

**PROBEXPERT: AN ENHANCED Q&A
PLATFORM FOR REDUCING TIME SPENT ON
LEARNING AND FINDING ANSWERS
2021-155**

Thennakoon T.M.K.H.B

IT18004564

Bachelor of Science Special (Honors) in Information Technology
Specializing in Software Engineering

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

**PROBEXPERT: AN ENHANCED Q&A
PLATFORM FOR REDUCING TIME SPENT ON
LEARNING AND FINDING ANSWERS
2021-155**

Thennakoon T.M.K.H.B

IT18004564

Bachelor of Science Special (Honors) in Information Technology
Specializing in Software Engineering

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date:

The above candidate has carried out research for the bachelor's degree Dissertation under my supervision.

Signature of the supervisor:

Date:

ABSTRACT

With the rise of developer social networking sites like Stack overflow, LinkedIn and GitHub, a huge amount of important information about developers' activities are generated every second on the internet. With the rapid increase of these data, plenty of research have been performed to detect experts in the questions answer communities and social coding platforms. But still, these studies focused on assess developers' skills based on a single platform. Therefore, this study will evaluate and assign overall expertise scores to developers considering their contributions to the developer community. Therefore, developers will be evaluated by considering both the qualitative and quantitative factors in two major categories, which are Q&A skills and coding skills. Stack Overflow and GitHub will be used to gather out the necessary data, respectively. Then, a scoring model was developed to analyse user activities and behaviours on the above-mentioned platforms to calculate a score according to developers' proficiency. Then by normalizing the generated score using normalCDF, users are classified into the classes accordingly. Since most hiring managers consider GitHub, Stack Overflow profiles are more reliable than a candidate's custom-made resume. The portfolio system of the ProbExpert will offer developers a fully fledged auto-generated portfolio by filtering the necessary information of the above-mentioned gathered dataset. This portfolio will contain references to the user's research papers, blog articles, git contributes, academic qualifications, employment history, etc. Finally, the Q&A, and coding scores will be displayed next to the developer's profile picture using the results generated by the above-mentioned ranking algorithm.

.

Keywords: bell curve, normal CDF, portfolio generation, user ranking

Acknowledgement

I would like to take the procession of this section to express my sincere gratitude to all the individuals who guided me through this journey since day one. First of all, I would like to thank the Sri Lanka Institute of Information Technology (SLIIT) for according this opportunity to release innovative ideas through this project as a compulsory requirement of the course. Also, I take this opportunity to thank each faculty member and lecturer who lent their hand in guidance and support throughout this research project.

It was an honour to have a supervisor who assisted us to get back in the right direction when we chose the wrong path. So, I would like to express my heartiest gratitude to Ms. Dinuka Wijendra, who willingly agreed to supervise this research through the year and provided advice to enhance the worth of end result. I would like to thank our co-supervisor Ms. Anjalie Gamage, who willing to supervise this project throughout the year.

Finally, my gratitude would be paid to the colleagues of my team, my friends and my family members who encourage and support to strengthen.

Last but not least, I would like to thank all others whose names are not listed here, but have given their utmost encouragement and support in every possible manner

TABLE OF CONTENTS

1	INTRODUCTION.....	1
1.1	Background Literature.....	1
1.2	Research Gap.....	5
1.3	Research Problem.....	7
1.4	Research Objectives.....	12
2	METHODOLOGY	15
2.1	System Overview	15
2.2	Data Gathering.....	17
2.2.1	Git-Data-Miner	17
2.2.2	Dataset.....	17
2.2.3	Data visualization and feature extraction.....	20
2.3	GitHub Scorer Model.....	30
2.3.1	User Classification based on Bell Curve.....	31
2.3.2	StackOverflow and LinkedIn Scorer Models.....	35
2.4	Automated Portfolio Generation	35
2.5	Commercialization aspects of the product.....	39
2.5.1	Premium Accounts for Hiring Managers	39
2.5.2	CV generate option as a premium service	39
2.5.3	Section for display Open job Opportunities	39
2.5.4	Display Sponsored Advertisements.....	40
3	TESTING & IMPLEMENTATION	41

