Gitulyse
Functional Specification

Adrian Irwin - 20415624
Afolabi Fatogun - 20409054
06/11/2023

1. Introduction

Overview

Gitulyse is a code reporting/summarising tool which uses the GitHub API to parse useful analytical information and metrics to a web app/dashboard. The web app/dashboard will display the information in a user-friendly way. It will provide a one-stop platform for tracking progress, viewing contributions, and gaining valuable insight into a GitHub repository. The project will be useful for persons who want to see the progress of contributors to a GitHub repository.

The project idea came from the need for individuals and people in management positions to constantly monitor the progress of a project. Gitulyse aims to quicken the progress of code analysis to satisfy this need.

Glossary

Term	Definition
GitHub	A platform to store repositories that can be shared and worked on with other users.
Repository	A place where code and files are stored with version history.
Commit	A record of the changes made to files at a certain time and date
Pull Request	A proposed feature change that includes multiple commits. It needs approval and once approved gets merged into a larger codebase.
Mockup	A design of a system before implementation.

2. General Description

2.1 Product / System Functions

2.1.1 User Registration and Authentication

- Scenario: A new user wants to use the reporting tool.
- Steps:
    1. User opens the web app.
    2. User signs in using a GitHub account
    3. The system verifies the input and logs in the user.

2.1.2 Connecting a GitHub Repository

- Scenario: User wants to track a GitHub repository.
- Steps:
    1. User logs in to the web application.
    2. User navigates to the appropriate “Repository” section.
    3. User provides the repository URL.
    4. The system validates the GitHub repository and establishes a connection.
    5. The repository metrics are displayed on the dashboard

2.1.3 Viewing Repository Metrics

- Scenario: User wants to see contributor metrics for a connected repository.
- Steps:
    1. User selects a connected repository from the dashboard.
    2. The system fetches data using the GitHub API and processes it.
    3. The dashboard displays metrics such as commit count, pull requests, issues, etc.
    4. User can filter metrics by date range, contributor, or file type.
    5. User can compare one or more contributions in the same repository

2.1.4 Generating Reports

- Scenario: User wants to generate a summary report for a specific date range.
- Steps:
    1. User selects a connected repository and chooses a date range.
    2. The system retrieves the relevant data using the GitHub API.
    3. The system generates a summary report including metrics like contributions per contributor, file changes, etc.
    4. User can download the report as a CV.

2.1.5 Dashboard Customization

- Scenario: User wants to customise the appearance and layout of the dashboard.
- Steps:
    1. User logs in to the web app.
    2. User navigates to the “Settings” section.
    3. User can choose different themes, set default metrics to display, and arrange widgets on the home page.
    4. Changes are saved and reflected in the dashboard.

2.2 User Characteristics and Objectives

Generally, there are three different user classes, Managers, Employers and Individual Developers. The expertise of these users ranges from none or minimal interaction with computer systems and GitHub to interaction with these systems on a daily basis. Although some users could have no interaction with these, it would be expected that the user has some knowledge of how GitHub works to be able to utilise the application.

2.2.1 Individual Developers

An individual developer would span a wide range of skill levels in terms of GitHub usage but overall would have the largest subset of users that interact with GitHub daily. There would also be a considerable amount of users that have little GitHub experience or are still learning how to use GitHub. Overall, all of these users would want the system to provide enough information and insight on their repositories and contributions that will help boost their productivity and code quality.

A developer would have many requirements for this system, this would include:

- A dashboard that has customisable options so that users can have a quick view of the metrics that matter the most to them.
- Metrics that update in real-time so that users will not have to wait to get information on recent contributions.
- An option to export statistics into a virtual CV that can be provided to employers.

2.2.2 Managers

For Managers, it would be expected that they have a good understanding of how GitHub works even if they aren’t using it daily. This is so that they can interpret the metrics provided by the system. Managers would want a system that allows them to easily view the contribution over time and the split of work between employees. This

would allow managers to see project progress and give them more metrics to better facilitate decision-making.

A manager would also have multiple requirements for this system:

- Comparison metrics and graphs between employees to see the amount of work done by each employee.
- Access to historical data on metrics, to get an overview of how much has been contributed over a selected time period.

