AUTOMATING SELF EVALUATIONS FOR SOFTWARE ENGINEERS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Jonathan Rodrigo Alaura Miranda

June 2016

COMMITTEE MEMBERSHIP

TITLE:	Automating Self Evaluations for Software Engineers
AUTHOR:	Jonathan Rodrigo Alaura Miranda
DATE SUBMITTED:	June 2016
COMMITTEE CHAIR:	David Janzen, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Franz Kurfess, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Lubomir Stanchev, Ph.D. Associate Professor of Computer Science

ABSTRACT

Automating Self Evaluations for Software Engineers

Jonathan Rodrigo Alaura Miranda

Software engineers frequently compose self-evaluations as part of employee performance reviews. These evaluations can be a key artifact for assessing a software engineer's contributions to a team and organization, and for generating useful feedback. Self-evaluations can be challenging because a) they can be time consuming, b) individuals may forget about important contributions especially when the review period is long such as a full year, c) some individuals can consciously or unconsciously overstate their contributions, and d) some individuals can be reluctant to describe their contributions for fear of appearing too proud [24].

UNBIASED, Useful New Basic Interactive Automated Self-Evaluation Demonstration, is a web application designed to tackle the challenges of performing a self-evaluation by automatically gathering data from existing third party APIs, performing an analysis on the data, and generating a self-evaluation starting point for software engineers to build off. The third party APIs currently supported are: Bitbucket, Gmail, Google Calendar, GitHub, and JIRA.

ACKNOWLEDGMENTS

- A special thanks to my Mom and Dad for everything. Shout out to my big brothers, JJ and Chris, my big sister, Cheryl, and my little brother, Christian. We made it!

- A big thanks to Dr. Janzen for the idea, keeping me motivated, and advising me through this entire thesis process. Thanks to the Cal Poly CSC Department.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Software engineers perform self evaluations to assess their strengths, accomplishments, contributions, and areas of improvement. These self evaluations can be used to generate feedback for improvement or as a way to rationalize a promotion or raise. Performing a self-evaluation is challenging because it can take a lot of time. It is also easy for software engineers to forget to include important details. Another challenge is ensuring that the self evaluation is unbiased and credible.

Software engineers use a variety of tools that track their activities and contributions. When performing a self evaluation, an engineer can consult these tools as a helpful resource. Project management tools hold a record of all the work a software engineer has done. Software repositories are a similar resource, but with changes made to the source code. Emails hold a record of communication between an engineer and their peers, clients, or supervisors. A calendar or planner is a record that tracks where an engineer spends their time.

1.1 Contribution

The contribution of this thesis is a new web application, Useful New Basic Interactive Automated Self Evaluation Demonstration, or UNBIASED for short. UNBIASED is designed to help software engineers perform their self evaluations. UNBIASED can automatically request data from third party APIs, analyze the data, and use the resulting analysis to generate content to fill out a self evaluation. UNBIASED currently supports the following third party APIs: Bitbucket [3], Github [6], Gmail [8], Google Calendar [11], and JIRA [15].

This thesis aims to answer the following hypotheses:

- HYPOTHESIS-1: Can we solve the challenges of the self evaluation process for software engineers?
- HYPOTHESIS-2: Can we automatically generate a self evaluation for a software engineer?

Chapter 2

BACKGROUND

2.1 Self Evaluations

This section discusses the background of self evaluations: questions commonly answered, challenges related to performing a self evaluation, and the use of self evaluations in software engineering courses.

2.1.1 Self Evaluation Questions

This thesis focuses on the following five questions commonly found in self evaluations:

- QUESTION-1: List each of your major personal accomplishments/jobs on the project this quarter.
- QUESTION-2: You may describe [some of] them in detail if you feel it's appropriate or useful.
- QUESTION-3: In what specific ways could/should you have contributed more to your team project? Explain without making excuses.
- QUESTION-4: Assign yourself a project grade.
- QUESTION-5: Explain the rationale for the grade and justify it with specific details. Do not repeat material from earlier sections.

Note that these questions are gleaned from a self evaluation form given to software engineering students in a Capstone course, but they are general enough to be applicable to all software engineers. QUESTION-1 and QUESTION-2 emphasize what the

individual has accomplished during the self evaluation period. This lets the reviewer know what the individual has actually achieved. QUESTION-3 asks individuals to point out areas of improvement. This lets the reviewer know that the individual is aware of their weaknesses, and can be used as a starting point for how the individual can improve in later self evaluations. QUESTION-4 asks the individual to grade themselves on a familiar and common academic scale (A to F). This lets the reviewer know how the individual views the quality of their own work. QUESTION-5 asks the individual to rationalize the grade they gave themself. This is an opportunity for individuals to point out their strengths and to back up their claims with evidence.

2.1.2 Challenges

This thesis will focus on the following three challenges software engineers face when performing a self evaluation:

- CHALLENGE-1: Self evaluations can be time consuming.
- CHALLENGE-2: Individuals can forget about important contributions.
- CHALLENGE-3: Self evaluations can be biased and lack credibility.

CHALLENGE-1 highlights that self evaluations can be time consuming. The quality of the self evaluation will increase with more time, thought, and consideration. To perform a proper self evaluation, an engineer has to take time and reflect. First, the engineer has to go through a variety of resources to help them recall what they worked on. And then the engineer has to write their self evaluation, which comes with its own challenges.

CHALLENGE-2 highlights that individuals do not always remember what they contribute to a project. Consider that it is especially hard to remember accomplishments you made a year ago, versus accomplishments you have made in the last

four months. Work performed by software engineers is commonly documented in one form or another, but this still presents a challenge. There is some overlap with CHALLENGE-1, where individuals may have to spend a lot of time going through various records to help them recall what they have done. And once they find the data they need, they still need to prioritize what tasks are significant.

CHALLENGE-3 highlights that self evaluations lack credibility. Because self evaluations are written by oneself, there is potential for individuals to understate or overstate their contributions. In a 2006 study by Herbert [24], they found that individuals tend to not mark themselves accurately. The psychology behind these findings aside, providing concrete evidence can make all statements credible.

2.1.3 In Capstone Courses

At Cal Poly, software engineering students take an academic year Capstone sequence that consists of three courses: *Software Requirements Engineering*, *Software Construction*, and *Software Deployment*. In these courses, teams each develop the same system for an industrial sponsor. As part of their grade in each course, students must perform a self evaluation (see questions in Section 2.1.2) to grade themselves, highlight their accomplishments and contributions to the team, as well as evaluate where they could have done more. These courses also allow students to develop multiple skills and a unique perspective on their industry [22]. Chapter 5 will report on the evaluation of UNBIASED in Cal Poly's capstone courses.

2.2 Email

Emails are important to self evaluations because they contain a large record of communication. As an example of the usefulness of emails, consider the case when an email is received every time a user is being asked to perform a code review, and an

email is sent every time a user requests a code review. Other use cases for email include: invitations to events and meetings, meeting agendas, reports, documents, etcetera. Software engineering students use email to communicate with their teacher, class, project clients, and group members. One unique thing about emails is that there is no standard format with regard to content structure. This lack of a standard presents the challenge of figuring out what emails are important and relevant when running an analysis.

2.3 Calendar

Many software engineers use a calendar to keep track of different types of scheduled events: meetings, interviews, trainings, conferences, traveling, social gatherings, and even dedicated coding time. An event is a general term for an item on the calendar. Google Calendar is Google's calendar application and it allows for basic calendar functionality as well as attendee management (inviting attendees and tracking attendee responses). Some users may use one calendar for all of their events, but it is also possible for a user to separate their events into different calendars. Consider the use case of having multiple calendars, one for work and another for personal events. Regardless of a calendar event's source, the real data lies in the event information. Event information includes a title, start and end dates and times, a location, a description, whether or not it repeats, and information on the guests (who was invited, who accepted, who declined, who has not answered). Calendar analysis is important to self evaluations because it reminds the software engineer where their time has been spent.

2.4 Source Code Repositories

Source code repositories are used to store code and track changes in software projects. These repositories are a useful resource because they provide a way for engineers to take responsibility for code as well as commit messages. In this section, we will discuss two popular web-based code hosting services for source code projects. Between GitHub, that allows unlimited public repositories, and BitBucket, that allows unlimited private repositories, we are able to cover a large amount of software engineering teams.

GitHub is the largest code host with more than 35 million repositories [6]. GitHub allows software development teams to track issues so that they can stay on top of bugs and add features. GitHub data includes user account information, repository names, branch names for a repository, pull requests, issues, and commit information (messages, size, lines touched, files touched) on a branch, and comments on a commit.

Bitbucket supports both Git and Mercurial repositories. Features include pull requests and code reviews, branch comparison and commit history, and high integration with JIRA, a popular project and issue management app discussed in Section 2.5.

With data from code repositories, it is possible to answer the following questions relevant to self evaluations: What have I been working on? What files am I working on? When do I usually commit code? Which of my commits are large? Which of my commits are small? Who performs code reviews on my commits? Whose code commits do I review? What issues have taken the most amount of my time? What types of issues have taken the most amount of my time?

2.5 Software Management Tools

Software management tools are used by software engineers to help them manage their projects. Features often include planning, tracking, and releasing software. Software engineers are able to collaborate and track bugs, features, and all other issues related to their software. JIRA is one popular software management tool, and it can be used by all members of a software engineering team. With JIRA, software engineers can create, report, and be assigned to different types of issues. Users can easily log the amount of time spent on an issue with an optional comment to describe what they have accomplished [15].

Chapter 3

UNBIASED FEATURES

This chapter discusses the main features of the UNBIASED application.

- FEATURE-0: The UNBIASED system attempts to address the challenges of the self evaluation process.
- FEATURE-1: As a user, I can choose what data UNBIASED can access.
- FEATURE-2: As a user, I can request UNBIASED to analyze my data on supported third party APIs.
- FEATURE-3: As a user, UNBIASED can automatically fill out parts of my self evaluation.

3.1 Solving Self Evaluation Challenges

UNBIASED aims to solve the three challenges of self evaluations discussed in Section 2.1.2. They are presented below, again, for convenience.

- CHALLENGE-1: Self evaluations can be time consuming.
- CHALLENGE-2: Individuals can forget about important contributions.
- CHALLENGE-3: Self evaluations can be biased and lack credibility.

UNBIASED aims to solve CHALLENGE-1 by automating parts of the self evaluation process. Instead of manually going to each third party service and looking through record after record after record, users can use the easy-to-use interface of

UNBIASED to do the work for them, and all in one place. After gathering and analyzing the data, UNBIASED can even fill out parts of the user's self evaluation form: accomplishments, strengths, and areas of improvement.

UNBIASED aims to solve CHALLENGE-2 by looking at all records in the third party APIs. From this, UNBIASED attempts to prioritize and bring attention to high impact or important records. For example, project issues that are listed as having Critical Priority are probably a bigger contribution than a project issue that is listed as having a Low Priority. As a fallback, in the case where UNBIASED gets the order of items mixed up, the user is still able to examine all issues in the user interface.

UNBIASED solves CHALLENGE-3 implicitly because all of the input data to the system has an associated third party source. This source can be used as evidence to add credibility to claims made by the author.

3.2 Accessing User Data on Third Party APIs

We designed UNBIASED with users' trust in mind. UNBIASED allows users to opt-in to which third party APIs they want to authorize, and users are also responsible for when and what data UNBIASED analyzes. The following subsections discuss more about how users interact with third party APIs through the UNBIASED interface.

3.2.1 Gmail

Gmail provides a way to filter or query a user's emails using their own specialized syntax [1]. For example, to list all emails received or sent to or from a calpoly.edu address, the user would enter: [*from:*@calpoly.edu OR to:*@calpoly.edu*]. If a user wants to see all the people they interviewed, they could enter the query [*interview*] and that would look up all the emails that contain the keyword [*interview*]. For every

query a user wants to perform, they must enter the query in the input field and click on the "Search" button, and the results of each query will be shown below the search element. The user interface can be seen in Figure 3.1.

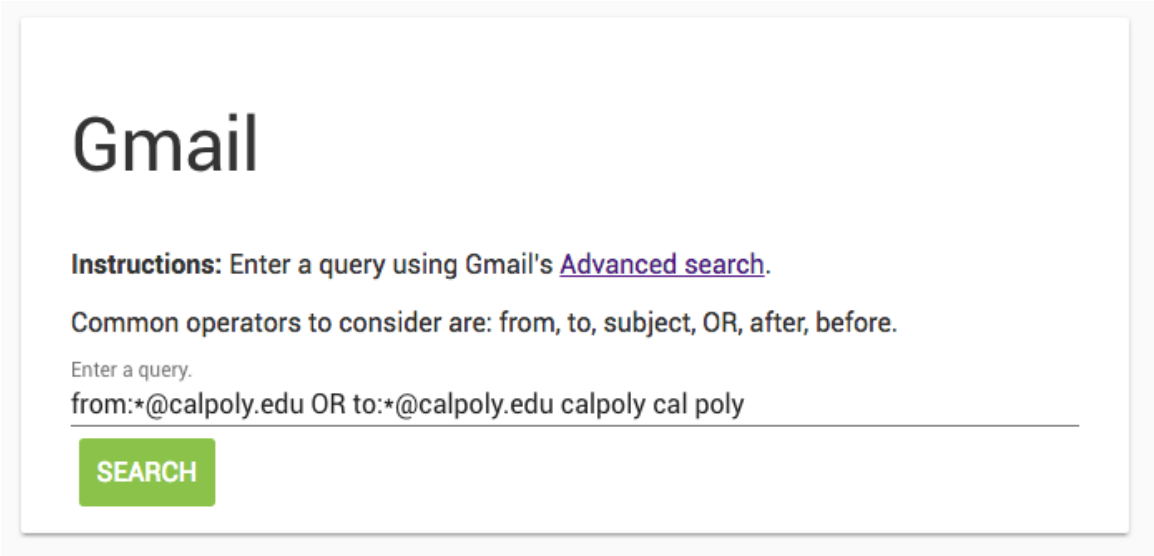


Figure 3.1: User interface to query the Gmail API.

3.2.2 Google Calendar

Once the user grants UNBIASED access to their Google Calendar data, we are able to present them with a list of all of the calendars to which they have access. This affords the user the option to select which calendars they want UNBIASED to analyze. The user has the option to limit the search query between a start and end date. The user interface can be seen in Figure 3.2.

