MIDDLE EAST TECHNICAL UNIVERSITY

NORTHERN CYPRUS CAMPUS

CNG491 Computer Engineering Design I

First Iteration of Report and Presentation

Project Name: Git Assessor - Automated Assessment of Git Usage of Students

Project Participants:

Ziya Taner Keçeci – 2152049

Yusuf Mete Kabakçı - 2151983

Dogukan Çatal - 2257996

Mahmut Ali Şahin - 2243673

1. Introduction

Git is a popular source-code version control system that allows software developers to work simultaneously and keep track of their changes over time. In this project, we will develop a web application that can be used by instructors to assess the git repositories of their students.

1. Motivation

For that assessment, instructors will choose some rules which will be added by our team to the app so that instructors will be able to get an evaluation/assessment report of their students' projects. Creating a web application that helps both instructors and students is our main **motivation** and for that **we are planning** to give as much detail as possible in the Assessment Reports.

2. Aim & Objectives

In Git Assessor, instructors will be able to enter multiple Git Repository links and for every Repository link there will be the list of rules that have been chosen by the instructor. In that list rules instructor will be able to see which rules have been covered and which rules haven't been covered. Also, instructors will be able to see participants of each repository with their performance score. Participant performance score will start from 100 and for every rule violation their point will be deducted, and instructor will be able to see which participant violated which rule. To increase the readability of the Assessment Report there will be bar/pie charts for detailed visualization of students' performances.

3. Methodology

Most critical point of this project is the set of rules that we will add to the application. Utility of the Assessment Report will be dependent on the rules that have been provided and to provide effective rules we will be using an academic paper named "Errors and Poor Practices of Software Engineering Students in Using Git". We will implement the rules that have been covered in this paper and instructors will be able to choose the rules they want.

2. Requirements

2.1. Stakeholders

Instructors that will use application to get assessment for their students' git projects.

Students that will be evaluated based on their performance.

2.2. Functional System Requirements

1- An instructor shall be able to provide GitHub repository links to system.

•A text box will be provided as an input for instructors to enter their desired repository links.

2- An instructor shall be able to delete any repository link from the list before getting the report.

•The instructor can use the check boxes to select his/her desired repository links to remove/delete those links by pressing to the remove button.

3- An instructor shall be able to choose which rules will be used for assessment.

•Check boxes will be provided with each rule and the instructor can choose any rule that he/she wants to check whether students obeyed or not.

4- An instructor shall be able to get an assessment report for repositories and participants.

•Assess button will be provided for the instructor to get an assessment report when the repository links and the desired rules are selected.

5- For the rule set, the system shall provide the list of rules and instructors shall be able to configure them.

6- Assessment report will provide Visual representation of the data. Such as Bar charts of student performances.

•A new page will be provided to let the instructor to observe the students' performances, mistakes, and grades. For easier reading some of those data will be provided as charts (eg. Bar chart, ⋯,).

2.3. Non-functional Requirements

1- The System shall be available 7/24 for the users.

2- An Instructor shall be able to receive assessment report in less than 10 seconds.

3- An Instructor shall be able to get assessment report for up to 5 repositories at once.

4- The System shall provide visualized charts in the assessment reports.

5- The system shall use the GitHub repository link that has been provided by the instructor to get data from GitHub.

6- The system shall give error messages when an instructor tries to add an invalid repository link to the repository list.

7- The system shall give an error message when an instructor does not provide any link or/and rule to the system.

2.4. Domain Requirements

1- Git API doesn’t allow too many requests thus the number of requests and repository sizes are constraints.

2.5. Assumptions and Justifications

1- We assume repositories should be public.

2- We assume project repositories will not be huge so that GitHub doesn’t cut off the connection.

3- We assume students’ consents have been taken to be assessed by their performance.

3. System Modelling

3.1. Structured Use Case Diagram

In figure 1, we provide Structured Use Case Diagram for our Git Assessor Web Application while User, GitHub (Git API), Git Assessor Web Service are actors.