3.1	Testing.....	41
3.2	Implementation.....	43
3.2.1	Portfolio Front-end	43
3.2.2	Portfolio Back-end	43
3.2.3	Web Scrappers	44
3.2.4	GitHub OAuth2 flow	45
3.2.5	CV generation.....	45
3.2.6	Deployment	46
4	RESULTS & DISCUSSION	48
4.1	Results.....	48
4.2	Research Findings.....	49
4.3	Discussion.....	50
4.4	Summary of the Student Contribution.....	51
7	CONCLUSION.....	52

LIST OF FIGURES

Figure 1.1 – Summary of responses for maintaining a portfolio.	8
Figure 1.2 – Summary of responses for the importance of having a portfolio.	8
Figure 1.3 - Summary of responses for reasons to not maintain a portfolio.....	9
Figure 1.4 - Summary of responses for update frequency to the portfolio.....	9
Figure 1.5 – Summary of likeness to find and follow the experts related to the career path.....	10
Figure 2.1: High-level architecture diagram.....	16
Figure 2.2: ECDF distribution of total followers count.....	21
Figure 2.3: Boxplot graph of the usage of programming languages.....	22
Figure 2.4: GitHub profile age distribution	23
Figure 2.5: Availability for hire.....	24
Figure 2.6: Contributed Repositories (GitHub Stars \geq 10,000)	25
Figure 2.7: Contributed Repositories (10,000>GitHub Stars \geq 5,000).....	25
Figure 2.8: Contributed Repositories (5,000>GitHub Stars \geq 500).....	26
Figure 2.9: Total Pull Request Count.....	27
Figure 2.10: ECDF graph of the discussion data	27
Figure 2.11: ECDF graph of the approved comments as answers.....	28
Figure 2.12: Distribution of the received stars.....	29
Figure 2.13: Box plots of the created issues	29
Figure 2.14: High level diagram of the GitHub Scorer	30
Figure 2.15: Classes distribution on a bell curve	31
Figure 2.16: Bell curve - test case 01	33
Figure 2.17: Bell curve - test case 02.....	34
Figure 2.18: Bell curve - test case 03.....	34

Figure 2.19: Use-Case Diagram of the Portfolio System.....	36
Figure 2.20: Portfolio UI (WIP)	37
Figure 2.21: ProbExpert Leaderboard.....	38
Figure 2.22: ProbExpert Marketing Strategy	40
Figure 3.1: Jest code coverage results.....	41
Figure 3.2: Testing API endpoints via Postman.....	42
Figure 3.3: Sample user document on MongoDB.....	44
Figure 3.4: OAuth implementation	45
Figure 3.5: Frontend deployment on Vercel.....	46
Figure 3.6: Backend deployment on Heroku	47

LIST OF TABLES

Table 1.1: Prevailing algorithms' appropriateness to assess developers' skills.....	6
Table 1.2: User profile details of the leading developers' platforms.....	7
Table 2.1: Distribution of the Classes	32
Table 2.2: User info - test case 01.....	32
Table 2.3: User info - test case 02.....	33
Table 2.4: User info - test case 03.....	34
Table 4.1: Summary of student contribution	51

LIST OF EQUATIONS

2.1	31
-----------	----

LIST OF ABBREVIATIONS

API	:Application Program Interface
Q&A	:question and answer
URL	:Uniform Resource Locator
CV	Curriculum Vitae
PDF	Portable Document Format
CDF	Cumulative Distribution Function
SSR	Server Side Rendering
WIP	work-in-progress

1 INTRODUCTION

1.1 Background Literature

Modern software development heavily depends on social coding platforms such as GitHub and GitLab to increase productivity and reduce time-to-market[1]. As a result of this, the platforms mentioned above are frequently used by developers to demonstrate their expertise and establish an online portfolio of their development activities[2]. Nevertheless, these platforms are not developed for this scenario; Active programmers use the limited social components of these platforms to promote themselves by showcasing their exciting projects and contributions[2]. Since the information of these platforms could not be altered, and they contain all the actual development statistics of a developer, profiles created from such social coding platform accounts are seen as more dependable than Resumes in terms of assessing a candidate's technical abilities[3]. Therefore, potential employers and recruiters often consider reviewing their candidates' profiles in the recruitment process.