3.2.3 GitHub

We present the user with four input fields. The first two input fields for the GitHub repository URL and the user's GitHub username are required. The third and fourth are optional input fields for the start and end dates which are used to limit the GitHub

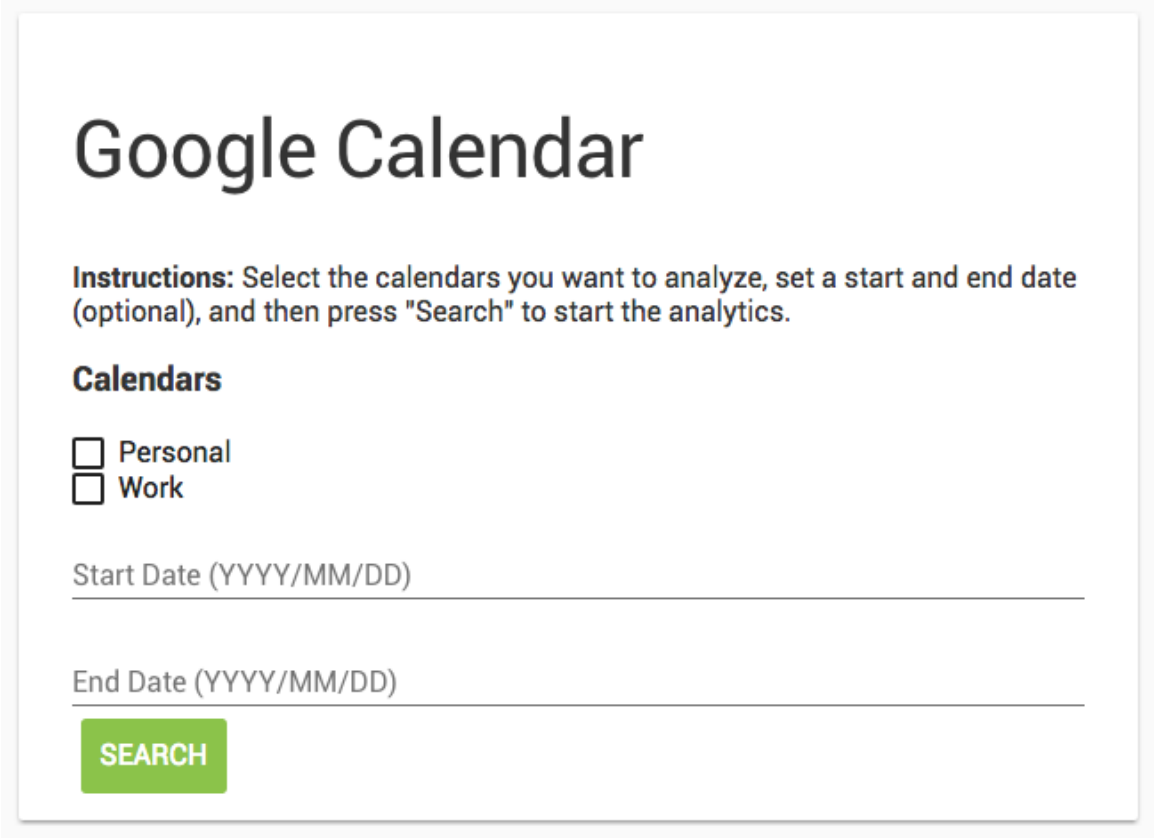


Figure 3.2: User interface to query the Google Calendar API.

response to be after the start date, before the end date, or between the two dates. To begin the GitHub search and analysis, the user can click the "Search" button. The user interface can be seen in Figure 3.3.

3.2.4 Bitbucket

Once the user authorizes UNBIASED to access their Bitbucket account, we present the user with a dropdown list of all the repositories to which they have access. The user must select which repository they want UNBIASED to analyze. The two optional input fields for the start and end dates are used to further limit the Bitbucket queries to be after the start date or before the end date or to be between the two dates. To begin the Bitbucket search and analysis, the user can click the "Search" button. The

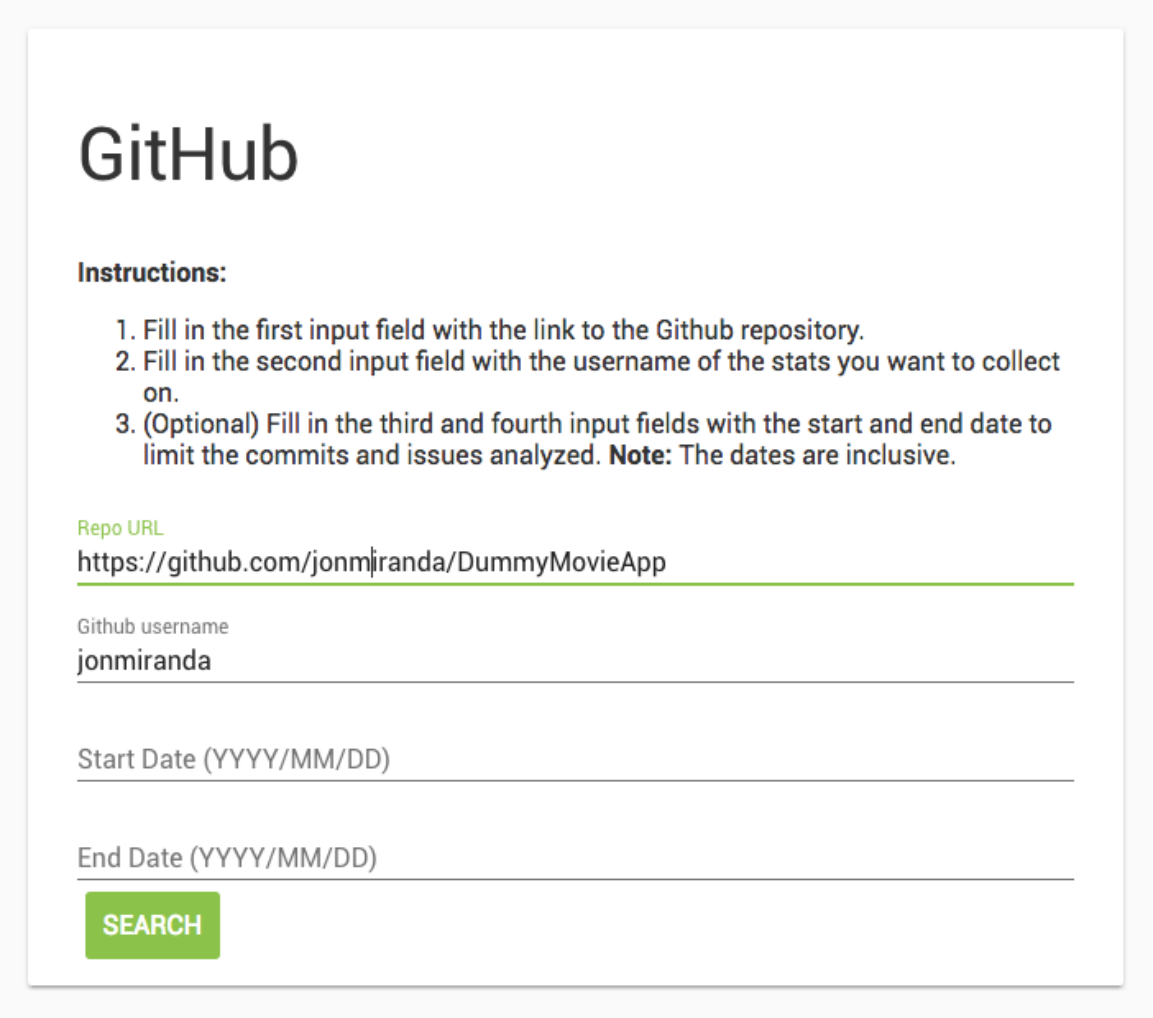


Figure 3.3: User interface to query the GitHub API.

user interface can be seen in Figure 3.4.

3.2.5 JIRA

UNBIASED is hooked into Cal Poly's Computer Science JIRA instance, where privacy between different projects and users is not of concern. Because of this, we are able to bypass the user-authentication step. We present the user with a dropdown of all the projects in the connected JIRA instance. We have an optional input field for the user's JIRA username. If the user presents their JIRA username, the search query

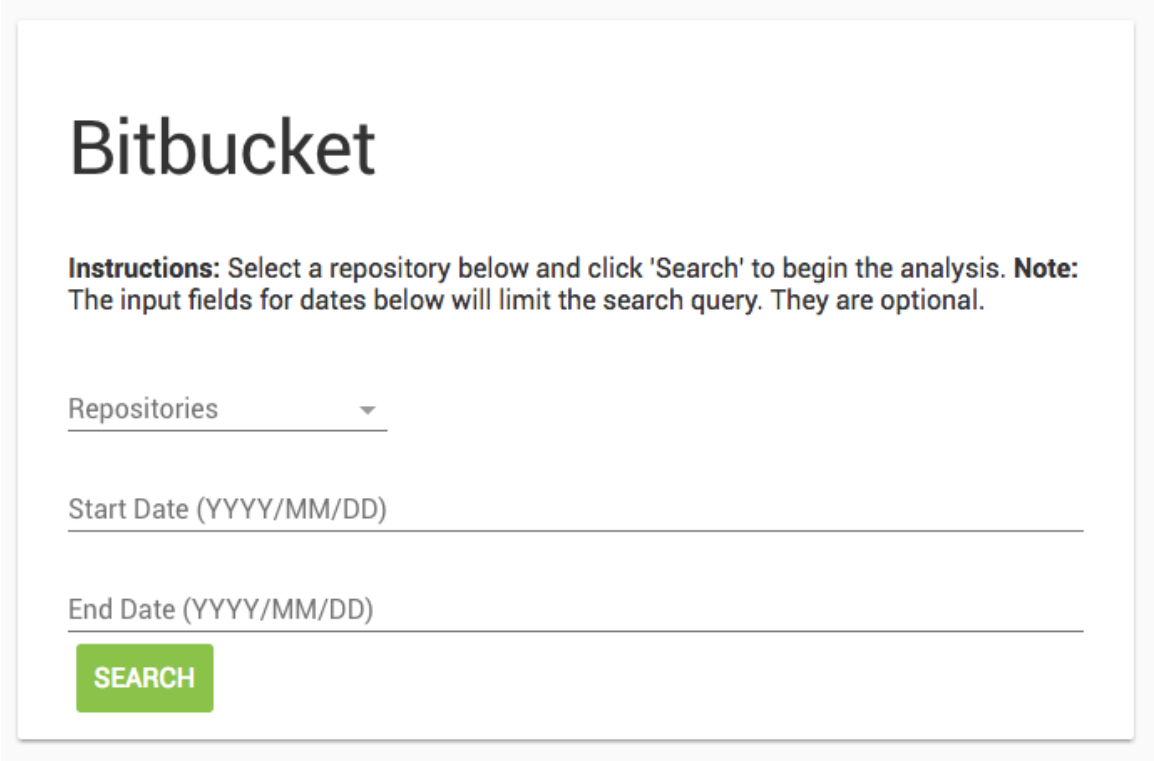


Figure 3.4: User interface to query the Bitbucket API.

will only look at issues where they are either the creator, reporter or assignee, or issues that they have logged work under. If the JIRA username is left blank, then JIRA will look at all issues within the selected Project, and will report stats for all users found in the project. To begin the JIRA search and analysis, the user can click the "Search" button. The user interface can be seen in Figure 3.5.

3.3 Third Party Data Analysis

In this section, we discuss the analysis performed on data from each supported third party API.

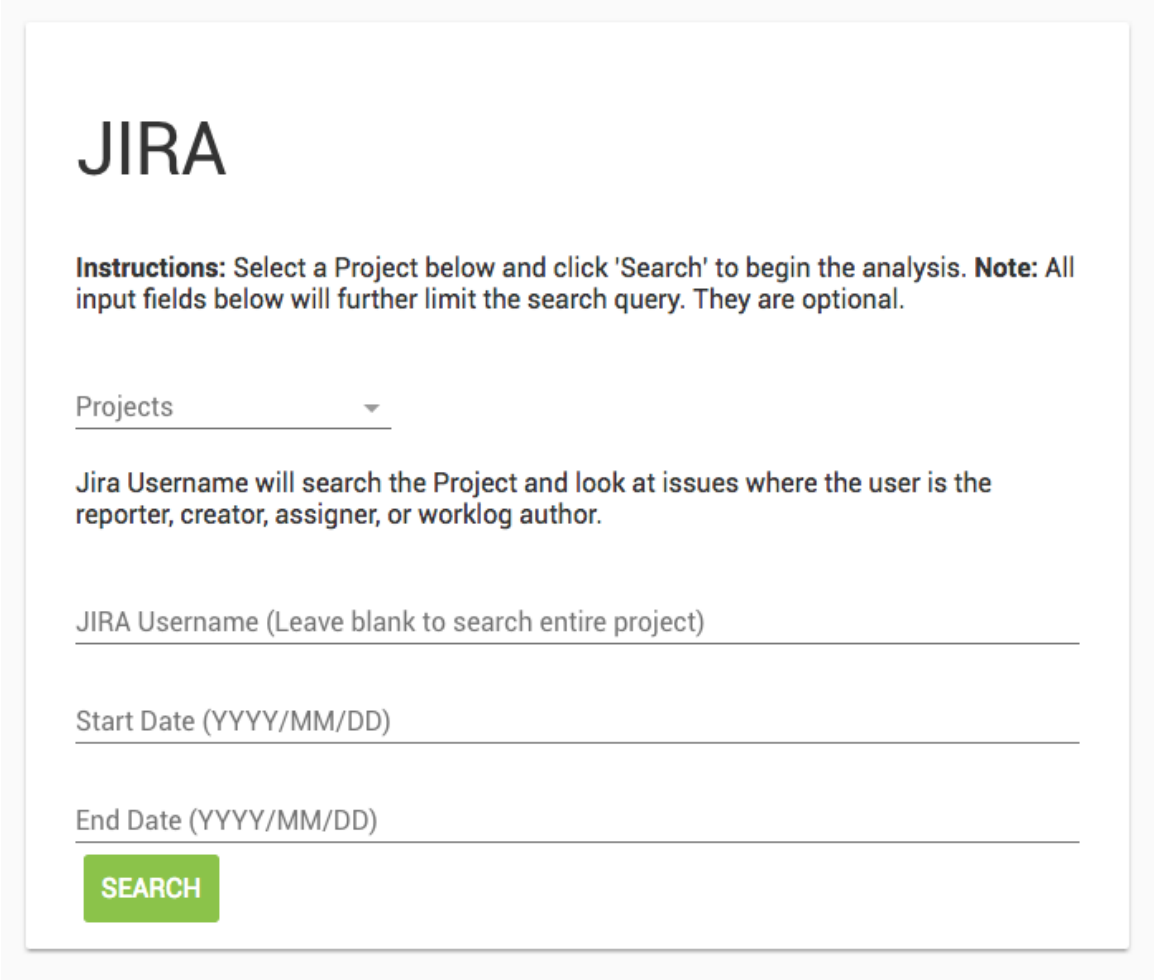


Figure 3.5: User interface to query the JIRA API.

3.3.1 Gmail

For performance reasons, UNBIASED only examines the email headers, subject, and snippet (email preview) provided by Gmail. The UNBIASED Gmail analysis is separated into: *Emails received by*, *Emails you sent to*, and *Word Frequency*. An example of a Gmail analysis by UNBIASED can be seen in Figure 3.6 and Figure 3.7.

- **Emails received by:** Counts the number of emails received by a particular address. This analysis can be used to let the user know who emails them the most.

