Project Plan

Code Dependencies visualization

|  |
| --- |
| **Date : 2/16/2023** |
| **Version : 2** |
| **State : In Draft** |
| **Author : Author** |

#### Version history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** | **State** |
| 2 | 02/16/2023 | Lokesh Agnihotri | Updated project scope from MRR to multiple repositories. Fixed the letter type to uniform.  Updated analysis part regarding using the multiple technologies. Like Gephi and Github dependency graph. Updated scope. Updated communication plan with coach.  Updated main research Question.  Updated End Prouct of Project.  Improved expansion on testing strategy. | Planning State |
|  |  |  |  |  |
|  |  |  |  |  |

**Distribution**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Receivers** |
|  |  |  |
|  |  |  |

Contents

[1. Project assignment 4](#_Toc42673512)

[1.1 Context 4](#_Toc42673513)

[1.2 Goal of the project 4](#_Toc42673514)

[1.3 Scope and preconditions 4](#_Toc42673515)

[1.4 Strategy 4](#_Toc42673516)

[1.5 Research questions 4](#_Toc42673517)

[1.6 End products 4](#_Toc42673518)

[2. Project Organisation 6](#_Toc42673519)

[2.1 Stakeholders and team members 6](#_Toc42673520)

[2.2 Communication 6](#_Toc42673521)

[3. Activities and time plan 7](#_Toc42673522)

[3.1 Phases of the project 7](#_Toc42673523)

[3.2 Time plan and milestones 7](#_Toc42673524)

[4. Testing strategy and configuration management 8](#_Toc42673525)

[4.1 Testing strategy 8](#_Toc42673526)

[5. Finances and Risk 9](#_Toc42673529)

[5.1 Project budget 9](#_Toc42673530)

[5.2 Risk and mitigation 9](#_Toc42673531)

# Project assignment

## Context

Sioux, wants to use the power of visualization to visualize various dimensions of a software code base.

This project aims at visualizing the dependencies of a github repository.

## Goal of the project

The objective of this project is to create a visual representation of the dependencies utilized in a Github repositories. This visualization will allow us to understand the impact that changes in a particular dependency may have on the various python scripts in the repository.

The final outcome of the project will take the form of a graph that showcases the interconnections between the dependencies and the corresponding scripts.

In a company that operates with complex software stacks and extensive codebases, it is essential to keep track of the dependencies and their versions. Any alteration in a version can potentially affect multiple files, leading to unforeseen consequences. By having a clear visual representation of the dependencies and their relationships with the scripts, we can proactively assess the impact of upgrading or downgrading a particular dependency.

## Scope and preconditions

|  |  |
| --- | --- |
| **Inside scope:** | **Outside scope:** |
| 1. Visualization of Dependencies | 1. Visualization of subdependencies |
| 1. Visualization of multiple github repository dependencies |  |

To visualize the dependencies used in a Github repository, I will checkout tools such as:

GraphViz: GraphViz is a powerful open-source tool for creating visual representations of complex data structures, including software dependencies. It can be used to create diagrams of dependencies between packages and libraries used in a Github repository.

GitKraken Glo Boards: GitKraken Glo Boards is a project management tool that integrates with Github and provides a visual way to track the dependencies between different scripts in a repository. You can use Glo Boards to create cards that represent scripts and dependencies, and connect them with lines to show the relationships.

Gephi: Gephi is an open-source network visualization tool that can be used to create graphs of dependencies used in a Github repository. You can import data from your Github repository and use Gephi's visualization tools to create interactive graphs that showcase the dependencies between scripts.

GitHub Dependency Graph: GitHub provides a built-in Dependency Graph that shows the dependencies used by a repository and the projects that depend on it. The Dependency Graph can be accessed from the "Insights" section of a Github repository and provides a visual representation of the dependencies and the scripts they are used in.

These tools can be used to create visual representations of the dependencies used in a Github repository and help understand the relationships between scripts and dependencies. This information can be valuable when making changes to the dependencies or upgrading to a new version, as it will give you a clear understanding of the impact the changes may have on the scripts.

Many of these tools are capable of connecting to Python and visualizing the dependencies used in Python scripts.

For example, GraphViz can be used to create graphs of dependencies used in Python scripts by parsing the import statements in your code and generating a diagram of the relationships between packages and libraries.

GitKraken Glo Boards can also be used to track the dependencies in Python scripts by creating cards for scripts and dependencies and connecting them with lines to show the relationships.