Apart from social coding platforms such as GitHub, the other most significant platform developers spend most of their time finding answers for their tempting questions is StackOverflow. StackOverflow is the dominant Q&A system on the internet for programmers[4]. Stack Overflow has over 14 million registered users and has received over 21 million questions and 31 million answers as of March 2021. As a result, developers who made successful significant contributions via StackOverflow get recognised by the community as the experts of their subsequent fields[5]. Furthermore, these contributions lead them to have more excellent job opportunities in their fields.

However, the above scenario is only valid for the top 0.001 contributors, whom StackOverflow itself nominates[5]. Furthermore, it does not provide a feature for identifying other developers apart from its built-in reputation system. Since this reputation system is solely based on the upvote and downvote mechanism, A beginner

user could get lucky by asking a question related to the latest technology, which hasn't been asked on the platform before and get plenty of upvotes[4]. Along with that scenario, there are few other scenarios that are thoroughly explained in the Sparrows and owls[4]. As a result, recruiters have to spend a major quota of their time going through the candidates' StackOverflow accounts to find out whether their contributions are significant or not[6]. However, to determine a developer is really skillful enough, scanning a StackOverflow profile is not enough because it gives theoretical knowledge of the developer rather than practical coding and software development skills[7]. Therefore, recruiters also have to consider developers' actual coding behaviors via reviewing their activities on social coding platforms such as GitHub to assess a developer in full[7].

Since both of these above platforms do not provide any out of box feature for this need, plenty of research have been carried out to detect experts in question-answer communities and social coding platforms. As an extension of these researches and to give solutions to the above-discussed issue, this work focuses on creating a fully fledged auto-generated portfolio and a ranking system for developers based on considering their opensource contributions to GitHub and StackOverflow. It is developed and integrated as a feature into the ProbExpert Novel Q&A platform.

Finding experts on questions answering platforms has become a crucial task in recent years with the rapid increase of Q&A platforms[8]. The primary reason for this is individuals admit that experts have the knowledge to solve issues within considerably less time and present profound guidance[9] for their questions. Stack Overflow, GitHub issues[2], Quora play a significant role when this comes to the developers' online communities. Like all other social platforms, such community-driven Q&A platforms also suffer from the quality issue[8]. Leading developer's online community platforms like Stack overflow and other Q&A platforms follow community feedback[10] to evaluate the answers by using an upvote, downvote, edit, and flag mechanism[5]. Still, it may take a long time to receive these feedbacks from the community[4], and for less popular topics, it usually takes days. On the other hand, when the number of unsolved question overgrows, some questions will not receive the

attention they should receive[11]. Therefore most of the time, these questions will get answered by novice users[9], and the asker will not get the optimal answer for their problems[11]. Consequently, the asker and other users who follow the solution will waste a lot of time due to the novice answer's nature. Therefore it is essential to have a way to find out whether the user who answered the question is an expert or not in the related field[9]. Consequently, researchers have taken many approaches to overcome this issue.

Link analysis-based ranking algorithms are the earliest approach to this problem. These Link analysis ranking algorithms prove their strength in ranking web pages relevant for a particular search query[12]. HITS[13] and PageRank[14] algorithms are the most famous and widely used algorithm for this purpose. Hyperlink Induced Topic Search (HITS) could find and rank webpages relevant for a particular search. This algorithm's idea is that a website should link to other relevant sites and be linked by other important sites[12]. Therefore HITS uses hubs and authorities to define this recursive relationship between the webpages[15]. Researchers used this algorithm to estimate users' expert level by showing question answering relationships in a user graph. In this graph, each user is depicted as a node, and edges indicate the question-answering relationship between two users[8]. In 2007, Zang developed the ExpertiseRank model[16], which is based on the PageRank algorithm. His model considers the number of users one has helped and also whom they helped. In the same year, Jurczyk proposed a HITS algorithm[15] model to calculate user expertise score by evaluating askers as hub nodes and answerers as the authority nodes. However, all these models have failed to assess the given answers quality information like best, informative, and usefulness.