- **Emails you sent to:** Counts the number of emails sent to a particular address. This analysis can be used to let the user know who they email the most.

- **Word Frequency:** Uses natural language processing to count the number of occurrences of a word found in all analyzed emails. This analysis can be used to highlight words that the user may find important. The user can dive deeper into the analysis by opening a dialog that links back to all of the emails that contain a particular keyword.

3.3.2 Google Calendar

UNBIASED returns two analyses performed on Google Calendar data: *Title Frequency* and *Title Word Frequency*. In both cases, the *Title* is based off of the title of the calendar event. The goal of both analyses is to identify where the user spends their time. An example of an UNBIASED Google Calendar analysis can be seen in Figure 3.8.

- **Title Frequency:** Returns a list of events, grouped by events that have the same exact title, sorted in decreasing order by the total amount of hours spent. There is also a column that displays the number of occurrences of each event. This is useful to see how much time is spent in recurring events like Weekly Meetings.

- **Title Word Frequency:** Uses natural language processing to tokenize the event's title and returns a list of events, grouped by events that contain the word, sorted in decreasing order by the total amount of hours spent. There is also a column that displays the number of occurrences each word appears in a title. This is useful for events that don't have the same exact title but can

be grouped together. For example, consider that all meetings, regardless of the nature of the meeting, may have the word "Meeting" in its event title.

3.3.3 Bitbucket

The Bitbucket analysis returns an analysis of: *Accomplishments*, *Issues*, *Commit Word Frequency Breakdown*, and *Commits* for a particular repository.

- **Accomplishments:** List of accomplishments generated from Bitbucket data. See Section 3.4 for more information on how this is achieved.
- **Issues:** Returns a list of issues sorted by priority (Blocker, Critical, Major, Minor, Trivial), then by status (Resolved, Open, New, On Hold, Invalid, Duplicate, Won't Fix) and finally by kind (Enhancement, Bug, Proposal). We do this to bring high impact issues to the top of the list.
- **Commit Word Frequency Breakdown:** Uses natural language processing to break down commit messages and count the number of times a word appears in a commit message. This lets the user know of frequently occurring keywords that may indicate what they worked on.
- **Commits:** Returns a list of all commits made by the user sorted by date.

3.3.4 GitHub

UNBIASED analyzes all issues, pull requests, milestones, labels, and commits where the user is concerned. It uses all of this information to figure out what the user is working on. The analysis can be broken down into two parts.

The *Issue / Pull Request Analysis* looks at all issues and pull requests the user has either created or worked on.

- **Issues Assigned:** A list of issues where the user is the assignee.
- **Pull Requests Assigned:** A list of pull requests where the user is the assignee.
- **Number Stats:** A count of the number of issues and pull requests the user has created.
- **Duration Stats:** We then compute an average duration spent on: all closed issues, all closed pull requests, and all closed issues and all closed pull requests combined.

The *Commit Analysis* consists of three parts: *Accomplishments*, *File Breakdown*, *Commit Word Frequency Breakdown*, and *Commits*.

- **Accomplishments:** List of accomplishments generated from GitHub commit messages. See Section 3.4 for more information on how this is achieved.
- **File Breakdown:** A list of files sorted by the number of times the user has modified it. This lets the user know what files they touched the most, which may indicate where most of their work has been spent.
- **Commit Word Frequency Breakdown:** Uses natural language processing to break down commit messages and count the number of times a word appears in a commit message. This lets the user know of frequently occurring keywords that may indicate what they worked on. For example, if the word *Fixed* appears more often than *Added*, this may indicate that the user has spent most of their time fixing bugs than adding features.
- **Commits:** A list of all commits made by the user sorted by date.

3.3.5 JIRA

The JIRA analysis can be performed on a per-project basis. Users can opt into limiting their search to a specific user. The analysis consists of five parts: *Issue stats*, *Issue time breakdown by issue type*, *Issue time breakdown by issue status*, and *All issues this user logged hours on.* Examples of the JIRA analysis can be seen in Figure 3.13 and Figure 3.14.

- **Issue stats:** Counts the number of issues the user has created, assigned, and reported. Also shows the total amount of hours logged by the user.
- **Issue time breakdown by issue type:** Reports the number of hours spent by the user by issue types (Story, Task, Sub-Task, Bug, Epic).
- **Issue time breakdown by issue status:** Reports the number of hours spent by the user by issue status (Done, In Progress).
- **All issues on which this user logged hours:** A list of all issues on which the user has logged work. Each issue has bullet points that represent the work logged by the user: number of hours the user has spent, the date, and a description of the work. Each issue's priority, status, issue type, and the total amount of hours spent by the user in that issue is clearly labeled. This is the information where a user can easily glean where most of their time was spent and what they accomplished during that time.
- **Evaluation:** Accomplishments, accomplishment details, personal strengths, and areas of improvement generated from JIRA data. See Section 3.4 for more information on how this is achieved.

3.4 Automatic Self Evaluation Generation

In this section, we discuss what parts of the self evaluation can be generated by UNBIASED. To generate a self evaluation, the user must authorize at least one of the following third party APIs: Bitbucket, GitHub, or JIRA. Below are the five self evaluation questions that UNBIASED supports (covered in Section 2.1.1). All of the questions besides QUESTION-4, which absolutely requires user input, can be automatically generated. An example of a generated self evaluation can be found in Figure 3.16.

- QUESTION-1: List each of your major personal accomplishments/jobs on the project this quarter.
- QUESTION-2: You may describe [some of] them in detail if you feel it's appropriate or useful.
- QUESTION-3: In what specific ways could/should you have contributed more to your team project? Explain without making excuses.
- QUESTION-4: Assign yourself a project grade.
- QUESTION-5: Explain the rationale for the grade and justify it with specific details. Do not repeat material from earlier sections.

GitHub and Bitbucket analysis can generate a response for QUESTION-1 and QUESTION-2 using the same methodology. We generate a list of accomplishments based off of commit messages and issue titles and descriptions.

JIRA analysis can generate a response for QUESTION-1, QUESTION-2, QUESTION-3, and QUESTION-5. With JIRA, we are able to analyze all individual contributions separately, and then perform a comparative analysis by comparing each individual

to the other members in the project to gain more useful insights. Theoretically, the same can be done for both GitHub and Bitbucket, but UNBIASED only has this implemented for JIRA. UNBIASED can extract both accomplishments (QUESTION-1) and details of those accomplishments (QUESTION-2) based off of issue titles, descriptions, and worklog comments. We attempt to answer (QUESTION-3) and (QUESTION-5) by highlighting the user's weaker and stronger contributions, respectively, in comparison to the contributions made by others in the same JIRA project. These contributions are based off of the following criteria: number of issues assigned, number of issues reported, number of issues created, and total numbers of hours logged. Rationale for a higher grade could be for having the highest value in any of these criteria. To identify ways that the user could have contributed more, we check whether the user's stats are more than one standard deviation below the mean of all team members' stats for a particular criteria. An example of the JIRA evaluation can be seen in Figure 3.15.

from:*@calpoly.edu OR to:*@calpoly.edu calpoly cal poly

Emails Received By

Email	# Emails
grad@calpoly.edu	6
computer-science@calpoly.edu	2
no-reply@piazza.com	1

Emails You Sent To

Email	# Emails
thesis-advisor@calpoly.edu	3

Word Frequency

	Word	# Occurrences
	thesis	3
	deadline	3
	reminder	3
	graduate	1

Figure 3.6: An example Gmail analysis.

Emails that contain "note"

Email
Instructor Franz J. Kurfess posted a new Note. Possible Spatial Cognition Tutorial on July 25/26 One
Instructor Franz J. Kurfess posted a new Note. Activities Week 7: May 10 & 12 Planned Lecture
Instructor Franz J. Kurfess posted a new Note. Make-up Questions for Quizzes If you want to improve
Instructor Franz J. Kurfess posted a new Note. TinMan Platform and Methodology for Real-time
Instructor Franz J. Kurfess posted a new Note. IBM Watson and Bluemix Examples for the use of
Instructor Franz J. Kurfess posted a new Note. Global Accessibility Awareness Day 2016: Thursday, May
Instructor Franz J. Kurfess posted a new Note. Schedule for A1 Team Presentations We'll do two
Instructor Franz J. Kurfess posted a new Note. Activities Week 6: May 3 & 5 Planned Lecture
Instructor Franz J. Kurfess posted a new Note. Internet of Things Panel Tuesday, May 10, 2016. 11 -
Instructor Franz J. Kurfess posted a new Note. Financial Markets Presentation on Thu, May 4, 6-7 pm
Instructor Franz J. Kurfess posted a new Note. Activities Week 5: April 26 & 28 Planned Lecture
Instructor Franz J. Kurfess posted a new Note. AI Nugget Proposals Graded I finished grading the AI
Instructor Franz J. Kurfess posted a new Note. Lab 5 available: AI in the real world Lab 5 is now
Instructor Franz J. Kurfess posted a new Note. Guest lecture on Thu, April 28 I'll be out of town
Instructor Franz J. Kurfess posted a new Note. Project Documentation - Overview Graded I just
Instructor Franz J. Kurfess posted a new Note. Activities Week 4 - April 19 & 21 Planned Lecture
Hi Jonathan: Your name is too long for the commencement bulletin and the note that may be receiving
Instructor Franz J. Kurfess posted a new Note. Piazza Groups for Project Teams I created groups on
Instructor Franz J. Kurfess posted a new Note. Submission of Project Documentation Please use the
Instructor Franz J. Kurfess posted a new Note. Activities Week 3 - April 12 & 14 Class Dates

Figure 3.7: An example Gmail analysis that shows when a user wants to see which emails contain a particular keyword.

Calendar Analysis

Summary Frequency

Summary	Time Spent (hrs)	# Occurrences
Career Fair	9.50	2
Weekly Meeting	4.00	4
Monthly Meeting	2.00	1

Summary Word Frequency

Word	Time Spent (hrs)	# Occurrences
career	9.50	2
fair	9.50	2
meeting	6.00	5
weekly	4.00	4
monthly	2.00	1

Figure 3.8: An example Google Calendar analysis.

Commits

Below are a list of commits you have contributed for this repo.

Tip: If you do not see all of your commits, check that your commits on bitbucket.org are mapped to you. This may happen if you made commits under different usernames (look up Bitbucket Unmapped Users).

Date	Message
2016-03-03	Updated category spinner UI. Sets default to be whatever is in the icap value at that time.
2016-02-25	Added new app icon.
2016-02-25	Merge branch 'master' into jon-development Conflicts: app/build.gradle
2016-02-25	May fix KitKat crash https://medium.com/@chrisbanes/appcompat-v23-2-age-of-the-vectors-91cbaf a87c88#.xj04zpxfg
2016-02-24	Added simple dialog to tell the user we are trying to save their ICAP entry.

Figure 3.9: An example Bitbucket commit analysis.

Issues

Below are a list of issues you are responsible for. They are sorted in order of priority, status, and kind.

Date	Title	Priority	Status	Kind
2016-01-31	ListView: Remove divider lines on ClassListActivity:	major	resolved	enhancement
2016-01-31	ICAP: Instructions have small text on tablets.:	major	resolved	enhancement
2016-01-31	ICAP Activity: change Title to be class name.:	major	resolved	enhancement
2016-01-31	Add FAB for ICAP page.:	major	resolved	enhancement
2016-02-02	New ICAP entry: Add padding so the touch targets are clearer between the bottom row of radio buttons and the save button.:	major	resolved	enhancement
2016-02-02	New UI for adding a new ICAP entry.: * "Table" with editable fields. * Allow users to "toggle" between this version and the original version.	major	resolved	enhancement

Figure 3.10: An example Bitbucket issue analysis.

Issue / Pull Request Stats

Issues Assigned

Issues	State
Add weather based on location	open
Add feature for due dates	open

Pull Requests Assigned

Issues	State
Fixed all the bugs.	closed

Issues created	4
Pull requests created	1
Average Duration spent on all closed issues	0 hours
Average Duration spent on all closed pull requests	3 hours
Average Duration spent on all closed issues and closed pull requests	0 hours

Issue / Pull Request Breakdown

Milestone	# Issues
None	4

Label	# Issues
enhancement	2

State	# Issues
open	4

Figure 3.11: An example GitHub issue and pull request analysis.

File Breakdown

Filename	# Times Updated
js/app.js	18
index.html	17
css/app.css	13

Commit Word Frequency Breakdown

Word	# Occurrences
added	10
refactor	7
ui	3
bug	2
storage	2
fix	2

Commits

Date	Commit	Stats
03/15/16	UI changes. Removed background-based-on-time code.	+38 -84
03/11/16	Fix bug associated with data-done-time. The result was being wrapped in double quotes.	+11 -7
03/09/16	Added min-widths and stylized add button.	+16 -5
03/09/16	UI changes. Refactored CSS into own file.	+60 -42
03/09/16	Refactor code around.	+26474 -73
03/09/16	Added a way to add/save (locally) new items to different lists.	+82 -19

Figure 3.12: An example GitHub commit analysis.

Jon Miranda (jonmiranda)

Issue stats

# Issues Created	21
# Issues Assigned	22
# Issues Reported	0
Time Logged in all issues	89.00 hrs

Issue time breakdown by issue type

Issue Type	Time Spent (hrs)
Story	86.50
Bug	2.50

Issue time breakdown by issue status

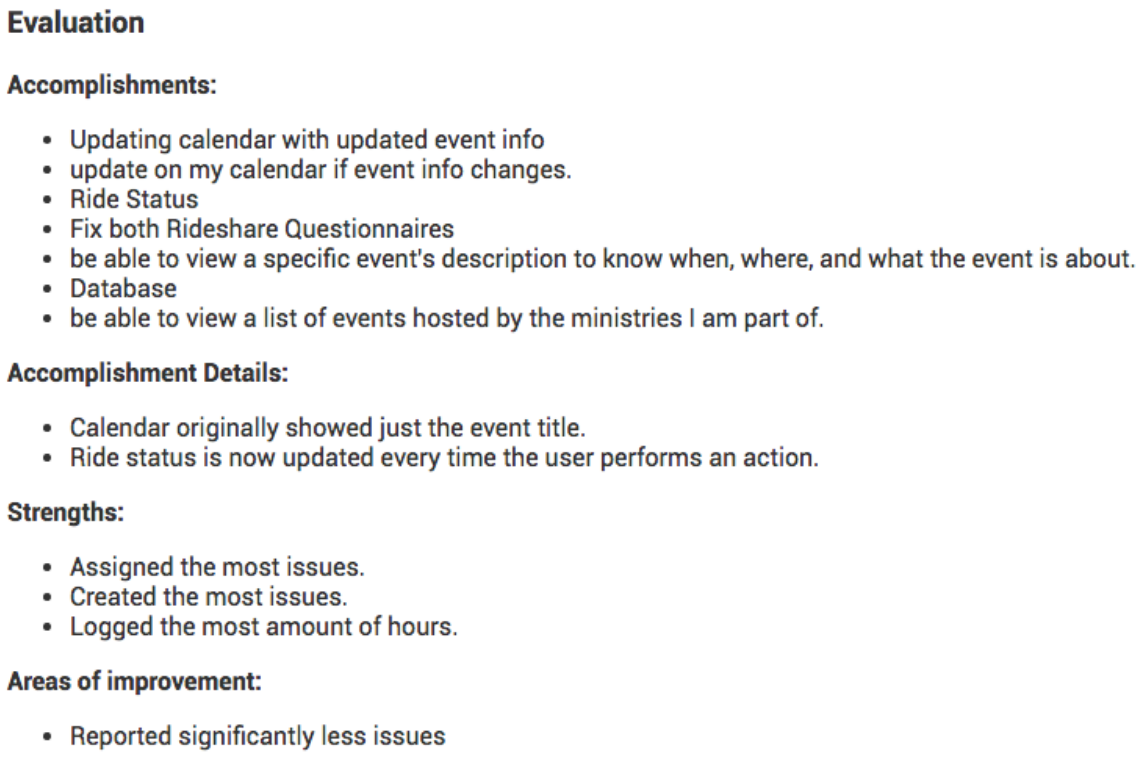
Issue Status	Time Spent (hrs)
Done	74.00
To Do	15.00

Figure 3.13: An example JIRA issue analysis for Issue Stats, Issue time breakdown by issue type, and Issue time breakdown by issue status.