2.2.3 Employers

For Employers, there could be a mix of users who are familiar with GitHub and users who have not used GitHub before. The majority of these users will fall into the group that have not used GitHub, as such the system needs to be simple enough for them to understand how these metrics display the amount of work put into a project by a contributor. This system would assist an Employer to be more effective in the review of a candidate based on data from the GitHub metrics.

Some requirements that an employer would have for the system are:

- A profile for an inputted candidate that shows a rundown of everything that the candidate has worked on and their proficiency in different languages.
- An option to export a report on a candidate to use when recommending a candidate to a hiring team.

2.3 Operational Scenarios

Management tool

Gitulyse GitHub code reporting tool will serve as a comprehensive management tool for optimising collaborative code development. It provides a one-stop platform for tracking progress, viewing contributions, and gaining valuable insight into a GitHub repository. Gitulyse would be an asset for project managers and team leads looking to enhance productivity and collaboration in their code development endeavours.

Virtual CV

Gitulyse would provide and export summarised detailed reports which would provide valuable insight into a contributor’s proficiency and workflow. Gitulyse will provide insight into a developer’s technical skills and work habits, which are valuable information for potential employers.

Personal Reporting/Tracking Tool

For individuals, Gitulyse will act as a powerful personal reporting and tracking tool. The Gitulyse dashboard will display metrics in an easy-to-understand style, allowing users to assess their progress, track commit counts, and analyse pull requests and

issue activity. Customizable settings and the option to filter metrics based on particular criteria offer a personalised tracking experience. It will also produce detailed summaries for certain date ranges, providing a user with relevant data for self-improvement.

2.4 Constraints

2.4.1 Hardware Platform

Due to the number of information that will be shown to the user, Gitulyse will be designed for usage on desktop browsers. Mobile screens have a substantially lower amount of real estate, thus, making it harder to show all the necessary data to users.

2.4.2 Speed Requirements

In terms of the speed at which updates made on GitHub are reflected on Gitulyse, the aim is to have updates reflected in real-time with an upper bound of 10 seconds. Some factors impacting this would include:

- Issues with the GitHub API/Webhooks causing data to not be sent or being delayed.
- Large changes to the repo causing a large amount of information needing to be parsed all at once.

3. Functional Requirements

3.1 Log In

Description - To use Gitulyse, the user must log in. The user logs in using their email and a password, the user will then have the choice to link their GitHub account to their login. This allows the system to be populated with all the repositories that the user either has created or has contributed to.

Criticality - High

Technical Issues - If the system cannot connect to the database the user will not be able to log in, leaving the whole system inaccessible.

3.2 Log Out

Description - To stop using Gitulyse, the user must have the option to log out. The user will be able to log out by pressing a button provided in the user interface.

Criticality - High

Technical Issues - Log out button not working.

Dependencies with Other Requirements - This requirement relies on the Login requirement. The user must log in to the system first for the logout button to be accessible to the user.

3.3 Selecting default repository

Description - Gitulyse will allow the user to choose a default repository that will be shown to them on their dashboard at login. The user should be able to pick any repository that they have contributed to.

Criticality - High

Technical Issues - The system will store the chosen repository and must have a way to change the chosen default repository.

Dependencies with Other Requirements - The user must be logged in to get a list of the user's repositories that the user has contributed to.

3.4 Repository search

Description - Gitulyse will provide users with a way to search for any repository that they have access to/contributed to on GitHub.
Criticality - Medium

Technical Issues - The system must not show any repositories that the user has not contributed to. Also, it must show private repositories that the user has access to.

Dependencies with Other Requirements - The user must be logged in so that a list of their repositories can be found.

3.5 Metric viewing

Description - Gitulyse will allow the user to access a dashboard that shows them chosen metrics on a chosen repository.

Criticality - High

Technical Issues - Gitulyse must be able to get all relevant data from the GitHub API for a repository. This data will then need to be parsed correctly so the user can easily read the metrics.

Dependencies with Other Requirements - A repository needs to be chosen from the repository search or as a default repository for metrics to be shown.

3.6 Metric filtering

Description - Gitulyse should allow users to filter metrics based on specific criteria, such as time range, contributors, or specific metrics (e.g., lines of code, commits, issues resolved etc).

