and how should the solution look like:

Upon interview I got following points:

1. Problems:
   1. Some dependencies have malware and use needs to be notified.
   2. Dependency visualization is being done manually in organization and hence need a tool to keep an update of dependencies every day.
   3. Some dependencies have same name (in team) but totally different code in it.
   4. Not all dependencies need immediate attention so how to find which need immediate attention and to separate them from others.
   5. Some users prefer GUI where they can click on a button to see the graph.
   6. Some users do not prefer the GUI or the CLI they just want to get notified in case something is wrong with the dependencies.
2. UI requirements
   1. Upon interviewing people regarding the UI I realized there were mixed opinions as stated above regarding how the app should look like.
   2. One interesting UI idea that was suggested was to give score to the vulnerabilities and color code them as per the threat level.

Conclusion:

I decided that it is better to have a CLI that can be run from time to time, and it keeps an update of the dependency graph by the date.

The reason for this is that this tool will be used by the developers, and it does not need aesthetic properties and most of the users prefer the CLI version. The script can be run every day, and it keeps an updated copy of the dependency graph in the modules stored in the folder updated. Color code will be given to the dependencies as per their threat level. And user shall be notified when the threat is found.

Hence, after this idea I created a prototype which can be seen below to understand the functioning of the application.

Link: https://vjimjj.axshare.com