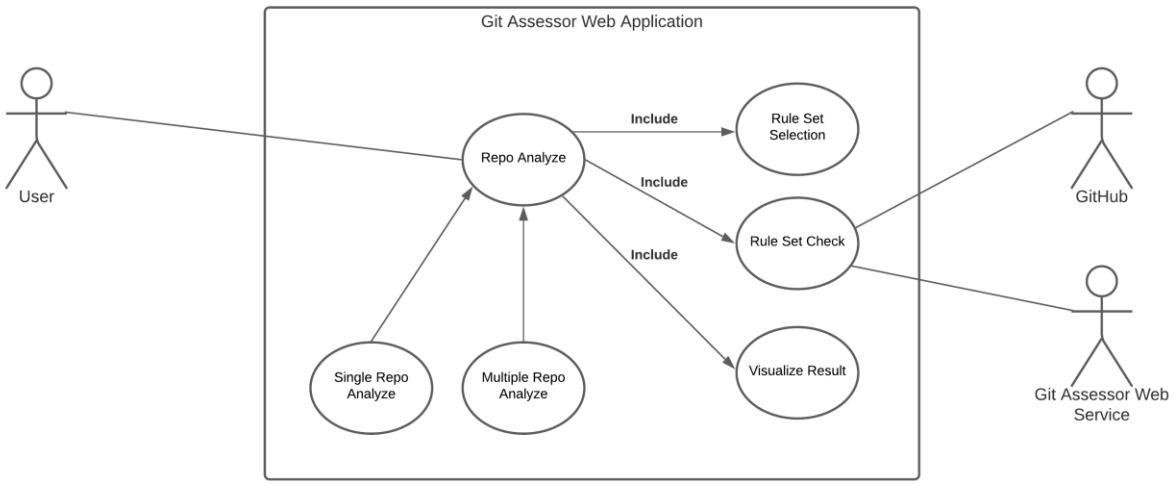


Figure 1 Structured Use Case Diagram

3.2. Sequence Diagram for Major Use Cases

In Figure 2, we provide Sequence Diagram for our major use case. Git Assessor Web Application is getting all the repos and rules from user in one time but asking Git Assessor Web Service about repositores once at a time.