Criticality - High

Technical Issues - The system must effectively filter out redundant data to ensure the viability of the metrics. Additionally, the UI should provide intuitive options for users to select and apply filters.

Dependencies with Other Requirements - This requirement will require GitHub API integration to retrieve and process metrics based on the selected filters.

3.7 Customising dashboard

Description - Gitulyse will provide users with the ability to customise the dashboard layout, including selecting which metrics to display, arranging their placement, and setting personal preferences for visual representation (e.g., charts, graphs).

Criticality - Medium
Technical Issues - The system will implement a drag-and-drop interface for dashboard customization similar to interfaces like Grafana.

Dependencies with Other Requirements - This requirement may be intertwined with the Metric Filtering functionality, as customised views may involve selecting specific metrics.

3.8 Export metrics report

Description - Gitulyse shall support the exporting of summarised metrics and reports in various formats (e.g., PDF, CSV) for offline access and further analysis and as a CV.

Criticality - Medium

Technical Issues - The system will implement export functionalities. It will also handle any formatting required for different export formats.

Dependencies with Other Requirements - This requirement may interact with the Metric Filtering functionality, as users may want to only export specific metrics.

3.9 Comparing individual contributions

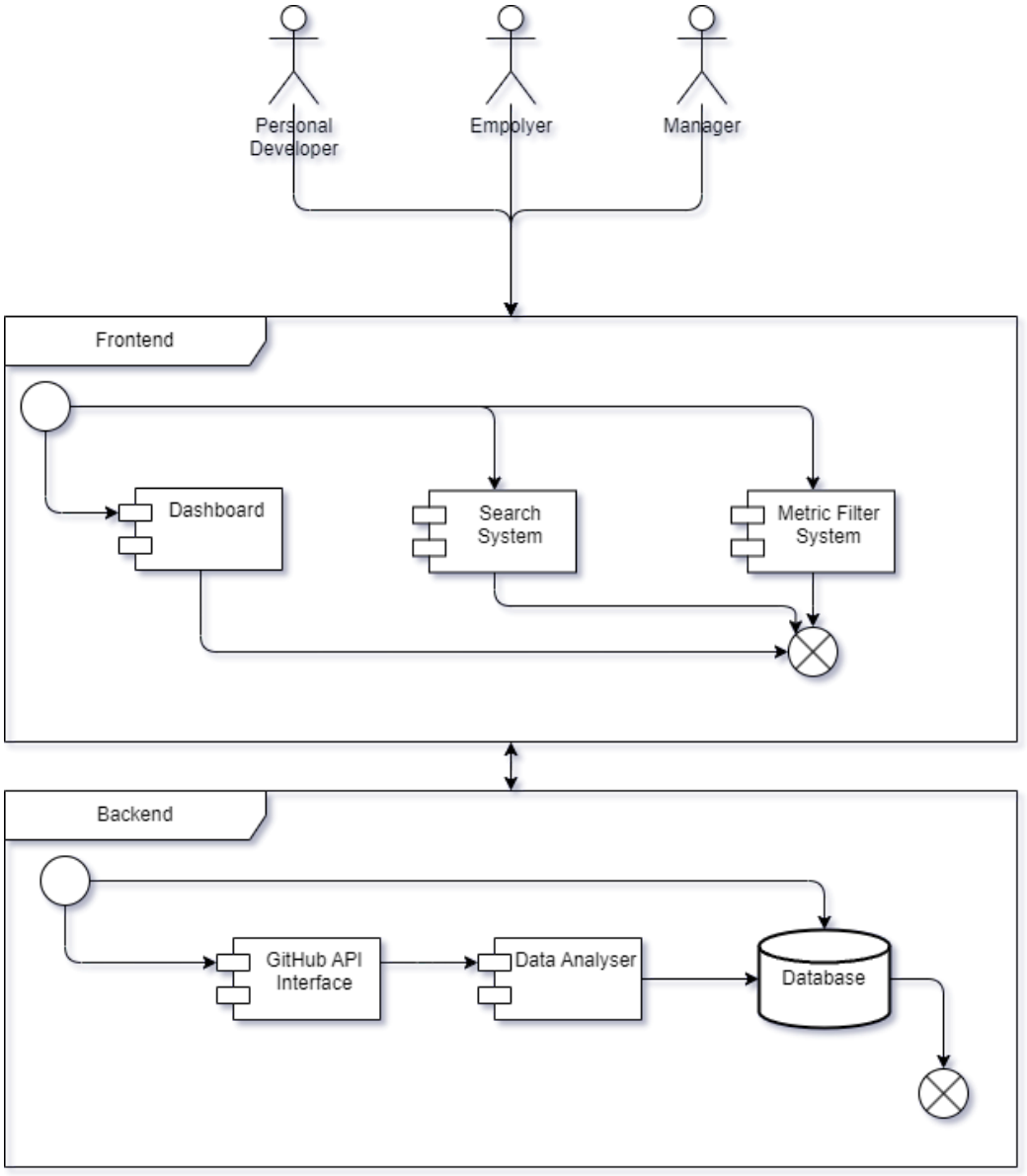
Description - Gitulyse will provide a feature to compare individual contributions, allowing users to view metrics for specific contributors side by side.