As a result of this, all most all the following researches mind to include the best, top voted answer tag when estimating the expertise score of users[8]. In 2009, Bian et al. took an approach to this problem by proposing a mutual reinforcement[17] method to evaluate user expertise score with the answer quality. With the rapid increase of the Q&A platforms, researchers found out that user behaviors also have significant weight when detecting experts. Patil et al. study[9] showed that expert prefers to answer the

questions which already have answered but haven't received the optimal answer. So they tend to answer this type of problem more. By doing that, they believe they can make more valuable contributions to the community rather than answering random questions. Based on this finding, Bougessa et al. model users' question selection bias to identify experts[18]. That model also considers the number of the best answer tags when filtering experts from the average users. In 2010, Wei-Chen et al. proposed a novel hybrid approach[19] that considers user subject relevance, user reputation, and authority of a category in finding experts. It is the first model up to that time which considers both users' reputation and the users' domain knowledge to the target questions. In 2014, Zhao proposed a topic-sensitive probabilistic model[8] for detect experts in the question-answer communities.

Since this study focused on finding experts in the programming field, evaluate their skills solely on Q&A platforms will not be enough. To improve the proposed user score estimation algorithm's success rate, its essentials to consider developers' statistics on social coding platforms such as GitHub and Gitlab. These platforms enable developers to efficiently work on projects, connect with other developers, volunteer for open source projects, and "be seen" by the community[3]. Consequently, employers and HR managers often scan candidates' GitHub profiles in the interview process to learn more about their skills and interests. In reference [6], Capiluppi et al. identified four main insight that employers can derive from developers' GitHub profiles by doing an interview-based study with several IT employers. Those are

- Shared open-source values
- Community acceptance of works and contribution quality
- Project management skills
- Passion for coding.

Joao Eduardo et al. propose an unsupervised machine learning approach to identify experts in software libraries and frameworks among GitHub users. Joao collected 13 features about developers' activities on GitHub in three popular JS libraries to build their unsupervised (based on clustering) model.

However, these previous researches do not provide an overall expertise score for a developer because they focus on the general type of platforms or are platform dependent. Although other professional social platforms like LinkedIn evaluate developer's skill levels in more intangible ways, such as endorsement from peers and the number of connects. They do not include information about a user's own hands-on experience; instead, they depend on the opinions of a user's contacts to build a picture of his or her knowledge.

1.2 Research Gap

Since plenty of research have been performed to detect experts in the questions answer communities and social coding platforms within the last two decades, most of these studies focused on evaluating developers on individual platforms. Furthermore, they are focused on just finding the experts. No studies focused on classifying users according to their skill level (e.g., beginner, intermediate and advanced). The Prevailing algorithms that developed to detect experts on Q&A platforms only focused on the general type of Q&A platforms such as Quora and Yahoo Answers, and No study focused on the platforms in the specialized field such as programming. Therefore, it isn't idle to use previous research outcomes directly to evaluate developers' proficiency in specialized platforms such as Stack Overflow [20], [2], and [21] are the most famous studies conducted to assess developers on social coding platforms like GitHub. As in the Q&A platform-based research, these are also solely focused only on GitHub.

Table 1.1: Prevailing algorithms' appropriateness to assess developers' skills

Algorithm	Suitability to assess developers
Page Rank	Not suitable (Assess users via link analysis)
HITS	Not suitable (Assess users via link analysis)
Mutual Reinforcement approach	Considers how many people are involved and who has helped whom.
Hybrid approach of Wen-Chen	Considers five types of relationships between users.
Topic sensitive probabilistic model	Considers the interests, expertise, and reputation of users. (extended version of Page Rank)
Purposed model	Assess, score and rank developers based on their opensource contribution statistics.

In this era, advanced knowledge-sharing social platforms have increasingly attracted people's attention due to their importance and impact on society. These platforms combine a broad range of knowledgeable data, including discussion forums (like Quora, Stack Overflow), blogging networks (Medium, DEV), code hosting platforms (GitHub, Gitlab, Bitbucket), and professional networking sites such as LinkedIn. With the unstoppable rise of popularity, these platforms have moved beyond personal use. They have been increasingly used by organisations when recruiting employees. Therefore, these platforms start to focus more on generating a detailed profile for their users. These profiles contain users' reputations, contributions, activities, etc. But the problem is that these profiles are platform dependent. Therefore, up to today, there are no methods users can use to showcase all of their works and contributions in a single place (single platform).