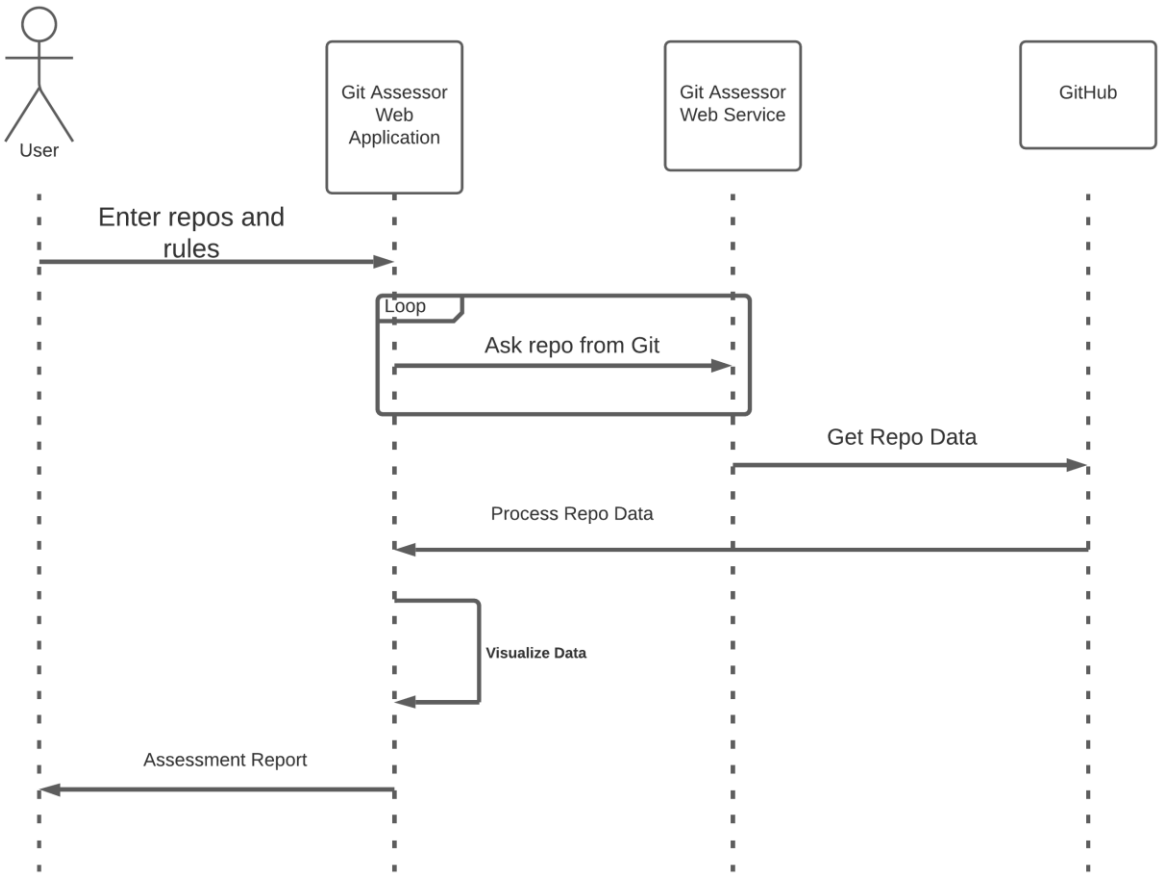


Figure 2 Sequence Diagram

3.3. Context Model

In Figure 3, we provide Context Model. Git Assessor Application is connecting to GitHub through GitAssessor Web Service.

Figure 3 Context Model

3.4. Architectural Model

In Figure 5, we provide Architectural Model. Git Assessor Web Service contains GitHub API and Rule Checker. Git Assessor Application contains Rule Selector and Report Generator. Git Assessor Web Service sends/receives data to/from GitHub and Git Assessor Application (Web Application).

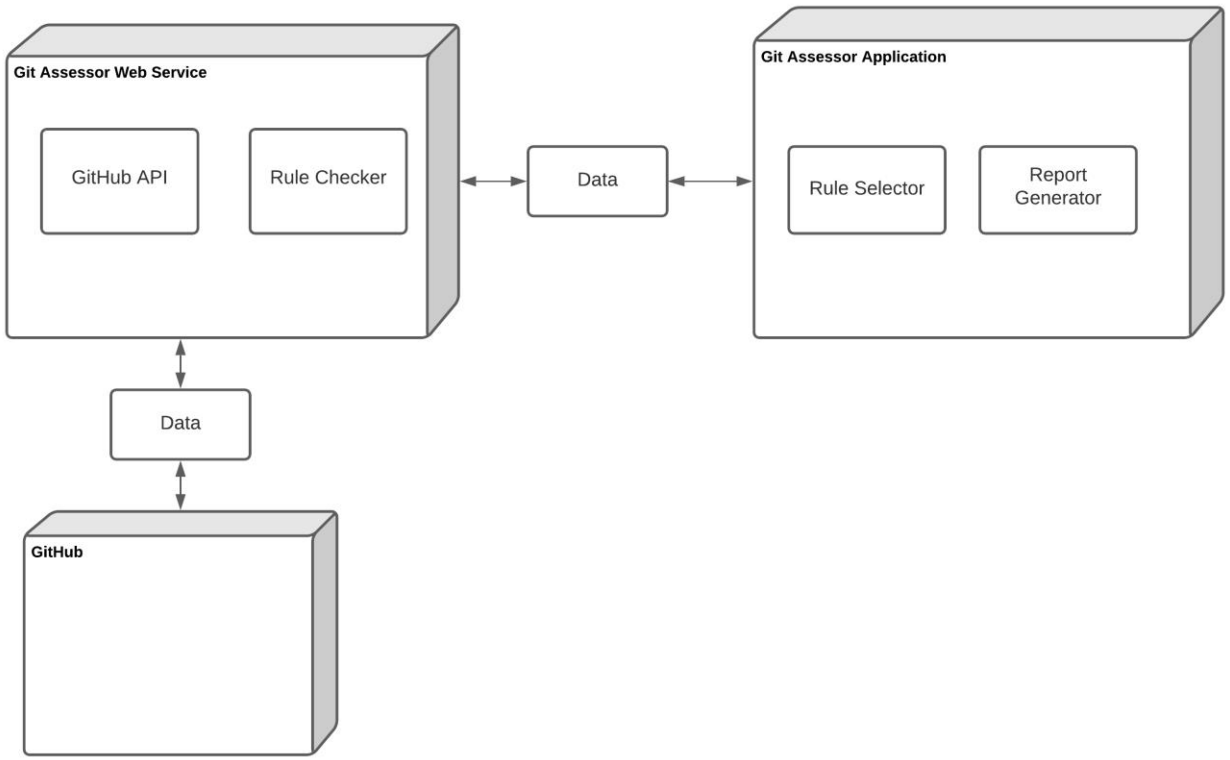


Figure 4 Architectural Model

3.5. Process Model

Application will start with asking for a Repo Link to be entered. If the user wants to add more links, he/she will be able to do that. After adding all desired links, the user will need to select rule(s). When all links and rules are selected, the user will be able to assess all those links with selected rules to get a report.