Criticality - Medium

Technical Issues - The system should retrieve and process individual contribution metrics from the GitHub API and present them in a comparative format.

Dependencies with Other Requirements - This requirement may interact with the Metric Filtering functionality to select specific contributors for comparison. The system may also facilitate the comparison between all contributors in a repository.

4. System Architecture

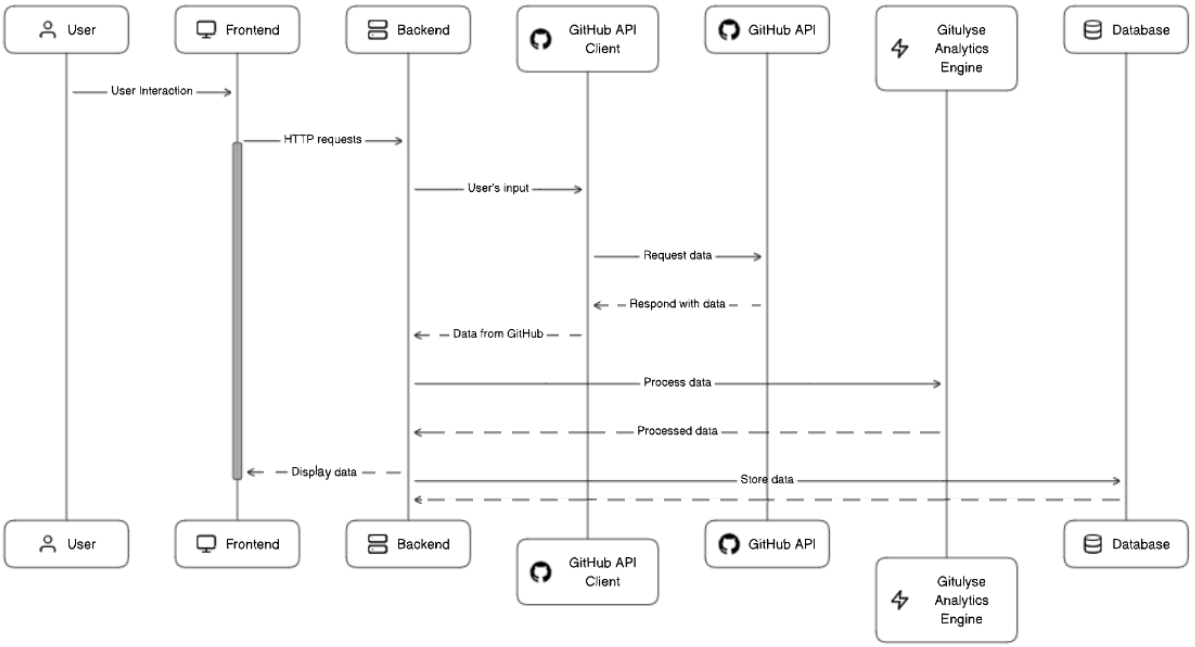


The system architecture diagram gives a high-level overview of the components across the Gitulyse system. The users all interact with the system using the frontend. The frontend consists of three major components, these being, the Dashboard, Search System and the Metric Filtering system. The user can initially interact with any of these components and each will give the user information about chosen repos.