Table 1.2: User profile details of the leading developers' platforms

Platform	User-level	Publications	Open-Source contributions	Detailed Bio	Blogs
Stack Overflow	Based on the platform reputation points	None	None	Medium	None
LinkedIn	Based on endorsements	User has to post manually	None	Good	None
Medium	None	None	None	Low	Platform Dependent
GitHub	None	None	Available	Low	None
ProbExpert	Based on all the platforms activities and behaviors	Available	Available	Good	Platform Independent

1.3 Research Problem

In this technological era, the digital portfolio has become the primary method of showcasing individual work. Certain professions like designers, artists, and decorators used to showcase their works using a portfolio, but today, any professional in any career can and should consider starting a portfolio. According to the survey conducted with undergraduate university students, Figure 1.1 shows 68% of undergraduates do not maintain a portfolio to showcase their work. Yet, according to Figure 1.2, more than 90% of them see maintaining a portfolio as an important thing to do.

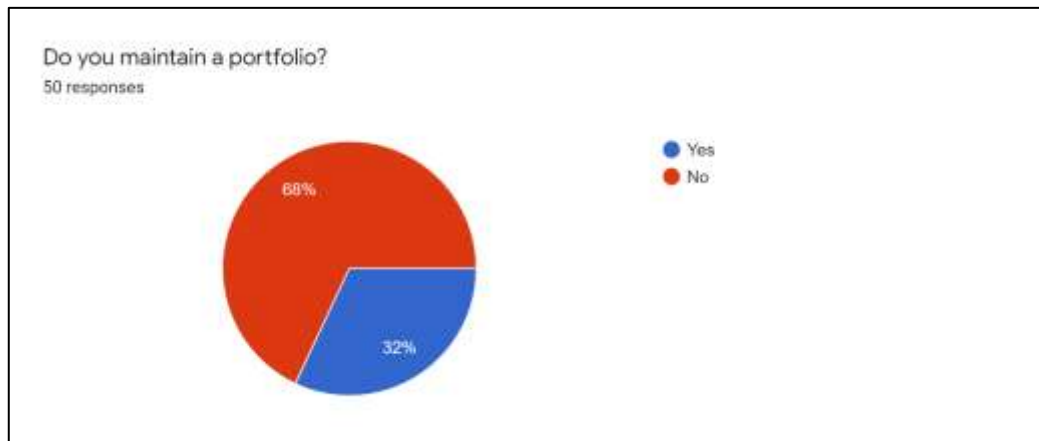


Figure 1.1 – Summary of responses for maintaining a portfolio.

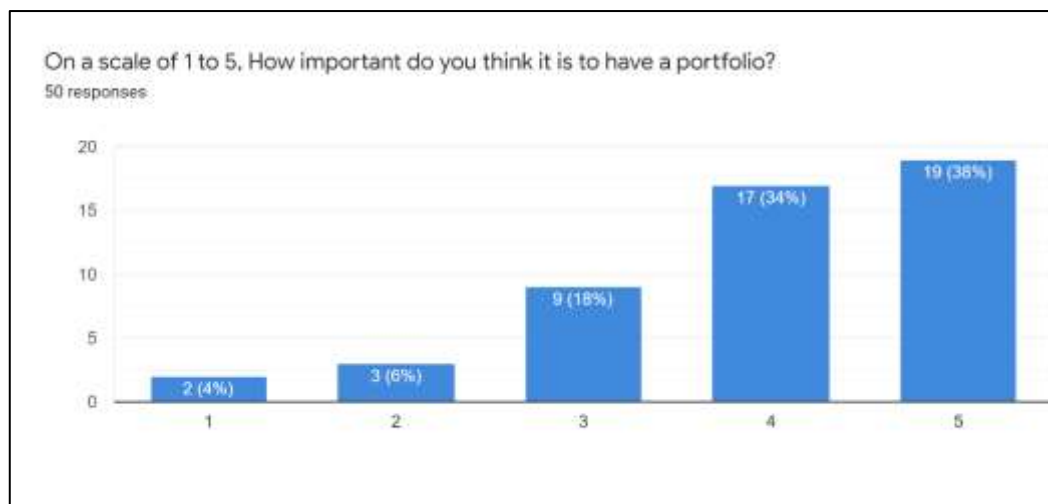


Figure 1.2 – Summary of responses for the importance of having a portfolio.