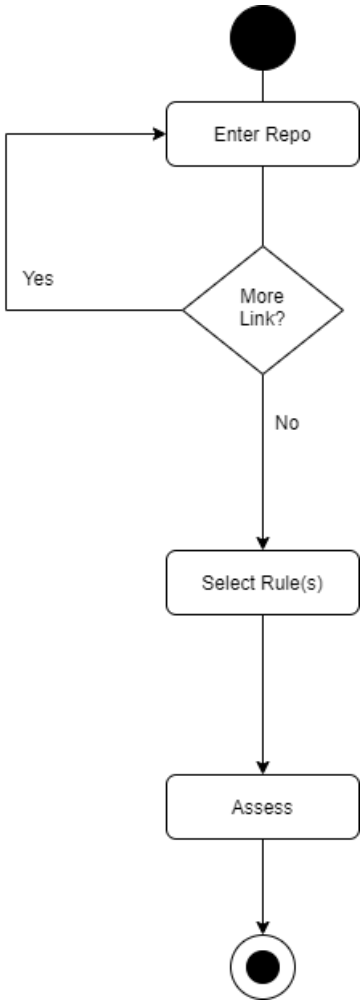


Figure 5 Process Model

4. (Graphical) User Interface

4.1. Home Page

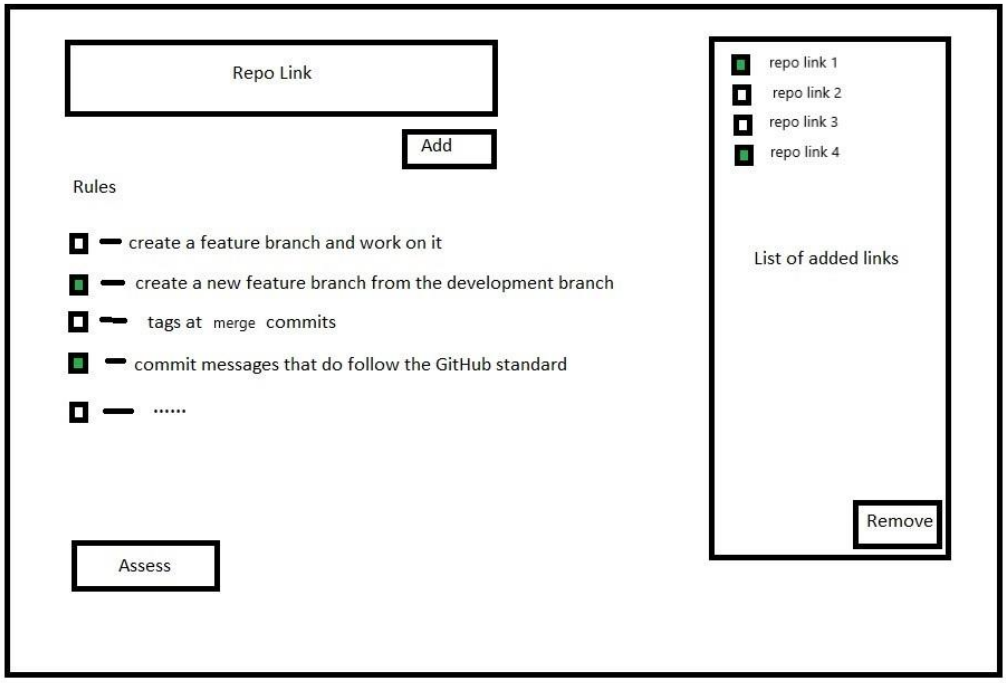


Figure 6 GUI Screen 1

4.2. Assessment Report Page

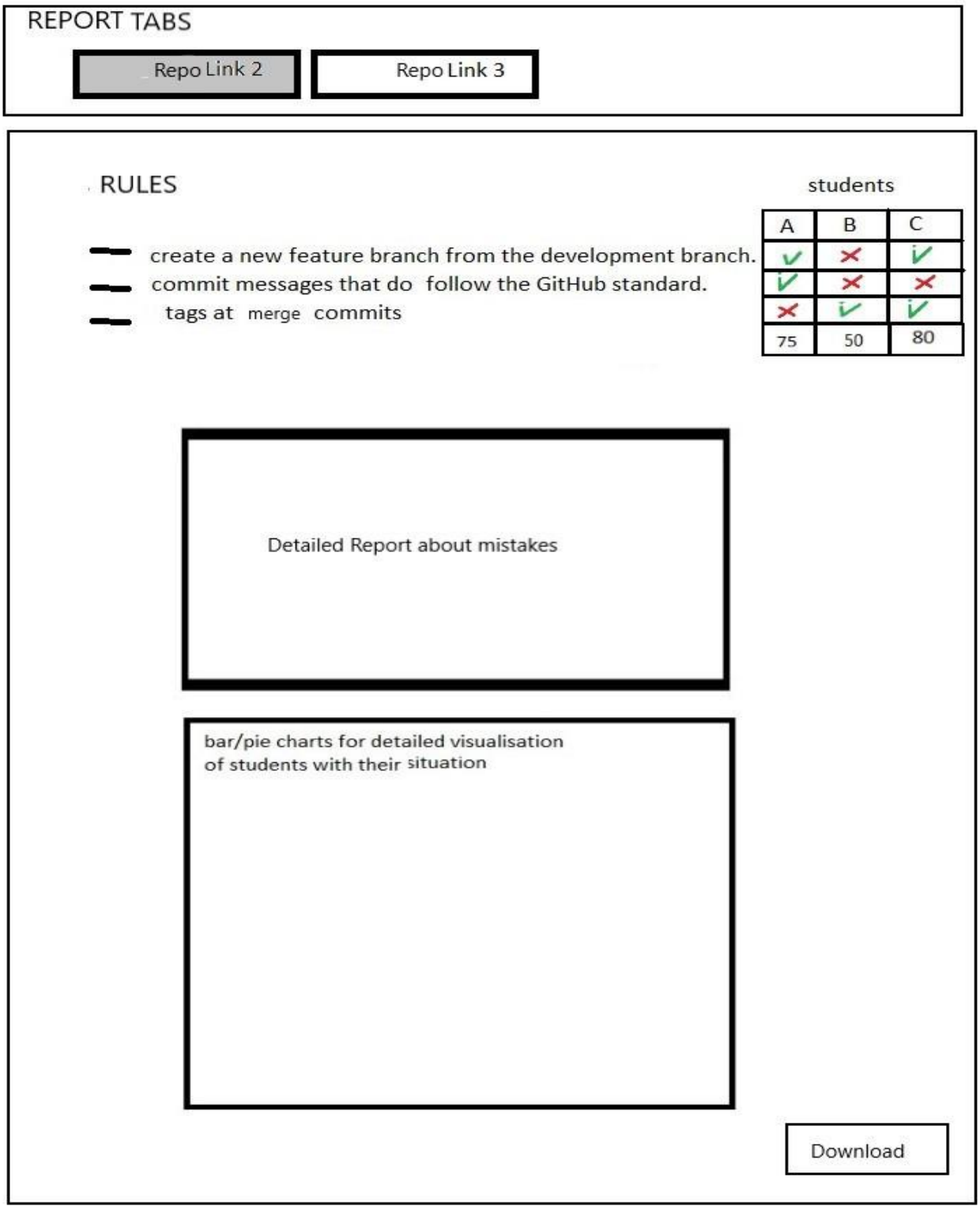


Figure 7 GUI Report Page

5. Agile Development with Scrum

5.1. Sprint Backlog

1- Watching an introductory lecture video for Git and GitHub.

2- Reading some introductory Git resources.

3- Reading about some Git workflows.

4- Reading about errors and poor practices of students in using Git.

5- Defining functional and non-functional requirements.

6- Writing an example use-case text

7- Writing domain requirements.