The Frontend then interacts with the Backend which also consists of three major components, these being the GitHub API Interface, Data Analyser and the Database. The backend can initially be interacted with by the frontend through either the Database or the GitHub API Interface. The system first checks the Database for any existing data to display then will enter the GitHub API Interface to gather new data.

5. High-Level Design

Sequence Diagram

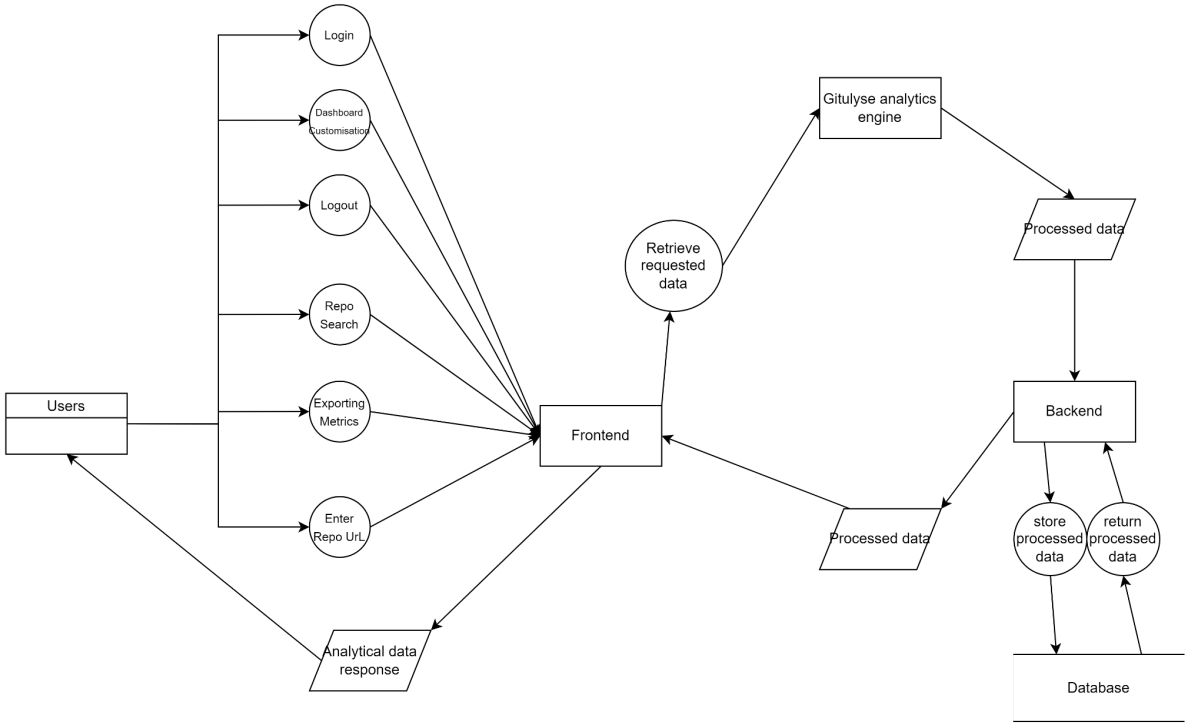


A sequence diagram illustrates the interactions between different components or objects in a system over a specific period of time. The above sequence diagram demonstrates the process of a user querying and receiving analytics data for a specific GitHub repository.

Explanation of the above diagram:

- A user interacts with the Gitulyse web application frontend through their browser
- Upon successful authentication, the backend/API server sends a request to the GitHub API client for repository data.
- The GitHub API client interacts with the GitHub API to retrieve the requested repository data.
- The GitHub API responds with the corresponding repo data, which is sent to the GitHub API client
- The GitHub API client passes the repository data to the backend/API server.
- The backend/API server processes the data using the Gitulyse Analytics Engine for analytics and is then stored in the database.
- The backend/API server sends the analytical data response from the backend to the frontend.
- The frontend displays the analytics data to the user on the web application frontend.