Since most hiring managers examine their candidates' portfolios while in the interview phase, developers also tend to maintain portfolios themselves. Developing a good-looking portfolio is a time-consuming task and requires a lot of skills such as Web development, UI/UX, SEO, etc. Figure 1.3 clearly shows most of developers suffer from these issues.

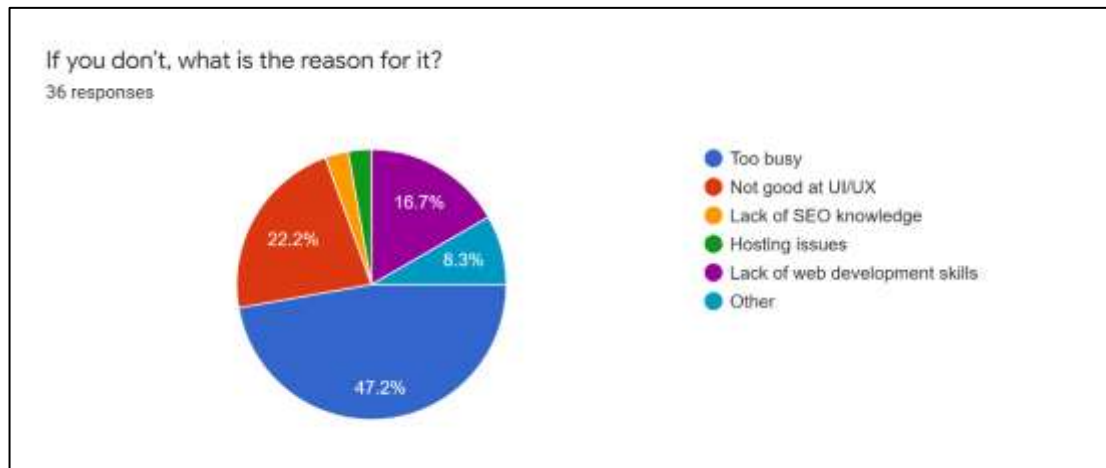


Figure 1.3 - Summary of responses for reasons to not maintain a portfolio.

Building a one without following these practices will depict developers' talent to the recruiters negatively. Therefore, most of the developers ignore making portfolios for themselves. Figure 1.4 shows, developers who somehow create a one also suffers from maintaining them in a timely manner.

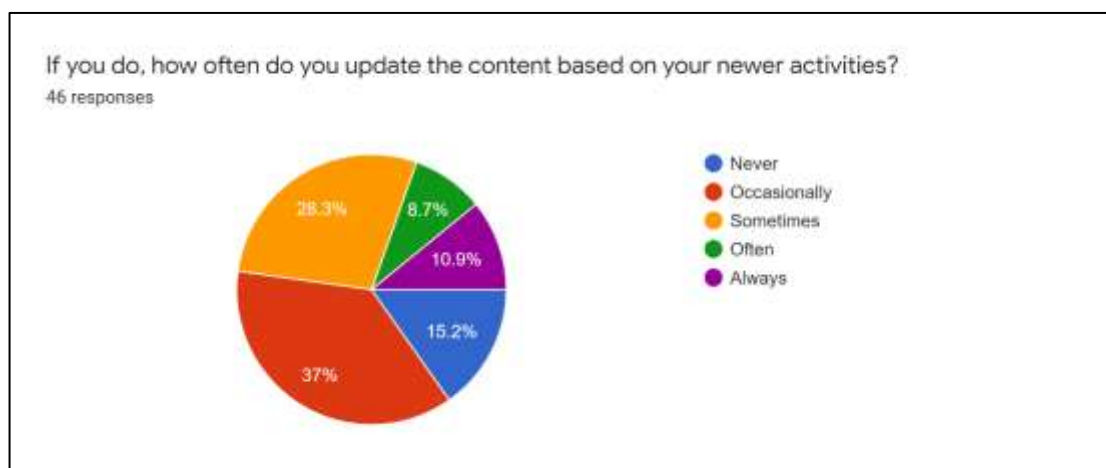


Figure 1.4 - Summary of responses for update frequency to the portfolio

The novel platform, ProbExpert, which is purposed, will offer developers a fully fledged auto-generated portfolio out of the box. It will track and showcase all the contributions committed to leading Dev communities on the internet. Therefore, users no need to