8- Creating structured use case diagram

9- Creating sequence diagram with major use cases

10- Creating a context model

11- Creating an Architectural model

12- Creating a process model

13- Describing first version of GUI with home page and report page

5.2. Sprint Burndown Chart

ID	User Story	Hours	Week1	Week2	Week3	Week4	Week5
T1	1- Watching an introductory lecture video for Git and GitHub.	1	1	1	0	0	0
T2	Reading some introductory Git resources.	1	1	1	0	0	0
T3	Reading about some Git workflows.	1	1	1	0	0	0
T4	Reading about errors and poor practices of students in using Git.	1	1	1	1	0	0
T5	Defining functional and non-functional requirements.	1	1	1	1	1	0
T6	Writing an example use-case text	1	1	1	1	1	0
T7	Writing domain requirements.	1	1	1	1	1	0
T8	Creating structured use case diagram	1	1	1	1	1	0
T9	Creating sequence diagram with major use cases	1	1	1	1	1	0
T10	Creating a context model	1	1	1	1	1	0
T11	Creating an Architectural model	1	1	1	1	1	0
T12	Creating a process model	1	1	1	1	1	0
T13	Describing first version of GUI with home page and report page	1	1	1	1	1	1
	TOTAL	13	13	13	10	9	0

Figure 8 Sprint Burndown Table

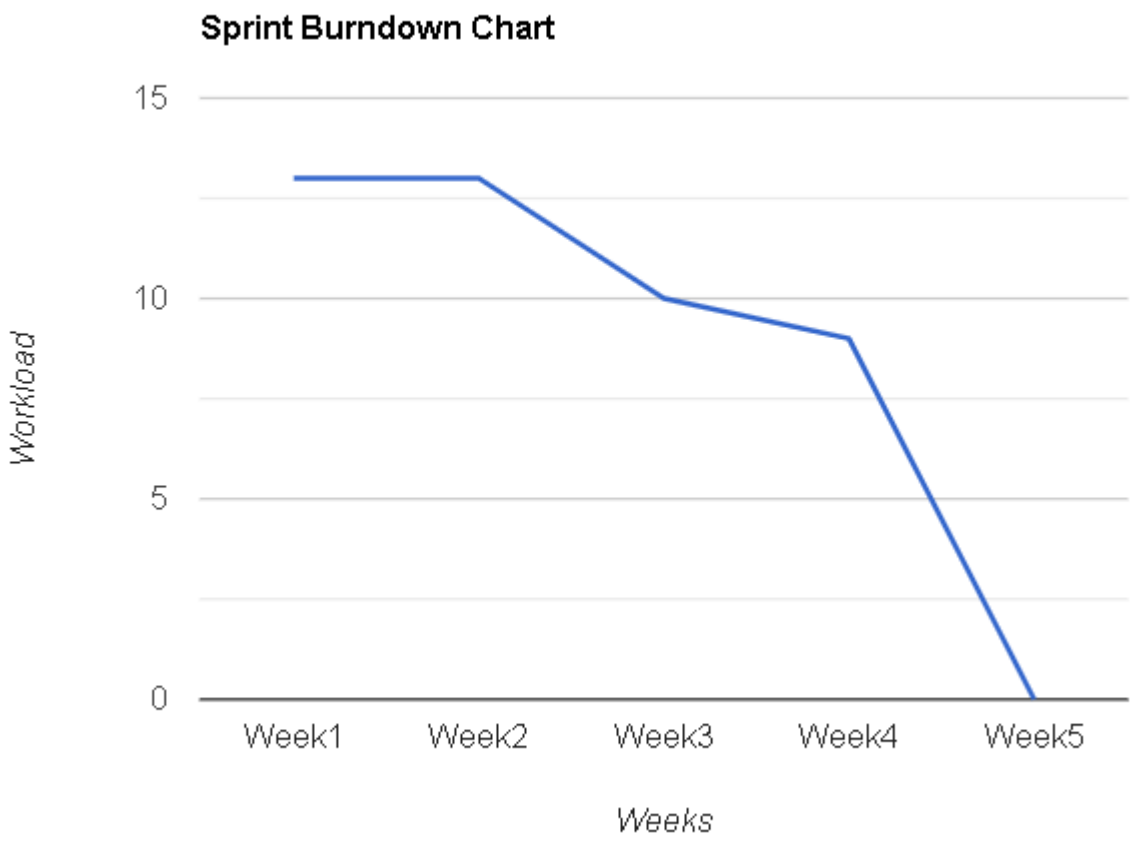


Figure 9 Sprint Burndown Chart

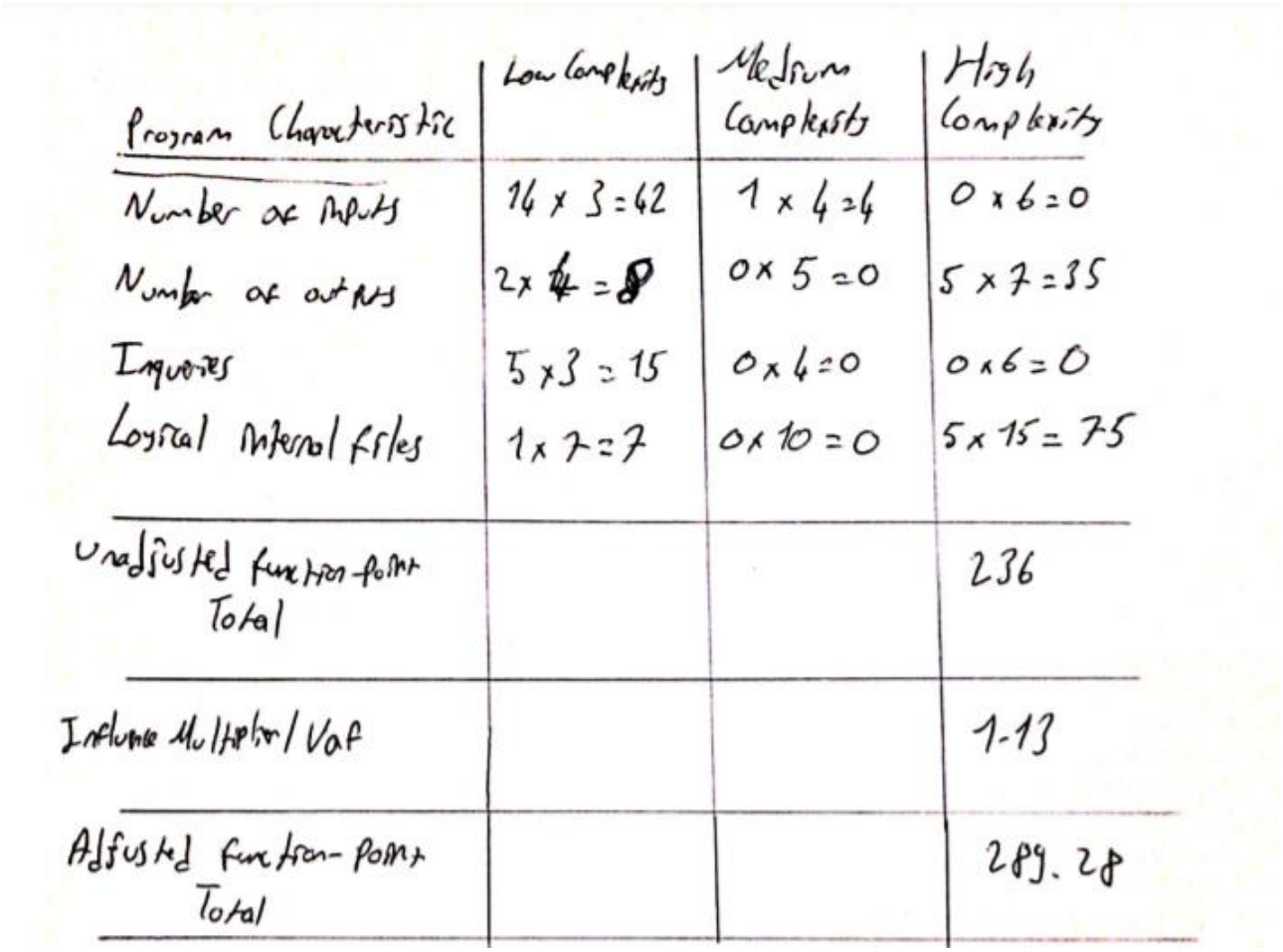
5.3. Sprint Review

Sprint has met its goals successfully. To achieve that, as a team, we divided tasks among team members, and before the due date, we held a meeting to discuss our processes. Fortunately, every team member had managed to finish individual tasks and because of that we only explained to each other what we had done and how we had done it.

5.4. Sprint Retrospective

Although we managed to catch the due date, it would be better if we had more meetings because we only held one meeting. In this sprint one meeting was enough but this may not be the same for upcoming sprints.