All issues this user logged hours on

Issue	Priority	Status	Issuetype	Time Spent (hrs)
Updating calendar with updated event info: As a user, I want my events to update on my calendar if event info changes. • 15.00 hrs spent on 2016-03-03 - No comment	Major	To Do	Story	15.00
Selecting RideShare tab crashes the app: App shows a list of events (defaulted to the events tab) • 0.50 hrs spent on 2016-04-11 - When grabbing data from database, not all the tuples have the same format. The crash was due to unable to grab the street field. I fixed the bug by checking if the tuple consists of a street attribute before trying to grab the value, rather than expecting a street field in the location.	Critical	Done	Bug	0.50
View Questionnaire For Driver: As a user, I want to be able to offer a ride to an event. • 2.00 hrs spent on 2016-02-04 - No comment	Major	Done	Story	2.00
View List of Drivers: Include creating the UI and setting it up to grab from database. • 2.50 hrs spent on 2016-02-04 - No comment	Major	Done	Story	2.50

Figure 3.14: An example JIRA analysis that lists all the issues on which the user has logged work.



Evaluation

Accomplishments:

- Updating calendar with updated event info
- update on my calendar if event info changes.
- Ride Status
- Fix both Rideshare Questionnaires
- be able to view a specific event's description to know when, where, and what the event is about.
- Database
- be able to view a list of events hosted by the ministries I am part of.

Accomplishment Details:

- Calendar originally showed just the event title.
- Ride status is now updated every time the user performs an action.

Strengths:

- Assigned the most issues.
- Created the most issues.
- Logged the most amount of hours.

Areas of improvement:

- Reported significantly less issues

Figure 3.15: An example JIRA analysis where UNBIASED can glean accomplishments, accomplishment details, strengths, and areas of improvement to populate the self evaluation form.

Enter your Cal Poly username.

jonmiranda

1. List each of your MAJOR personal accomplishments/jobs on the project this quarter.

// Generated from JIRA
Add events to Google calendar
UI changes
Add libraries

// Generated from GitHub
store and retrieve our movie data
Add Palette, Realm, Picasso, Retrofit to build

2. You may describe [some of] them in detail if you feel it's appropriate or useful.

// Generated from JIRA
Google Calendar events were not synced.
Added third party support librariess that made it easier for the entire team to develop.

3. In what specific ways could/should you have contributed more to your team project? Explain without making excuses.

// Generated from JIRA
Reported significantly less issues

4. Assign yourself a project grade for your TEAM PROJECT ONLY. Note: do not consider reading and homework assignments; this grade is just for your contributions to the team project.

Grade

A

5. Explain the rationale for the grade and justify it with specific details. Do not repeat material from earlier questions.

// Generated from JIRA
Assigned the most issues
Logged the most amount of hours.

Figure 3.16: An example self evaluation generated from JIRA and GitHub data. Note that the generation comments were left in for clarity but it is expected that users will remove those comments.

Chapter 4

UNBIASED SOFTWARE DESIGN

This chapter discusses the software design of the UNBIASED application. UNBIASED is written in Python on Google App Engine using a variety of frameworks and libraries. Section 4.1 discusses a high level overview of the application. Section 4.2 briefly discusses the technologies used in building the application.

4.1 Overview

In a very high level overview, the user makes a request to the UNBIASED system, the UNBIASED system interacts with third party API(s), and then the UNBIASED system returns a response back to the user. A simple deployment diagram of the UNBIASED system can be seen in Figure 4.1.

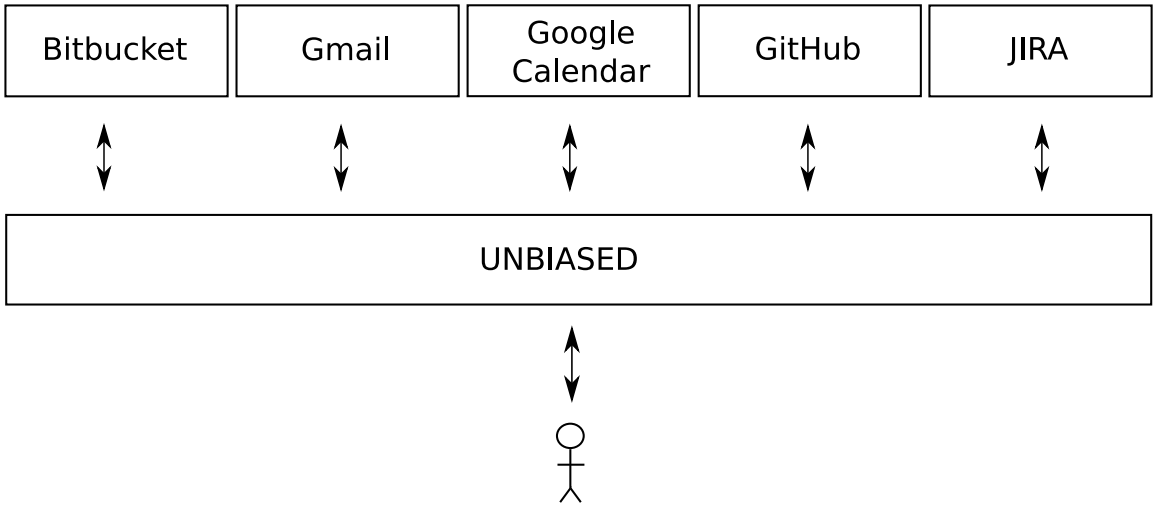


Figure 4.1: Simple deployment diagram of the UNBIASED system.

On the server, each third party API is split into three components: Service, Analyzer, and Handlers. A diagram of the interaction between these three components

can be seen in Figure 4.2.

- **Service**: Responsible for directly connecting with the third party API.
- **Analyzer**: Responsible for analyzing third party API data using the corresponding Service to gather the data needed.
- **Handlers**: Each Handler corresponds to a REST end point of the server. Some Handlers are responsible for interacting with UNBIASED's datastore, and others are responsible for interacting with Services and Analyzers.

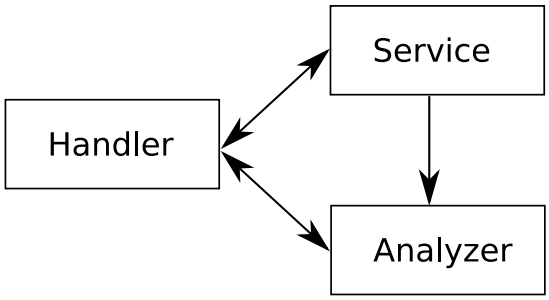


Figure 4.2: Diagram of the interaction between Handler, Service, and Analyzer.

4.2 Technologies Used

This section briefly discusses the technologies used in building the UNBIASED system.

4.2.1 Google App Engine

Google App Engine (GAE) is a platform as a service (PaaS) for developing and hosting web applications [10]. GAE does a lot of the grunt work of running a web application: scaling, load balancing, authorization, memcache, logging, search, and versioning control, just to name a few.

Other PaaS options we considered were Heroku and Amazon Elastic Compute Cloud. Each of these offer many of the same features, but we chose Google App Engine for the easiness of accessing the rest of Google services, and the reliability of running on Google's servers.

Jinja2

Jinja2 is a templating language for Python [21]. This thesis uses Jinja2 to display server side responses to the frontend. Jinja2 makes it easy to build reusable HTML templates. Jinja2 allows conditionals and loop logic.

Google Cloud Datastore

Google Cloud Datastore is a NoSQL document database built for automatic scaling, high performance, and ease of application development. While the Datastore interface has many of the same features as traditional databases, as a NoSQL database it differs from them in how it describes relationships between data objects [12]. We use Google Datastore Network Database (NDB) Client Library [20] to interact with the Cloud Datastore. NDB handles automatic caching using an in-context cache and Memcache.

4.2.2 Google Polymer

Google Polymer is a new library that makes it easier to make fast, beautiful, and interoperable web components [19]. Modern web applications are expected to be beautiful, fast, and responsive, and Polymer takes care of most of the work and provides boilerplate code. Polymer offers features such as one-way and two-way data bindings between code and HTML elements. This thesis makes use of Polymer's material design components [13], a full range of icons, two-way data binding, and HTML elements to handle AJAX requests.

4.2.3 OAuth 2.0

OAuth 2.0 provides authorization flows for applications [18]. The benefit of using OAuth 2.0 is getting access to data on a user's behalf, without requiring access to the user's username and password. This thesis uses OAuth 2.0 to interact with users' Google, BitBucket, and JIRA accounts.

4.2.4 Natural Language Toolkit

The Natural Language Toolkit (NLTK) is a Python library that allows developers to work with processing language data [16]. This thesis uses a special fork of the project developed by GitHub user *rutherford* which enables us to use NLTK on Google App Engine [17].

This thesis utilizes the NLTK tokenizer and the part of speech tagging. A tokenizer breaks a string (or sentence) into tokens, or sequences of characters that have a collective meaning. In a sentence, each word can be treated as a token. Speech tagging involves iterating through parts of a sentence and identifying words as nouns, verbs, adjectives, etcetera.

4.2.5 urlfetch

urlfetch is a Google App Engine library that allows for asynchronous HTTP requests. Prior to using this library, all HTTP requests were synchronous, which meant that each GitHub commit or JIRA issue was downloaded one after another. Using this library, we observed a 5x speed up in completing a GitHub analysis, and a 3x speed up in completing a JIRA analysis. Tables 4.1, 4.2, 4.3, and 4.4 report the time data comparisons with the different third party APIs.

Table 4.1: JIRA Time Data for Sequential and Batch Requests

Sample #	Sequential (seconds)	Batch (seconds)
1	150.6	41.76
2	148.2	49.54
3	149.0	52.92
4	147.3	52.31
5	149.9	51.98
Average (seconds)	149.0	49.70

Table 4.2: GitHub Time Data for Sequential and Batch Requests

Sample #	Sequential (seconds)	Batch (seconds)
1	162	24.05
2	168	35.83
3	168	35.38
4	162	33.05
5	168	32.27
Average (seconds)	165.6	33.12

Table 4.3: Gmail Time Data for Sequential and Batch Requests

Sample #	Sequential (seconds)	Batch (seconds)
1	10.64	29.95
2	8.08	22.21
3	8.06	31.62
Average (seconds)	8.93	27.93

Table 4.4: Google Calendar Time Data for Sequential and Batch Requests

Sample #	Sequential (seconds)	Batch (seconds)
1	9.77	3.49
2	7.03	3.55
3	6.80	3.15
Average (seconds)	7.86	3.40

4.2.6 apiclient.http

Google provides their own client library for performing batch HTTP requests on Gmail and Google Calendar. Prior to using this library, each Gmail message and Google Calendar event were downloaded sequentially. Based on our measurements, using this library has shown a 3x speed up when conducting a Gmail or Google Calendar analysis.

Chapter 5

VALIDATION

The validation framework of UNBIASED will be discussed in this chapter.

5.1 Validation Framework

Testing and validation was performed in two of Dr. Janzen's Capstone Courses (CPE 406 Software Deployment) in Spring 2016. We determined which sections would be the Control Group and Testing Group by a flip of a coin. We then constructed three different surveys:

- A **Pre-Evaluation Survey** that asks questions regarding their self and peer evaluations in Winter 2016.
- A **Post-Evaluation Survey for the Control Group** that asks the same questions as the Pre-Evaluation, but with questions related to Spring 2016.
- A **Post-Evaluation Survey for the Testing Group** that is the same as the Post-Evaluation Survey for the Control Group, but with additional questions related to using the UNBIASED system.

Using these surveys, we are able to answer the following questions:

- Does using UNBIASED save software engineers time?
- Does using UNBIASED help software engineers perform more accurate self evaluations?
- Does using UNBIASED increase the number of sources software engineers use when performing a self evaluation?

- Does using UNIBASED make it easier for software engineers to write their self evaluations?
- Does using UNBIASED increase the quality of the self evaluation?
- Do software engineers like using UNBIASED?

Both sections were given a Pre-Evaluation Survey, and both sections received their own version the Post-Evaluation Survey. Only the Testing Group was instructed to use the UNBIASED system between completing the Pre-Evaluation Survey and Post-Evaluation Survey. The surveys and research protocol were reviewed and approved by the Cal Poly Human Subjects Commmittee.

5.2 2016 Capstone Pre-Evaluation Survey

The 2016 Capstone Pre-Evaluation Survey that both the Control and Testing Groups took can be found in Appendix A.

The relevant questions are listed below for convenience:

- How much time do you think you spent on your self/peer evaluation in Winter 2016?
  - less than 10 minutes
  - 10-20 minutes
  - 20-30
  - 30-45
  - 45-60
- Which of the following did you search/review while completing your self/peer evaluation in Winter 2016? Check all that apply.

  - JIRA
  - Bitbucket or GitHub
  - Gmail
  - Google Calendar
  - Cal Poly Email
  - Slack
  - Facebook
  - Other (Fill in blank)

- On a scale from 1 (Easy) to 5 (Difficult), how difficult was it to write your self evaluation in Winter 2016?
- On a scale from 1 (Not very accurate) to 5 (Very accurate), how accurate (i.e. evaluation matched actual accomplishments) do you think was your self evaluation in Winter 2016?

5.3 2016 Capstone Post-Evaluation Surveys

5.3.1 2016 Capstone Post-Evaluation Survey for Control Group

The full 2016 Capstone Post-Evaluation Survey that the Control Group took can be found in Appendix B. This survey is identical to the 2016 Capstone Pre-Evaluation Survey, but with questions related to the current quarter of Spring 2016 instead of the previous quarter of Winter 2016.