update them with their recent works. The proposed platform is smart enough to identify and update itself according to user's recent contributions. Since all the information are precisely optimised for search engines, these portfolios will appear in the search results of major search engines such as Google, Bing, etc.

According to the recent researches [22], newbies tend to follow a role model in their respective fields or their dreamed career path. When it comes to the IT industry, developers often use LinkedIn, GitHub, Stack Overflow, and Medium to follow their role models [23]. But the problem here is developers have to stay in touch with all of these platforms to get a big picture of their role model.

They do not have a way to track and analyse all the significant contributions done by their dream role model in a one-stop. Figure 1.5 clearly proved that developers are suffering from not having this kind of platform to follow. Since the proposed platform has all of these details ordered precisely on a timeline, it will give newbies a solid idea of how they should shape their path accordingly to pursue their dreamed profession or career goals by seeing an expert's portfolio.

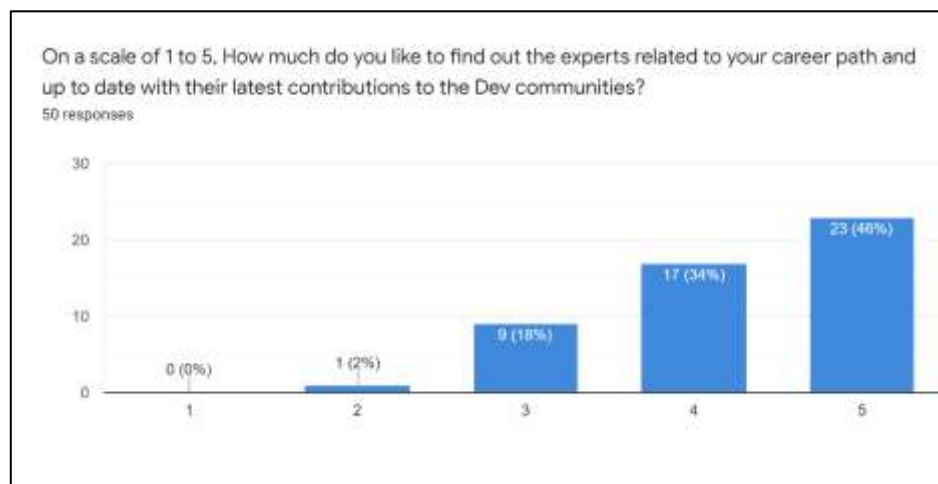


Figure 1.5 – Summary of likeness to find and follow the experts related to the career path.

Another major problem developers still suffer from is that no platform offers a method to get in touch with the experts. Using ProbExpert, the proposed platform, users will be able to get or book a session with their chosen experts by spending his/her earned points on the platform. The feature mentioned above is the most awaited one for many newbie developers. Since getting an answer to a posted question is a very time-consuming task, users can use this feature for emergency work.

From the view of an expert, no matter how many contributions have been made for a specific platform, there is less reputation when moving from different platforms. Because every platform generates users' reputation by only considering the contribution towards the platform regardless of the current proficiency of the developer. Therefore, every user has to earn reputation points from the ground up. Especially experts who will not receive a proper reputation for proficiency at the beginning tend to quit or neglect such platforms as building a fitting reputation is time-consuming [24]. Without having an adequate recognition of the knowledge, contributions to questions may be often overlooked by other users misunderstanding the expert as a novice.

Social coding platforms usually provide user-based statistics but do not summarise a developer's programming skills or highlight the programming languages that a software engineer masters during her/his active years of contributions [11]. Although other professional social networks, like LinkedIn [12], provide means to capture users' expertise in more intangible ways, e.g., via endorsements from peers, they do not provide insights into the actual hands-on experience a user has gained. They just rely instead on opinions from a user's connections in order to create a view of his/her expertise.