Data Flow Diagram

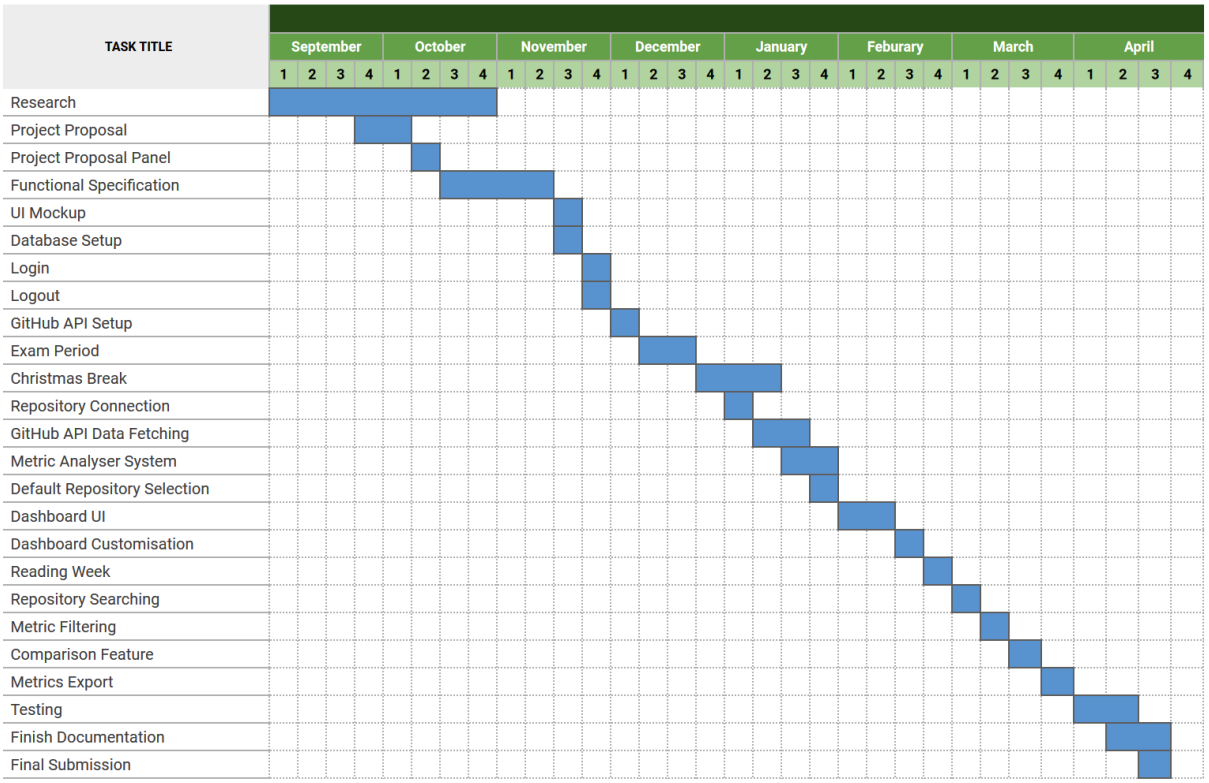


The data flow diagram maps out the flow and routes of information in the Gitulyse system. The above data flow diagram gives insight into the processes, products, Entities, and data storage in the Gitulyse system.[1]

6. Preliminary Schedule

The initial plan for the development of the system is to work on the backend first, then move towards working on the frontend as the backend nears completion.
The first number of tasks will be mainly tasks related to the infrastructure of the system, i.e. Database, Login/Register System, and mockups of the user interface will be created for a clearer vision of the frontend. The next step would be to work on setting up the connection to the GitHub API, retrieving data and analysing that data. Frontend will be worked on once the data fetching and analyser systems are completed. The initial components to implement on the frontend are the dashboard and the repository selection interfaces. After this, customisation, search, filters, comparison and export features will all be worked on to finalise the system.

The GANTT chart below outlines the timeline of the backend and frontend features, outlined above, up until the submission date.



7. Appendices

[1]Lucidchart, “What is a Data Flow Diagram | Lucidchart,” *Lucidchart.com*, 2017. https://www.lucidchart.com/pages/data-flow-diagram