5.3.2 2016 Capstone Post-Evaluation Survey for Testing Group

The full 2016 Capstone Post-Evaluation Survey that the Testing Group took can be found in Appendix C. This survey is identical to the 2016 Capstone Post-Evaluation

Survey for Control Group, but with additional questions. The additional questions are listed below for convenience:

- How much time do you think you spent on your self/peer evaluation in Spring 2016?
  - less than 10 minutes
  - 10-20 minutes
  - 20-30
  - 30-45
  - 45-60
- Which of the following did you search/review within the UNBIASED Self Evaluation System while completing your self/peer evaluation in Spring 2016?
  - JIRA
  - Bitbucket or GitHub
  - Gmail
  - Google Calendar
  - Cal Poly Email
  - Slack
  - Facebook
  - Other (Fill in blank)
- Outside of using the UNBIASED Self Evaluatino System, which of the following did you search/review while completing your self/peer evaluation in Spring 2016?
  - JIRA

  - Bitbucket or GitHub
  - Gmail
  - Google Calendar
  - Cal Poly Email
  - Slack
  - Facebook
  - Other (Fill in blank)

- Do you think you wrote a better, same, or worse self evaluation because you used the UNBIASED Self Evaluation System in completing your self evaluation in Spring 2016 (406)?
  - This eval was better than the past.
  - This eval was about the same quality as the past.
  - This eval was worse than the past.

- On a scale from 1 (Not likely) to 5 (Very likely), if you had to complete another self evaluation in the future, how likely would you want to use the UNBIASED Self Evaluation System again?

- Tell us what you think about the UNBIASED Self Evaluation System. What did you like? What did you not like?

5.4 Results

Of the 28 students in the Testing Group, 28 performed the Pre-Evaluation Survey, 25 performed the Self Evaluation, and 20 performed the Post-Evaluation Survey. Of the 28 students in the Control Group, 27 performed the Pre-Evaluation Survey, 28

performed the Self Evaluation, and 22 performed the Post-Evaluation Survey. The raw survey results can be found in Appendix D.

5.5 Analysis

To begin our analysis, we answer the questions outlined in the beginning of this chapter.

5.5.1 Does using UNBIASED save software engineers time?

In the Testing Group, we asked students the amount of time it took to perform a self evaluation using UNBIASED. As seen in Figure 5.1, 50% reported spending less than 20 minutes, 80% reported spending less than 30 minutes, 20% reported spending 30-45 minutes, and no respondents reported spending more than 45 minutes on their self evaluation. When comparing the amount of time taken to perform a self evaluation for Spring 2016 compared to Winter 2016, 72% of the Control Group reported spending the same or less amount of time. 70% of the Testing Group reported spending the same or less amount of time. See Figure 5.2. The 2% difference in favor of the Control Group is not significant, considering that the Testing Group is using a new and unfamiliar tool, and that they also reported using more sources (discussed in detail below). The users also encountered errors when trying to connect to the different API services; for example, the Cal Poly Computer Science JIRA instance went down when everyone hit the server at once.

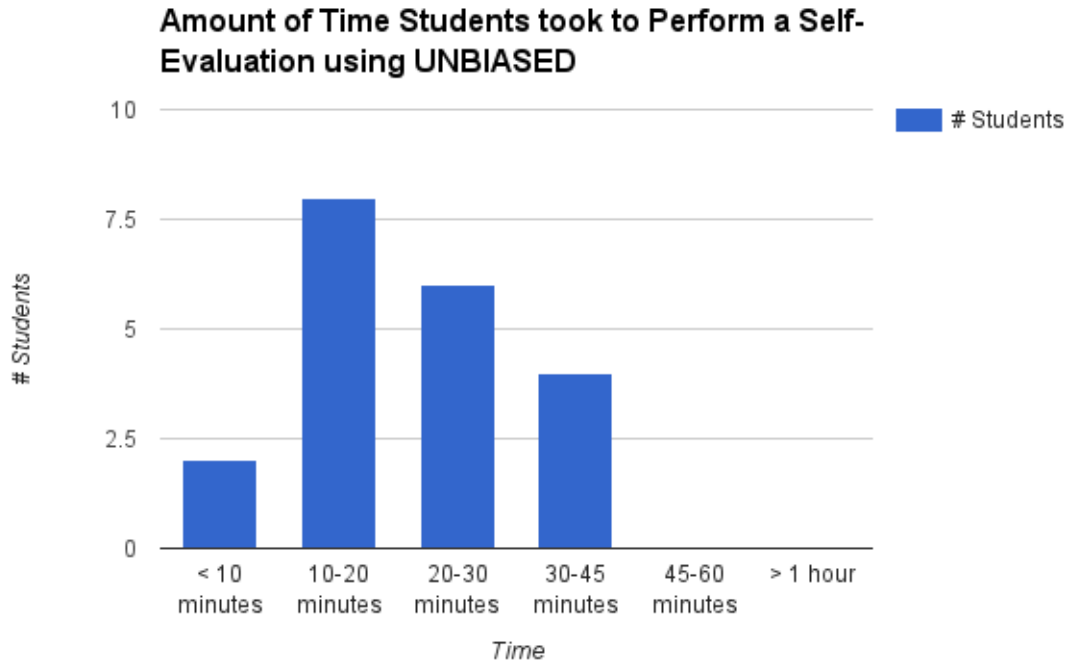


Figure 5.1: Survey results for the amount of time the Testing Group spent performing self evaluations in Spring 2016.

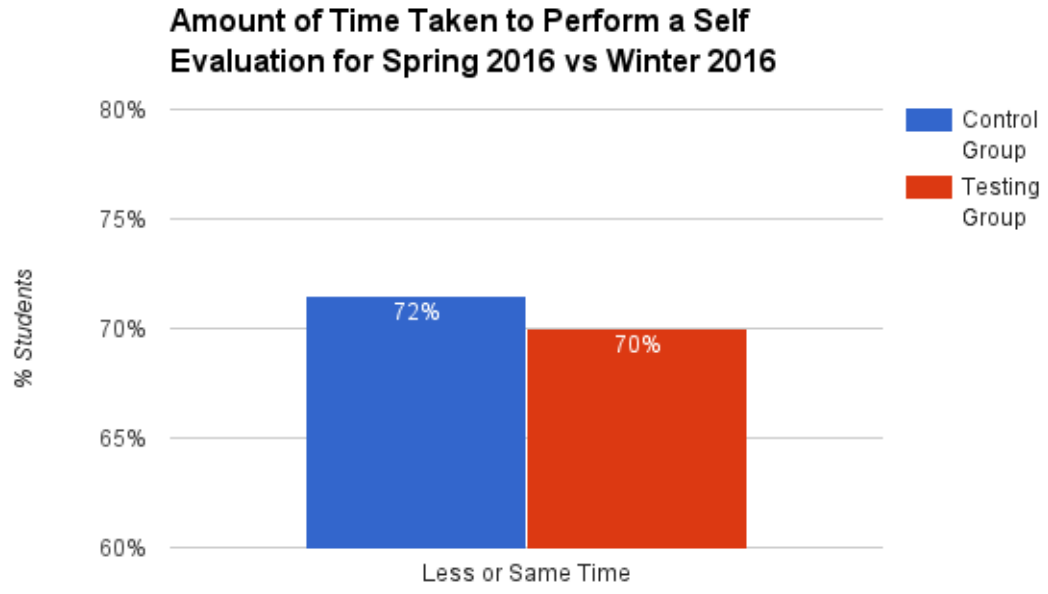


Figure 5.2: Survey results comparing the amount of time spent performing self evaluations in Spring 2016 vs Winter 2016.

5.5.2 Does using UNBIASED help software engineers perform more accurate self-evaluations?

In the Testing Group, we asked students to rank the accuracy of their self evaluations on a scale from 1 (*Not very accurate*) to 5 (*Very accurate*). 90% of the respondents gave a rating of 3 or higher, with 45% giving a rating of 4, and 15% giving a rating of 5. These are impressive results. Now when comparing the accuracy rating when performing a self evaluation for Spring 2016 compared to Winter 2016, 77% of the Control Group reported their evaluations as having either the same accuracy or more accuracy. There is only a 3% difference when compared to the 80% of the Testing Group who reported the same. Because the difference is so low, we cannot say that using UNBIASED has significantly helped software engineers perform more accurate self evaluations. See Figures 5.3 and 5.4.

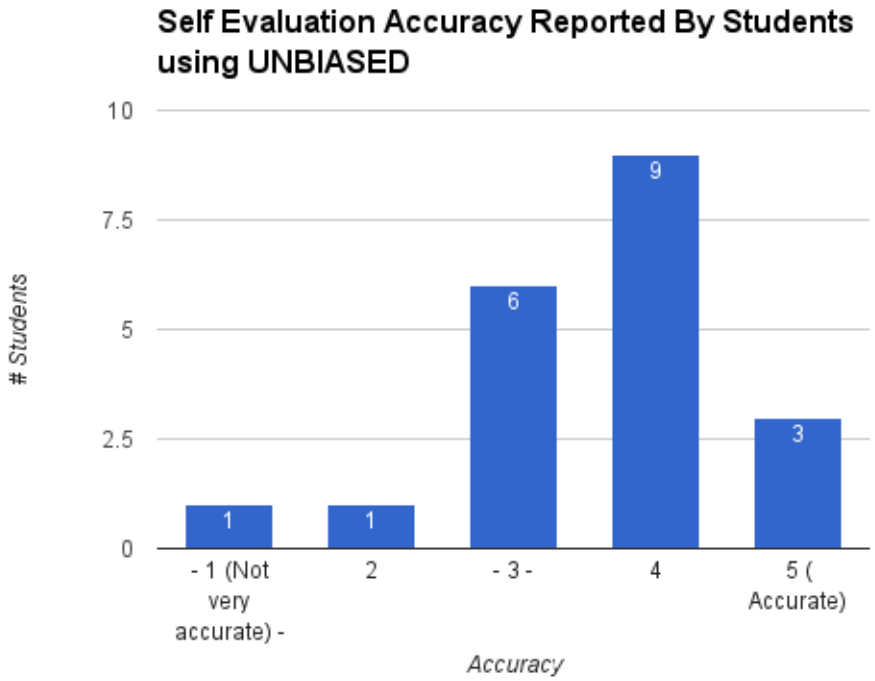


Figure 5.3: Survey results for how accurate the Testing Group ranked their self evaluations in Spring 2016.

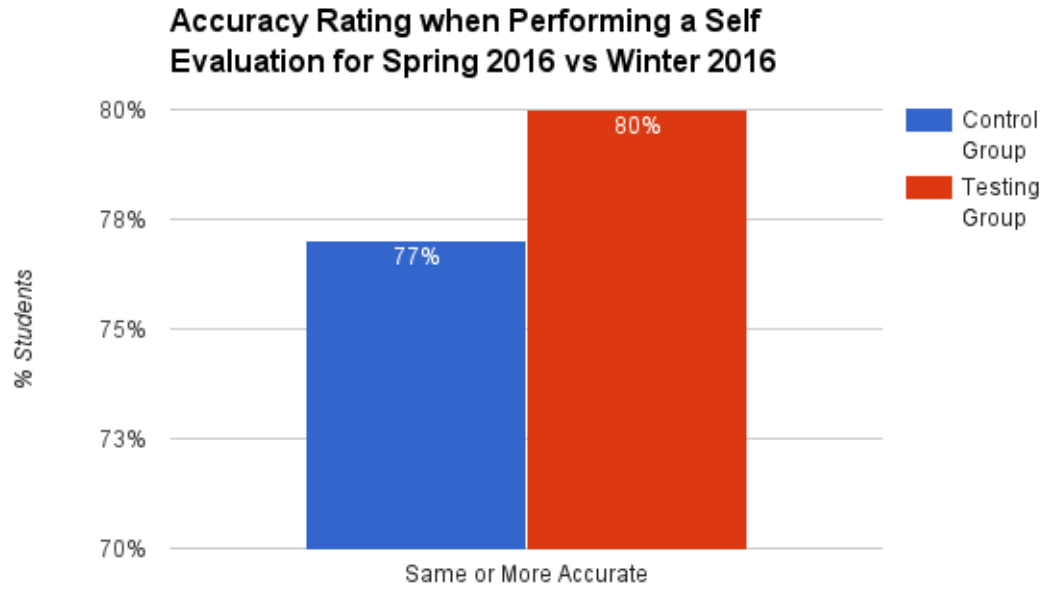


Figure 5.4: Survey results comparing the accuracy of the self evaluations in Spring 2016 vs Winter 2016.

5.5.3 Does using UNBIASED increase the number of sources software engineers use when performing a self evaluation?

In the Testing Group, we found that 16 respondents used JIRA as a source, and 17 respondents used Bitbucket or GitHub. No respondents used Gmail or Google Calendar when performing their self evaluation. Looking at the pre-survey results, no respondents used Gmail in previous self evaluations, and only one respondent in the Testing Group used Google Calendar. Based on feedback from the respondents, there didn't seem to be a lot of trust in the system to handle their sensitive email content. Additionally, students reported using instant messaging communication applications rather than communicating over email. When looking at the raw number of sources used when performing a self evaluation for Spring 2016 compared to Winter 2016, we found that 80% of the testing group said they used the same or higher number of

sources, compared to only 60% of the Control Group. See Figures 5.5 and 5.6.

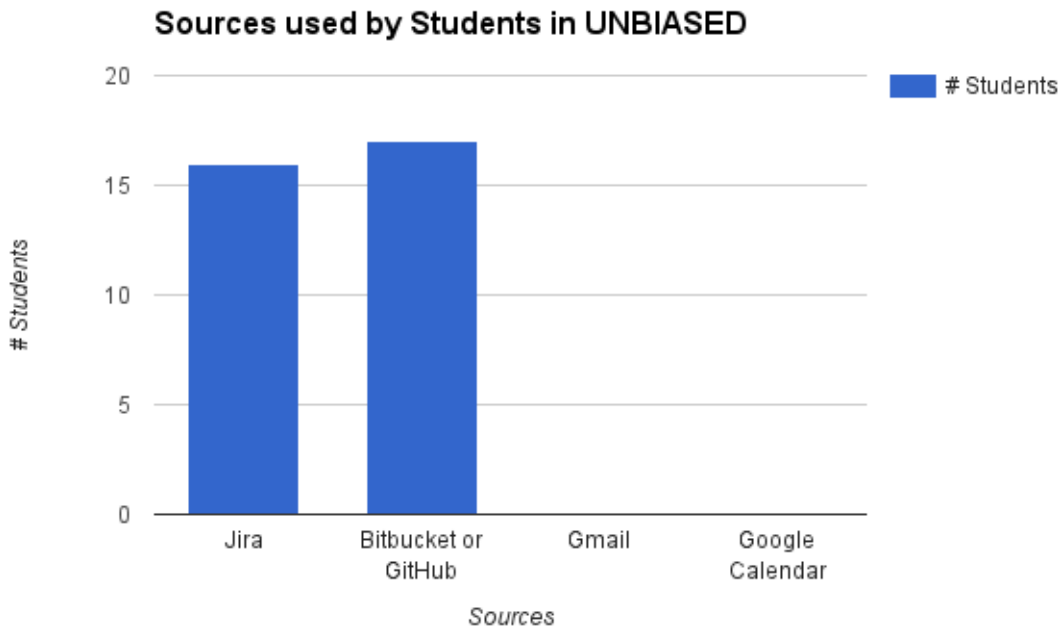


Figure 5.5: Survey results for what sources the Testing Group used when performing their self evaluations in Spring 2016.