6. Project Estimation



Program Characteristic	Low Complexity	Medium Complexity	High Complexity
Number of Inputs	14 x 3 = 42	1 x 4 = 4	0 x 6 = 0
Number of outputs	2 x 4 = 8	0 x 5 = 0	5 x 7 = 35
Inquiries	5 x 3 = 15	0 x 4 = 0	0 x 6 = 0
Logical Internal files	1 x 7 = 7	0 x 10 = 0	5 x 15 = 75
Unadjusted function point Total			236
Influence Multiplier / VAF			1.13
Adjusted function-point Total			289.28

Figure 10: Calculation Of AFPT

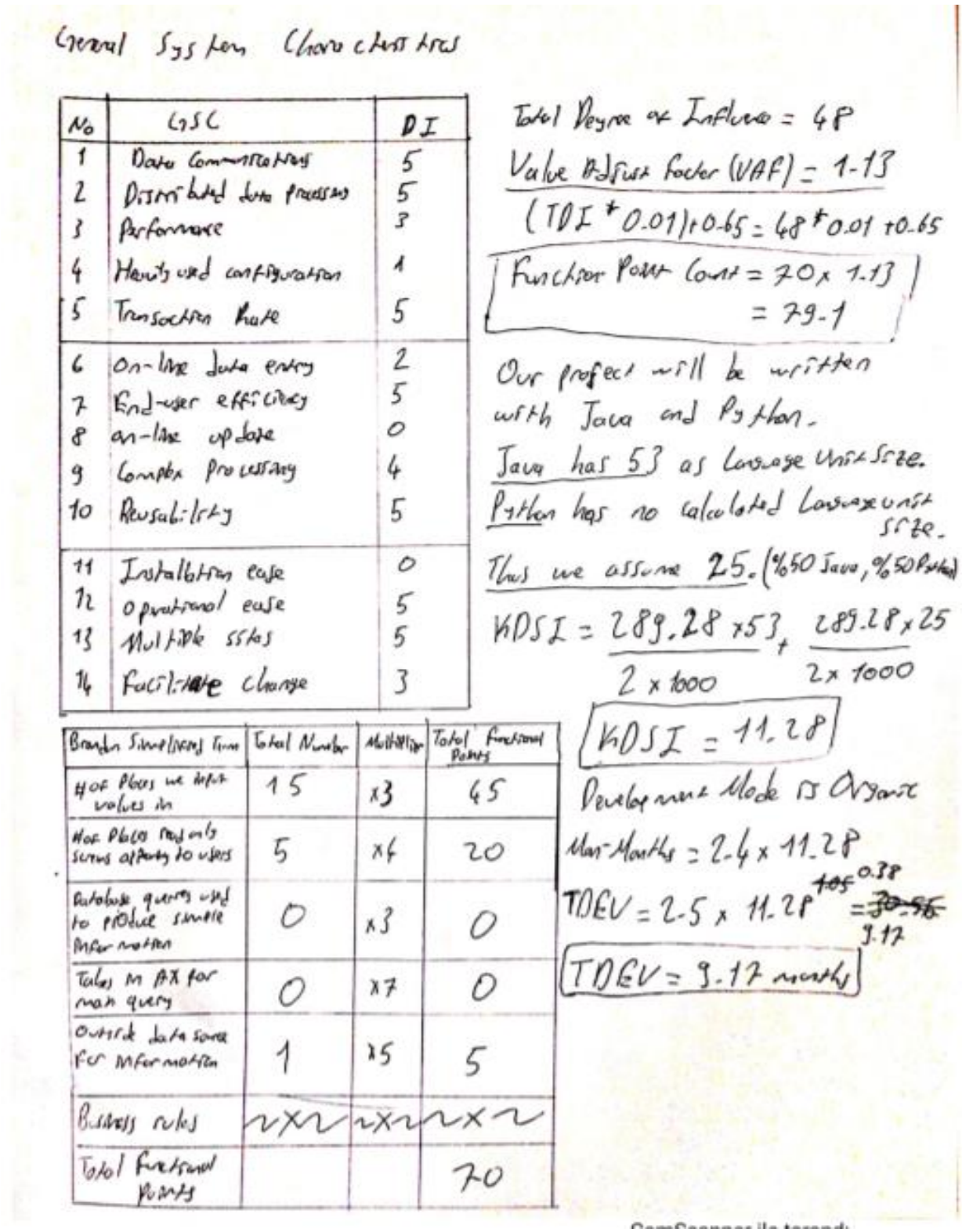


Figure 11: Calculation of TDEV and other parameters

VAF = 1.13 TDI = 48 TFP = 70 FPC = 79.1 AFPT = 289.28

KDSI = 11.28 MM = 30.56 TDEV = 2.4*(30.56^0.38) =9.17 months