Gephi can also be used to visualize the dependencies in Python scripts by importing data from your code and creating interactive graphs of the relationships between packages and libraries.

The GitHub Dependency Graph is capable of tracking the dependencies used in Python scripts and providing a visual representation of the relationships between packages and libraries.

Overall, these tools can be useful in visualizing the dependencies used in Python scripts and can help you understand the impact of changes to the dependencies on the code.

## Strategy

Sprint Planning:

I will be have a meeting daily with the coaches to discuss the project goals and determine what needs to be accomplished each sprint. I will establish clear and measurable goals fort he sprint.

Daily Stand-Up Meetings:

I will hold daily stand-up meetings with Patrick or with collegues (Ignas, Valerian, Rares, Neik, )to discuss progress, identify any roadblocks, and plan next steps.

I will also use these meetings to ensure that I am aligned and working towards the same goals.

Sprint Review:

At the end of each sprint, I will review the work that was completed and assess progress.

I will identify areas for improvement and make changes to the process as needed.

I will also plan for the next sprint and make any necessary adjustments to the project plan.

Sprint Retrospective:

I will hold a sprint retrospective meeting to reflect on their work and identify areas for improvement.

I will discuss what went well during the sprint, what didn't go well, and what they can do differently in the future.

I will use this information to make improvements and ensure that future sprints are even more successful.

Continuous Improvement:

I will continuously monitor the project and make adjustments as needed to ensure that they are on track to meet the goal.

I will also use their learnings from previous sprints to make improvements and ensure that future sprints are even more effective.

By following this Scrum strategy, the project I will be able to ensure that I am making progress towards my goals, addressing any roadblocks in a timely manner, and continuously improving the processes.

## Research questions

**Main research Question**

How to viaualize the dependency of various github repositories.

**Sub research Questions**

What are the most commonly used tools and techniques for visualizing software dependencies in a Github repository?

What are the benefits of visualizing software dependencies in a Github repository?

How can visualizing software dependencies in a Github repository help organizations improve the development process and reduce the risk of errors?

What are the challenges associated with visualizing software dependencies in a Github repository, and how can they be overcome?

How can organizations use software dependency visualizations to identify areas of their codebase that may be vulnerable to security threats?

How can software dependency visualizations help organizations manage the complexity of their software stacks and ensure compatibility between different components?

What are the limitations of current software dependency visualization tools, and how can they be improved?

How can organizations integrate software dependency visualizations into their development workflows and processes to maximize their effectiveness?

## End products



# Project organisation

## Stakeholders and team members

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Abbreviation** | **Role and functions** | **Availability** |
| Lokesh Agnihotri  [394616@student.fontys.nl](mailto:394616@student.fontys.nl)  0645133976 | LA | Developer | 24/7 |
| Patrick IJntema  [p.ijntema@fontys.nl](mailto:p.ijntema@fontys.nl) | PI | Coach | Weekdays |
| Eric Slaats  e.slaats@fontys.nl | ES | Coach | Weekdays on appointment |

## Communication

I will be meeting with Patrick everyday at TQ building Fontys. In case of being sick I will be informing him via teams and will also attend meeting via teams. Incase we are not able to meet, I will stay in communication via texts on teams.   
The communication will happen everyday. It

# Activities and time plan

## Phases of the project

Analyse

Advise

Design

Realize

## Time plan and milestones

|  |  |  |  |
| --- | --- | --- | --- |
| **Phasing** | **Effort** | **Start date** | **Finish date** |
| 1. Analyze |  | 02/01/2023 | 2/13/2023 |
| 1. Advise |  | 2/13/2023 | 2/15/2023 |
| 1. Design |  | 2/16/2023 | 2/19/2023 |
| 1. Realize |  | 2/20/2023 | 2/28/2023 |

## 

# Finances and risk

## Project budget

There will be no budget for the projects, All the technologies needed are available.

## Risk and mitigation

|  |  |  |
| --- | --- | --- |
| **Risk** | **Prevention activities** | **Mitigation activities** |
| 1. Late delivery | Regular communication with stakeholders to clarify requirements, adequate resource planning and allocation | Implementing a contingency plan to ensure timely delivery like working on weekend |
| 1. Technical problems | Thorough testing of all components, using updated technology | Regular monitoring and maintenance, having a backup plan in case of technical issues |
| 1. Poor collaboration | Regular meetings and open communication channels | Daily update to either of the coach. |