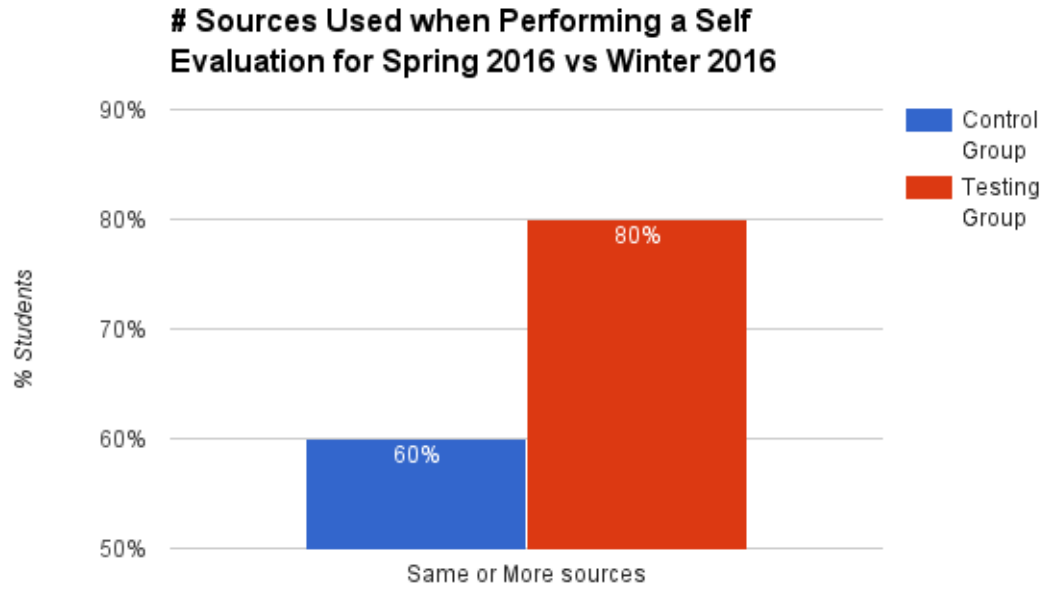


Figure 5.6: Survey results comparing the number of sources used when performing self evaluations in Spring 2016 vs Winter 2016.

5.5.4 Does using UNBIASED make it easier for software engineers to write their self evaluations?

In the Testing Group, we asked students to rate how difficult it was to perform their self evaluation using UNBIASED on a scale from 1 (*Easy*) to 5 (*Difficult*). 0% of the respondents reported having a Difficult (5) time, and 80% of the respondents rated a 3 or lower. When comparing the difficulty reported to perform a self evaluation for Spring 2016 compared to Winter 2016, we found two interesting results. Only 57% of the Control Group reported having the same or less difficulty, compared to the expected 100% because the Control Group would not experience any change in difficulty because they are using the same exact tool as they did in Winter 2016. The second interesting result is that 70% of the Testing Group reported having the same or less difficulty, which is 23% more than the Control Group. See Figures 5.7 and 5.8.

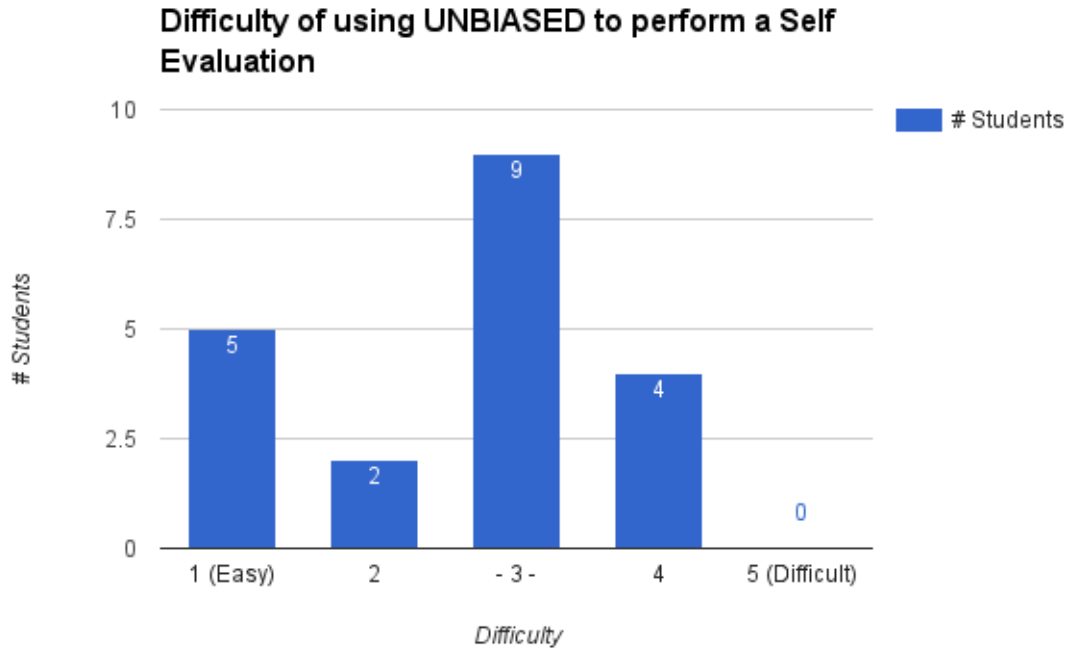


Figure 5.7: Survey results for how difficult it was for the Testing Group to perform their self evaluations in Spring 2016.

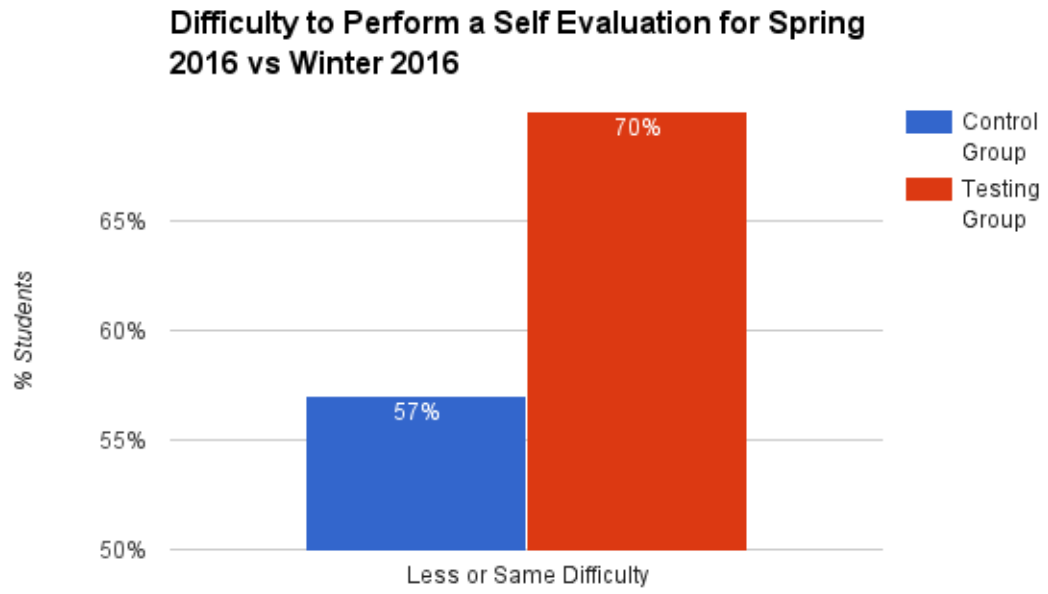


Figure 5.8: Survey results comparing the difficulty of performing self evaluations in Spring 2016 vs Winter 2016.

5.5.5 Does using UNBIASED increase the quality of the self evaluation?

As a metric for quality, we are using the average number of words per response to each question in both the Testing Group and the Control Group. For this analysis, we grouped QUESTION-1 and QUESTION-2 together and found that the Testing Group has 19 more words on average. The Control Group beats the Testing Group with a difference of about 7 words for their responses to both QUESTION-3 and QUESTION-5. Looking at the response to the self evaluation as a whole, the Testing Group writes 5 more words on average than the Control Group. See Figure 5.9.

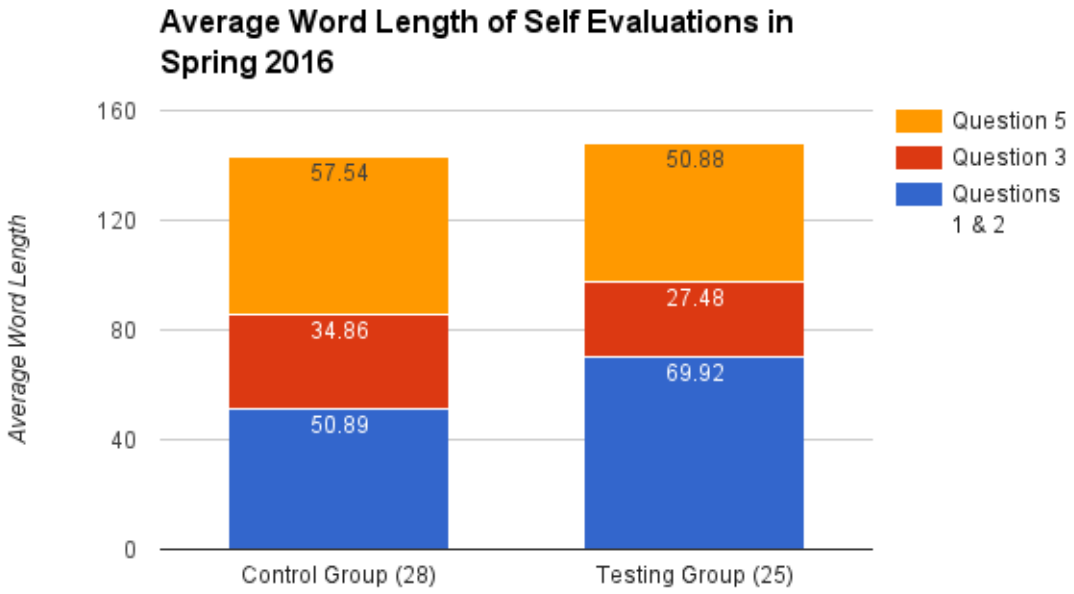


Figure 5.9: Average word length of Self Evaluations performed in Spring 2016.

5.5.6 Do software engineers like using UNBIASED?

The software engineers in the Testing Group reported both positive and negative reviews of the UNBIASED system. Our Post-Evaluation survey explicitly asks the Testing Group how likely they are, on a scale from 1 (*Not likely*) to 5 (*Very likely*),

they would want to use the UNBIASED system in future evaluations. Only 28.5% reported a 4 or 5, and 57.1% said 1 or 2 (see Figure 5.10). This is a negative result, but based on the feedback received it seems as if a lot of the problems users had with UNBIASED can be remedied. The raw feedback can be found in Appendix D.

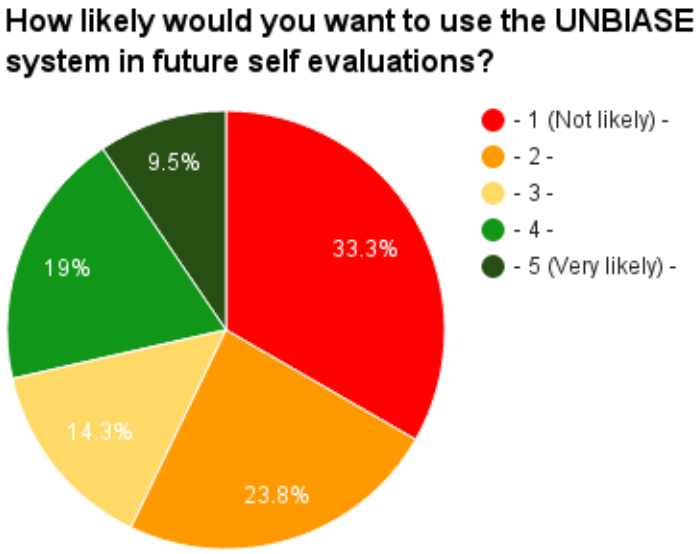


Figure 5.10: Survey results for how likely the Testing Group would want to use UNBIASED again in the future.

Students 50, 58, 58, 66, 67, 69, and 73 liked the idea, concept, and simplicity of the UNBIASED software. Student 66 noted that "centralizing the sources could be useful" and "the idea of streamlining the process... may contribute to more accurate analysis." Student 58 liked that it made the "self evaluations objective rather than subjective." Student 68 said that the "auto generation was really nice and was pretty accurate in displaying contributions." Student 72 said that the tool did not seem necessary for them but it would be "fine if it was used to evaluate peoples grades for someone not [in] the group." Student 76 said they mostly remember what they've done in the past two months, but "it's also nice to get a recap of what I've done."

The negative reviews can be summarized as being buggy, confusing, unhelpful, not needing the system, and not trusting the system. Many respondents (Students 51, 54, 63, 65, 66, 67, 71, 76, 77) reported having problems with using UNBIASED. When the users were first instructed to use UNBIASED, the Cal Poly instance of JIRA crashed for a few hours and this caused problems for UNBIASED. Students 50 and 54 reported that they were confused when using the application. Students 63 and 69 reported that the GitHub pull requests were of pull requests they merged and not created, which is a bug in the system. Student 58 said that UNBIASED "requires permissions to [their] accounts that make [them] uncomfortable." This may be due to a confusion where when you first open the application, a Google App Engine library asks for access to the user's Google email address. Some users interpreted this as asking for access to read the user's email and calendar data but in reality it only asks for the user's email address.

There was a lot of constructive feedback and many ideas were offered as well. Student 58 noted that they "did not document all [their] accomplishments throughout the quarter since I did not know I'd be using this." Student 77 expressed that "there are several issues that [they] forgot to log hours on, but still resolved" and noted that UNBIASED should display "Issues resolved by a particular user." Student 56 noted that UNBIASED does not "see the local development that may not have been completed or sent up to the software repository." Student 68 said that UNBIASED was not useful for instances of pair programming where only one engineer has the work credited to their name.

Chapter 6

RELATED WORK

This chapter discusses different tools and applications that perform an analysis on data from Bitbucket, Gmail, GitHub, Google Calendar, and JIRA.

6.1 Bitbucket Analysis Tools

This section discusses one Bitbucket analysis tool.

6.1.1 Awesome Graphs for Bitbucket

Awesome Graphs [2] for Bitbucket is an application that makes graphs and charts to visualize contribution statistics in Bitbucket repositories. Awesome Graphs is used to evaluate a team's performance and to get useful data to make teams more efficient. To do this, Awesome Graphs analyzes a specific repository and analyzes the commits made over the last year grouped by week. The tool uses interactive charts that can display more detailed data once expanded. Another analysis by Awesome Graphs is commit frequency, which counts which hours of the week have been the most productive for each team member. They do this by showing the commit frequency on each day of the week, at hourly time intervals.

6.2 Gmail Analysis Tools

This section discusses one Gmail analysis tool.

6.2.1 Gmail Meter

Gmail Meter [9] is a Google Apps Script that sends you an email containing different statistics about your email each first day of the month. Gmail Meter gives users different types of statistics that will help them analyze their Gmail habits. One nice feature of Gmail Meter is that because the script is run using Google Apps Script, Gmail data never leaves the Google ecosystem. The Gmail Meter analysis counts the number of conversations, the number of emails sent, the number of emails received, and the number of emails trashed. It counts the number of conversations marked as important and marked as starred. It also computes what percentage of emails were sent directly to the user and the percentage of how many times the user replied to those emails. The analysis includes the top five senders and the top five recipients. The analysis includes an area chart that shows the average flow of emails separated by time. It uses a pie chart to display what percentage of emails are in the inbox, in labels, in archive, or in trash. Gmail Meter also uses a bar chart to show the *time before first response* which says how long it takes, on average, for the user to reply and for the user to receive a reply. Another stat in a bar chart is the average length of your messages in both emails received and in emails sent.

6.3 Calendar Analysis Tools

This section discusses one Google Calendar analysis tool.

6.3.1 GTimeReport

GTime Report [14] is an analysis tool for Google Calendar. It allows users to select what calendars they want to analyze. The analysis then breaks down events by calendar, event name, the weekday, date, start time, duration, location, and description.

The analysis also shows the event attendees and indicates which of them are accepted, tentative, or rejected. One interesting analysis they perform is the merging of similar events by combining items that represent the same project or task. To perform the merge, the analysis requires one of two conditions to be met: the event titles are the same, or the event titles are the same up to the first colon symbol.

6.4 GitHub Analysis Tools

This section discusses three GitHub analysis tools.

6.4.1 GitHub Pulse

GitHub has their own repository analysis tool called Pulse [4]. Pulse shows all active committers and recent changes in a project's default branch. The analysis counts the number of active pull requests and splits them into merged pull requests and proposed pull requests. It also counts the number of active issues and splits them into closed issues and new issues. Pulse provides a way to see unresolved discussions, so users can see what still needs to be done. Pulse provides a nice summary of its results. An example summary is as follows: "67 authors have pushed 3,423 commits to all branches, excluding merges. On master, 2,051 files have changed and there have been 126,228 additions and 25,867 deletions." Pulse is a nice tool because it does all the work for the user automatically.

6.4.2 Git Commit Log Analysis Made Easy

Git Commit Log Analysis Made Easy (GCLAME) is a Git log analysis tool from eazyBi [5]. GCLAME uses charts and visualizations to bring attention to meaningful stats. Their analysis includes git changes by quarter and includes charting total

lines, number of commits, changes per commit, number of additions, and number of deletions. There is also an analysis that is performed based on the hour of the day: number of changes, commits, and changes per commit. These stats are helpful to software engineers that want to learn when and where (with regards to files) they do the most amount of work, but the stats are not necessarily useful for filling out a performance-based self evaluation. The way this tool works is that the user has to go on the command line and export their Git log and upload it to the service. After analysis, they allow exportation of the data via CSV, Excel, and other outputs.

6.4.3 Git Inspector

Gitinspector is another analysis tool for Git repositories [7]. This tool shows various statistics on a per author basis in the form of tables and chart visualizations. From the stats, authors are able to see their workload and activity in a commit. This tool allows filetype filtering, to ignore certain file types like xml or json files. One interesting metric they have is the number of lines of code in the repository that the author has written that are still in the current revision. This has the potential for indicating the quality of the code the author writes, but there is a lot of potential for bias as well. To use this tool, users have to download the application, edit a configuration file, and run a command on the command line.

6.5 JIRA Analysis Tools

This section discusses one JIRA analysis tool.

6.5.1 JIRA Reports

JIRA provides in-house agile reports, issue analysis reports, and forecast and management reports. Agile reports help track the project as a whole. The issue analysis reports analyzes issues on a per project basis. The forecast and management reports track time estimates for remaining issues based on certain criteria like project and user. These built-in reports *can* be helpful to software engineers performing a self evaluation, but none are especially useful for examining an individual's contribution to the project.

6.6 Discussion

UNBIASED takes inspiration and builds upon the ideas of many of these tools with regard to how the user queries the data, and the data analysis itself. For example, the specialized syntax that GTime Report requires inspired the automatic nature of the same functionality by tokenizing event titles for the Google Calendar analysis. Looking at analysis tools that aren't necessarily helpful to software engineers performing a self evaluation, like Gmail Meter, helped inspire viewing the data analysis from a new perspective. The complexities to run analyses like GCLAME and Git Inspector inspired UNBIASED's simple design.

Chapter 7

CONCLUSIONS

In this section we discuss the results of our hypotheses introduced in Chapter 1.

- HYPOTHESIS-1: Can we solve the challenges of the self evaluation process for software engineers?
- HYPOTHESIS-2: Can we automatically generate a self evaluation for a software engineer?

With regards to HYPOTHESIS-1, we have crafted solutions for the challenging aspects of the self evaluation process of Software Engineers. UNBIASED aims to solve the challenge of self evaluations being time consuming by automating parts of the self evaluation process: acquiring data, analyzing the data, and generating a result. Based on our results, we found that 30% of the Testing Group reported spending less time when using UNBIASED.

UNBIASED aims to solve the challenge of individuals forgetting about important contributions by showing the users all the information gleaned from third party APIs, and bringing special emphasis to high-impact / important information. UNBIASED aims to solve the challenge of self evaluations being biased and lacking credibility implicitly because all of the data extracted has a source. For both of these challenges, our results show that 90% of the Testing Group rated the accuracy of their self evaluation as 3 or higher (on a scale of 1 being *Not very accurate* and 5 being *Very accurate*. More detailed information can be found in Section 3.1.

With regards to HYPOTHESIS-2, we have developed a way to automatically generate a self evaluation for a software engineer. The generation includes extracting

accomplishments and accomplishment details, as well as figuring out strengths and areas of improvement by comparing users on a project-based level. We showed that it is possible, and some students (5 and 73) found that it was helpful. Because more than 50% of the students responded that it is unlikely they would want to use UNBIASED in the future, there is still more work that needs to be done. More detailed information can be found in Section 3.4.

7.1 Discussion

The final result of this thesis is an application that tackles the challenges of self evaluations for software engineers. UNBIASED currently supports the analysis of emails, calendars, software repositories and software management tools through the following third party APIs: Bitbucket, GitHub, Gmail, Google Calendar, and JIRA. Our results show that UNBIASED increases the accuracy of self evaluations, decreases the difficulty of performing a self evaluation, increases the number of sources used when performing a self evaluation, and in some cases UNBIASED decreases the amount of time needed to perform a self evaluation. To ensure that people will like and use UNBIASED in the future, we can use the feedback given by the students to make improvements to the system.

Chapter 8

FUTURE WORK

For future work, there are plans to commercialize an entire system that builds upon the UNBIASED application and incorporates additional peer and supervisory evaluations. Implementation-wise, there is room for integration with more third party services and APIs to better support a wider array of software engineers.

This tool is only as useful as the data it can analyze. Additional research into making instructions or defining standards can be made to aid software engineers in providing meaningful issue descriptions, worklog comments, commit messages, etcetera to make it easier for UNBIASED to analyze. In a recent study of over 23K+ Java projects it has been found that only 10% of the messages are descriptive and over 66% of those messages contained fewer than 20 words [23]. There is research in automatically generating Git commit messages [25] which could prove useful in providing more descriptive commits for our system to analyze.

There is also room for further research in the area of natural language processing. One idea is to create and train a specialized tokenizer and tagger specific to the different domains of emails, calendar events, commit messages, descriptions of work, and etcetera. This would be extremely helpful in the data extraction and analysis phase of the UNBIASED application.

BIBLIOGRAPHY

[1] Advanced Search . https://support.google.com/mail/answer/7190?hl=en.

[2] Awesome Graphs for Bitbucket . https://blog.bitbucket.org/2015/08/05/awesome-graphs-for-bitbucket-visualized-statistics-for-git-and-mercurial-repositories/.

[3] Bitbucket — The Git solution for professional teams . https://bitbucket.org/.

[4] Get up to speed with Pulse . https://github.com/blog/1476-get-up-to-speed-with-pulse.

[5] Git Commit Log Analysis Made Easy . https://eazybi.com/integrations/git.

[6] GitHub Features . https://github.com/features.

[7] GitInspector . https://github.com/ejwa/gitinspector.

[8] Gmail . https://www.google.com/intl/en/mail/help/about.html.

[9] Gmail Meter . https://gmail.googleblog.com/2012/04/know-your-gmail-stats-using-gmail-meter.html.

[10] Google App Engine . https://cloud.google.com/appengine/docs.

[11] Google Calendar . https://apps.google.com/products/calendar/.

[12] Google Cloud Datastore Documentation . https://cloud.google.com/datastore/docs/.

[13] Google Material Design . https://www.google.com/design/spec/material-design/introduction.html.

[14] GTime Report . https://www.gtimereport.com/.

[15] JIRA Software - Issue & Project Management . https://www.atlassian.com/software/jira.

[16] Natural Language Toolkit . http://www.nltk.org/.

[17] NLTK for App Engine . https://github.com/rutherford/nltk-gae.

[18] OAuth 2.0 . http://oauth.net/2/.

[19] Poylmer - 1.0 . https://www.polymer-project.org/1.0/.

[20] Python NDB Client Library Overview . https://cloud.google.com/appengine/docs/python/ndb/.

[21] Welcome to Jinja2 . http://jinja.pocoo.org/docs/dev/.

[22] N. Clark. Evaluating student teams developing unique industry projects. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*, ACE '05, pages 21–30, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[23] L. F. Cortés-Coy, M. Linares-Vásquez, J. Aponte, and D. Poshyvanyk. On automatically generating commit messages via summarization of source code changes. In *Proceedings of the 2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*, SCAM '14, pages 275–284, Washington, DC, USA, 2014. IEEE Computer Society.

[24] N. Herbert. Quantitative peer assessment: Can students be objective? In *Proceedings of the Ninth Australasian Conference on Computing Education -*

Volume 66, ACE '07, pages 63–71, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[25] W. Maalej and H. J. Happel. Can development work describe itself? In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pages 191–200, May 2010.

APPENDICES

Appendix A

PRE-EVALUATION SURVEY

(see following pages)

Pre-Evaluation Survey

This survey is to be completed before completing the 2016 Spring Self/Peer Evaluation. It will have no effect on your grade, but will be useful as we evaluate the self/peer evaluation process used in Capstone.

1. **What is your name?**

.....

2. **What team are you on?**
   Mark only one oval.

   - ◯ Android CRU
   - ◯ iCrew
   - ◯ JHM
   - ◯ Mystery Crew
   - ◯ WCFF
   - ◯ Scrubs
   - ◯ No Scrubs
   - ◯ Specific Atomics
   - ◯ Radar
   - ◯ Rocket

3. **How much time do you think you spent on your self/peer evaluation in Winter 2016?**
   Mark only one oval.

   - ◯ less than 10 minutes
   - ◯ 10-20 minutes
   - ◯ 20-30 minutes
   - ◯ 30-45 minutes
   - ◯ 45-60 minutes
   - ◯ over one hour

4. **Which of the following did you search/review while completing your self/peer evaluation in Winter 2016?**
Check all that apply.

- [ ] JIRA
- [ ] Bitbucket or Github
- [ ] Gmail
- [ ] Google Calendar
- [ ] Cal Poly Email
- [ ] Slack
- [ ] Facebook
- [ ] Other: .....

5. **How difficult was it to write your self evaluation in Winter 2016?**
Mark only one oval.

	1	2	3	4	5	
Easy	◯	◯	◯	◯	◯	Difficult

6. **How accurate (i.e. evaluation matched actual accomplishments) do you think was your self evaluation in Winter 2016?**
Mark only one oval.

	1	2	3	4	5	
Not very accurate	◯	◯	◯	◯	◯	Very accurate

Appendix B

POST-EVALUATION SURVEY FOR CONTROL GROUP

(see following pages)

Post-Evaluation Survey GA

This survey is to be completed after completing the 2016 Spring Self/Peer Evaluation. It will have no effect on your grade, but will be useful as we evaluate the self/peer evaluation process used in Capstone.

What is your name?

Your answer

What team are you on?

Choose

How much time do you think you spent on your self/peer evaluation in Spring 2016 (406)?

- ○ less than 10 minutes
- ○ 10-20 minutes
- ○ 20-30 minutes
- ○ 30-45 minutes
- ○ 45-60 minutes
- ○ over one hour

Which of the following did you search/review while completing your self/peer evaluation in Spring 2016 (406)?

- ☐

- [ ] JIRA
- [ ] Bitbucket or Github
- [ ] Gmail
- [ ] Google Calendar
- [ ] Cal Poly Email
- [ ] Slack
- [ ] Facebook
- [ ] Other:

How difficult was it to write your self evaluation in Spring 2016 (406)?

	1	2	3	4	5	
Easy	○	○	○	○	○	Difficult

How accurate (i.e. evaluation matched actual accomplishments) do you think was your self evaluation in Spring 2016 (406)?

	1	2	3	4	5	
Not very accurate	○	○	○	○	○	Very accurate

SUBMIT

Never submit passwords through Google Forms.

Appendix C

POST-EVALUATION SURVEY FOR TESTING GROUP

(see following pages)

REQUEST EDIT ACCESS

Post-Evaluation Survey CRU

This survey is to be completed after completing the 2016 Spring Self/Peer Evaluation. It will have no effect on your grade, but will be useful as we evaluate the self/peer evaluation process used in Capstone.

What is your name?

Your answer

What team are you on?

Choose

How much time do you think you spent on your self/peer evaluation in Spring 2016 (406)?

- ○ less than 10 minutes
- ○ 10-20 minutes
- ○ 20-30 minutes
- ○ 30-45 minutes
- ○ 45-60 minutes
- ○ over one hour

Which of the following did you search/review within the

UNBIASED Self Evaluation System while completing your

self/peer evaluation in Spring 2016 (406)?

- [ ] JIRA
- [ ] Bitbucket or Github
- [ ] Gmail
- [ ] Google Calendar
- [ ] Other:

Outside of using the UNBIASED Self Evaluation System, which of the following did you search/review while completing your self/peer evaluation in Spring 2016 (406)?

- [ ] JIRA
- [ ] Bitbucket or Github
- [ ] Gmail
- [ ] Google Calendar
- [ ] Cal Poly Email
- [ ] Slack
- [ ] Facebook
- [ ] Other:

How difficult was it to write your self evaluation in Spring 2016 (406)?

	1	2	3	4	5	
Easy	○	○	○	○	○	Difficult

How accurate (i.e. evaluation matched actual

accomplishments) do you think was your self evaluation in Spring 2016 (406)?

	1	2	3	4	5	
Not very accurate	○	○	○	○	○	Very accurate

Do you think you wrote a better, same, or worse self evaluation because you used the UNBIASED Self Evaluation System in completing your self evaluation in Spring 2016 (406)?

- ○ This eval was better than the past
- ○ This eval was about the same quality as the past
- ○ This eval was worse than the past

If you had to complete another self evaluation in the future, how likely would you want to use the UNBIASED Self Evaluation System again?

	1	2	3	4	5	
Not likely	○	○	○	○	○	Very likely

Tell us what you think about the UNBIASED Self Evaluation System. What did you like? What did you not like?

Your answer

SUBMIT

Never submit passwords through Google Forms.

Appendix D

SURVEY RESULTS

(see following pages)

All Pre-Eval Survey

ID	How much time do you think you spent on your self/peer evaluation in Winter 2016?	Which of the following did you search/review while completing your self/peer evaluation in Winter 2016?	How difficult was it to write your self evaluation in Winter 2016?	How accurate do you think was your self evaluation in Winter 2016?
1	30-45 minutes		4	5
2	20-30 minutes	Slack	2	4
3	20-30 minutes	Bitbucket or Github, Slack	3	4
4	10-20 minutes	JIRA, Bitbucket or Github, Gmail	1	5
5	10-20 minutes	I didn't need to check anything	1	4
6	20-30 minutes		3	4
7	10-20 minutes	JIRA, Bitbucket or Github, Slack	1	5
8	over one hour	Slack, I know what everyone's been doing.	2	3
9	10-20 minutes	JIRA	3	4
10	20-30 minutes	Bitbucket or Github, Slack	2	4
11	30-45 minutes	JIRA, Slack	5	4
12	45-60 minutes	JIRA, Bitbucket or Github	4	5
13	over one hour	JIRA, Bitbucket or Github, Slack	2	5
15	20-30 minutes	None	2	5
16	30-45 minutes		3	3
17	10-20 minutes		4	3
18	10-20 minutes	Bitbucket or Github, Slack	4	4
19	20-30 minutes		3	3
20	10-20 minutes	JIRA, Bitbucket or Github	1	4
21	20-30 minutes	JIRA, Bitbucket or Github	2	3
22	10-20 minutes	Bitbucket or Github	2	4
23	10-20 minutes		1	4
24	20-30 minutes		2	4
25	10-20 minutes	JIRA, Bitbucket or Github, Slack	2	4
26	10-20 minutes	JIRA, Bitbucket or Github, Slack	2	4
27	10-20 minutes	JIRA, Bitbucket or Github	2	4

All Pre-Eval Survey

ID	How much time do you think you spent on your self/peer evaluation in Winter 2016?	Which of the following did you search/review while completing your self/peer evaluation in Winter 2016?	How difficult was it to write your self evaluation in Winter 2016?	How accurate do you think was your self evaluation in Winter 2016?
28	10-20 minutes	Group Me	1	5
50	10-20 minutes	JIRA, Bitbucket or Github	3	4
51	10-20 minutes	JIRA, Slack	1	5
52	10-20 minutes	JIRA, Bitbucket or Github	4	3
53	20-30 minutes	JIRA, Bitbucket or Github	3	3
54	20-30 minutes		3	3
55	10-20 minutes	JIRA, Bitbucket or Github	1	4
56	30-45 minutes	JIRA, Bitbucket or Github, GroupMe	4	5
57	45-60 minutes	JIRA, Bitbucket or Github, Google Calendar, Slack	3	4
58	20-30 minutes	JIRA, Slack	4	4
59	10-20 minutes	JIRA, Bitbucket or Github	4	3
60	10-20 minutes		2	5
61	10-20 minutes	JIRA	3	5
62	30-45 minutes	JIRA, Bitbucket or Github, Slack	3	5
63	10-20 minutes		1	4
64	20-30 minutes	JIRA, Bitbucket or Github	4	3
65	10-20 minutes		4	1
66	10-20 minutes	JIRA, Bitbucket or Github, Slack	4	3
67	20-30 minutes	JIRA, Bitbucket or Github, GroupMe	3	3
68	20-30 minutes	JIRA, Bitbucket or Github	3	4
69	10-20 minutes		2	2
70	10-20 minutes	JIRA, Bitbucket or Github, Slack	3	4
71	30-45 minutes	JIRA, Bitbucket or Github, Slack	4	4
72	10-20 minutes	JIRA, Bitbucket or Github	1	4
73	20-30 minutes	JIRA, Bitbucket or Github, Slack	2	5
74	20-30 minutes	JIRA	3	3

All Pre-Eval Survey

ID	How much time do you think you spent on your self/peer evaluation in Winter 2016?	Which of the following did you search/review while completing your self/peer evaluation in Winter 2016?	How difficult was it to write your self evaluation in Winter 2016?	How accurate do you think was your self evaluation in Winter 2016?
75	45-60 minutes	JIRA, Bitbucket or Github, Slack	4	5
76	10-20 minutes	JIRA, Slack	2	4
77	10-20 minutes	JIRA, Bitbucket or Github	3	4

Control Group Post-Eval Survey

ID	How much time do you think you spent on your self/peer evaluation in Spring 2016 (406)?	Which of the following did you search/review while completing your self/peer evaluation in Spring 2016 (406)?	How difficult was it to write your self evaluation in Spring 2016 (406)?	How accurate do you think was your self evaluation in Spring 2016 (406)?
1	20-30 minutes		5	4
2	10-20 minutes	JIRA, Slack	1	4
3	20-30 minutes	JIRA	2	4
4	10-20 minutes	JIRA, Bitbucket or Github, Gmail, Slack	1	3
6	30-45 minutes		4	4
7	20-30 minutes		1	5
8	over one hour	Nothing	2	3
9	10-20 minutes	JIRA	1	4
10	45-60 minutes	JIRA	5	4
11	20-30 minutes	JIRA, Bitbucket or Github	4	4
13	45-60 minutes	JIRA, Bitbucket or Github, Slack	3	5
14	45-60 minutes	JIRA, Bitbucket or Github, Gmail, Google Calendar, Slack	2	4
18	30-45 minutes	JIRA, Bitbucket or Github, Slack	4	3
19	10-20 minutes		1	5
20	20-30 minutes	Status Updates	2	5
21	30-45 minutes	JIRA, Bitbucket or Github, Gmail	3	4
22	10-20 minutes	Slack	2	4
23	10-20 minutes	JIRA	2	4
24	10-20 minutes		2	4
25	10-20 minutes	JIRA, Bitbucket or Github, Slack	4	2
26	10-20 minutes	JIRA	4	3
27	10-20 minutes	JIRA, Bitbucket or Github	2	4

Testing Group Post-Eval Survey

ID	How much time do you think you spent on your self/peer evaluation in Spring 2016 (406)?	Which of the following did you search/review within the UNBIASED Self Evaluation System while completing your self/peer evaluation in Spring 2016 (406)?	Outside of using the UNBIASED Self Evaluation System, which of the following did you search/review while completing your self/peer evaluation in Spring 2016 (406)?	How difficult was it to write your self evaluation in Spring 2016 (406)?	How accurate do you think was your self evaluation in Spring 2016 (406)?	Do you think you wrote a better, same, or worse self evaluation because you used the UNBIASED Self Evaluation System in completing your self evaluation in Spring 2016 (406)?	If you had to complete another self evaluation in the future, how likely would you want to use the UNBIASED Self Evaluation System again?
50	20-30 minutes	JIRA, Bitbucket or Github	JIRA, Bitbucket or Github, Slack	3	4	This eval was better than the past	4
51	10-20 minutes	JIRA, Bitbucket or Github		3	3	This eval was about the same quality as the past	1
52	10-20 minutes	JIRA, Bitbucket or Github		3	3	This eval was about the same quality as the past	1
54	30-45 minutes			4	3	This eval was about the same quality as the past	1
56	10-20 minutes	JIRA, Bitbucket or Github	JIRA, Bitbucket or Github, Groupme	1	2	This eval was about the same quality as the past	4
58	less than 10 minutes	JIRA, Bitbucket or Github		2	4	This eval was better than the past	5
60	10-20 minutes	JIRA, Bitbucket or Github	JIRA	3	1	This eval was about the same quality as the past	1
62	30-45 minutes	JIRA, Bitbucket or Github	JIRA, Bitbucket or Github	3	5	This eval was about the same quality as the past	2

Testing Group Post-Eval Survey

ID	How much time do you think you spent on your self/peer evaluation in Spring 2016 (406)?	Which of the following did you search/review within the UNBIASED Self Evaluation System while completing your self/peer evaluation in Spring 2016 (406)?	Outside of using the UNBIASED Self Evaluation System, which of the following did you search/review while completing your self/peer evaluation in Spring 2016 (406)?	How difficult was it to write your self evaluation in Spring 2016 (406)?	How accurate do you think was your self evaluation in Spring 2016 (406)?	Do you think you wrote a better, same, or worse self evaluation because you used the UNBIASED Self Evaluation System in completing your self evaluation in Spring 2016 (406)?	If you had to complete another self evaluation in the future, how likely would you want to use the UNBIASED Self Evaluation System again?
63	20-30 minutes	Bitbucket or Github		3	5	This eval was about the same quality as the past	1
65	10-20 minutes	Bitbucket or Github	Bitbucket or Github	4	3	This eval was about the same quality as the past	1
66	30-45 minutes	JIRA, Bitbucket or Github	JIRA, Bitbucket or Github	4	4	This eval was better than the past	1
67	20-30 minutes	JIRA, Bitbucket or Github, GroupMe	JIRA, Bitbucket or Github	3	3	This eval was about the same quality as the past	3
68	30-45 minutes	JIRA, Bitbucket or Github	JIRA, Bitbucket or Github, Slack	2	4	This eval was about the same quality as the past	5
69	10-20 minutes	JIRA, Bitbucket or Github	JIRA, Bitbucket or Github	1	4	This eval was about the same quality as the past	3
71	20-30 minutes	JIRA, Bitbucket or Github	JIRA, Bitbucket or Github	1	4	This eval was about the same quality as the past	4

Testing Group Post-Eval Survey

ID	How much time do you think you spent on your self/peer evaluation in Spring 2016 (406)?	Which of the following did you search/review within the UNBIASED Self Evaluation System while completing your self/peer evaluation in Spring 2016 (406)?	Outside of using the UNBIASED Self Evaluation System, which of the following did you search/review while completing your self/peer evaluation in Spring 2016 (406)?	How difficult was it to write your self evaluation in Spring 2016 (406)?	How accurate do you think was your self evaluation in Spring 2016 (406)?	Do you think you wrote a better, same, or worse self evaluation because you used the UNBIASED Self Evaluation System in completing your self evaluation in Spring 2016 (406)?	If you had to complete another self evaluation in the future, how likely would you want to use the UNBIASED Self Evaluation System again?
72	10-20 minutes			1	4	This eval was about the same quality as the past	2
73	20-30 minutes	JIRA, Bitbucket or Github	JIRA, Slack	1	5	This eval was about the same quality as the past	4
74	less than 10 minutes	JIRA		3	4	This eval was about the same quality as the past	3
76	10-20 minutes	JIRA, Bitbucket or Github	JIRA	3	4	This eval was about the same quality as the past	2
77	20-30 minutes	JIRA, Bitbucket or Github	JIRA, Bitbucket or Github	4	3	This eval was about the same quality as the past	2

Testing Group Post-Eval Survey

Tell us what you think about the UNBIASED Self Evaluation System. What did you like? What did you not like?
I like the idea of it but it was a little confusing and took more time to finish the review.
Doesn't connect well with Jira or Bitbucket, added no meaningful content, requires permissions to my accounts that make me uncomfortable. Would not use again.
I'm not 100% sure that hooking it up to bitbucket worked. Then I lost all my progress when I tried to hook it up with gmail and got some weird error. I'm not even completely sure that my eval got submitted.
I like the simplicity it has in compiling individual contributions that were up on the remote repository. My biggest qualm is that it does not and cannot, from what I know, see the local development that may not have been completed or sent up. All in all I really like it though due to the simplification.
I really liked the convenience of it and how it made my self evaluation objective rather than subjective. I think the only reason why it may lack accuracy is because I did not document all my accomplishments throughout the quarter since I did not know I'd be using this. I would definitely use this in the future, but I would just make sure to optimize JIRA and github.
I didn't really like it just because it only entered one line for me and it wasn't exactly anything relevant.
I would rather enter information in the fields manually
It was unstable. GitHub looked at which PRs we merged instead of which ones we created.
Google account log in gives 404 error. JIRA doesn't work at all. Seems like GitHub was the only part that worked for me.
I like the concept, although systems like JIRA and Bitbucket already provide ways for me to isolate my contributions in a way that allows me to easily remember what I've done this quarter. However, centralizing those sources could potentially be useful. Neither tool worked for me, UNBIASED could only find two of my commits and could not connect to JIRA, so my experience was the same as in previous quarters. It is a good effort, however, and I appreciate the idea of streamlining the process and how that may contribute to more accurate analyses.

Testing Group Post-Eval Survey

Tell us what you think about the UNBIASED Self Evaluation System. What did you like? What did you not like?
I liked it but for GitHub it just kept loading and wouldn't end. And I wasn't sure when the form was complete and I could leave from the page. Other than that it was pretty nice.
The auto generation was really nice and was pretty accurate in displaying contributions. However, there are some intricacies that aren't addressed such as if two people pair program a task in Jira, but only one gets to complete it so it doesn't appear on the other's tasks (same with Git commits)
I like the idea of the system. In my case the generated self evaluation showed PRs I had merged, not created, so it was exclusively the work of other people. I had to go to github to see which PRs I had actually created.
It was cool how it aggregated the data for me. I did not like that it had some problems with JIRA / JIRA not working in general. However, I believe tools like this will be more prevalent in the coming years especially with so much data being passed around. :0
It just doesn't feel necessary. I have a pretty good idea of how my team does without needing to look at Jira or not. The tool seems fine if it was used to evaluate people's grades for someone not on the group (ie Dr Janzen) but from within it just is an annoyance to took it up with Jira, Bitbucket, etc.
I like it, I think it is a helpful resource that can be used a great starting point in writing self reflections.
It was too buggy. It's also nice to get a recap of what I've done, but I mostly remember what I've done for the past two months.
I don't like that I have to give it access to my entire bitbucket account. Also, the bitbucket search didn't work for me. I tried adding a username alias, but no luck. Also, there doesn't appear to be any way to un-link my account. For jira there are several issues that I forgot to log hours on, but still resolved- you should display "issues resolved" by a particular user.