1.4 Research Objectives

Main Objective

- **Detecting users' proficiency level along with an auto-generated portfolio**

To Address the main objective, A system has been developed to calculate developers' expertise according to their behaviors and activities on social code management platforms and community questions answers platforms. In this research, GitHub and StackOverflow have been selected respectively for the scenario mentioned above. Advanced custom-made web crawlers have been used to gather the developers' information. Then, the integrated scoring model predicts the developers' proficiency level (Newbie, Intermediate, Professional, Expert) and calculates a score for the developer. Finally, the system generates a fully fledged portfolio for the user by filtering out unnecessary details from the gathered data set. This portfolio contains all the significant opensource contributions, processed git statics, employment history, and more. Since most of these details are ordered precisely on a timeline, it will give newbies a solid idea of how they should shape their path accordingly to pursue their dreamed profession or career goals by seeing an expert's portfolio. Furthermore, this system is equipped with a leaderboard, which ranks all the registered users according to their generated score. Portfolios and this leaderboard will help recruiters identify the candidates' skills and proficiency level and ease the process of finding skillfully unemployed developers in no time.

Specific Objectives

In order to reach the main objective, the specific objectives mentioned below have to be fulfilled.

Serverless RESTful API to retrieve data from Github GraphQL service

GitHub provides a GraphQL API to query all of its public data, which makes it the primary resource for retrieve developer statistics for this research. Since it has a strict rating limit (5000 requests for an hour), and a response query for a developer does not changing frequently, A serverless cloud API called Git-Data-Miner (GDM) has been developed from our end to retrieve users' data and to stop duplicate requests within 24hrs in order to keep the request count below the Github rate limit. The developed restful API has been used throughout this research to communicate with GitHub to query all the required information.

Web Scrappers to gather other necessary developer's statistics

Since other platforms such as StackOverflow and LinkedIn do not provide a public API to collect user-related information, two different scrapping tools have been developed from the ground up to scrape the necessary pieces of information. BeautifulSoup4 and Selenium were mainly used to build these scrappers based on Python3. Since the system's main backend server is based on Node.js, its child process functionality has been used to run the python scrappers on top of it.

Scoring model to generate a score for developers activities and behaviours

To develop a scoring model to generate a score based on developer statistics on GitHub, First need to identify and extract the meaningful feature values out of the raw data set. Then adding each feature for specific weight, defining rewarding and penalty

factors by considering the sample dataset of 50 developers, the proposed scoring model has been developed.

Ranking algorithm to classify developers into suitable classes

Normal CDF has been used to generate the normalised score for each developer according to the received score from the scoring model. Then using the bell curve graph, the user has been classified into the related class (Newbie, Beginner, Intermediate, Advanced, Guru) accordingly.

Automated portfolio generation functionality

By consuming developer's GitHub, Stack Overflow, and LinkedIn's scraped data, it generates the portfolio by showcasing the developer score and other valuable and significant developer activities using timelines, progress bars, and graphs.

2 METHODOLOGY

2.1 System Overview

The ProbExpert developer ranking system consists of three individual scorer models to evaluate developers' skills in three different major categories: Coding, Q&A, and developer professional achievements. GitHub, Stack Overflow, and LinkedIn are used respectively to identify and evaluate the areas mentioned above of a developer. In these three communities, GitHub is the only platform that offers a public REST API to its complete data set. Therefore a tool called Git-Data-Miner has been developed from our end to communicate with the GitHub Graphql API. It helps to keep the ProbExpert backend and the frontend isolated from the Graphql API and minimize the code complexity of those two repositories.

Then to scrape data from GitHub and LinkedIn, separate two web scrapers have been developed from the ground up using BeautifulSoup 4 and Selenium. Python3 was used to develop these scrapers. Since ProbExpert's central server runs on Node.js server, its child processes have been used to run these python script on top of Node server. The high-level architecture of the system is shown in Figure 2.1.

Furthermore, when evaluating a developer, coding skills and contributed projects play a significant role. Therefore, the ProbExpert ranking algorithm gave the 80% of the total weight to the GitHub scorer model while giving only 20% to the other two models when finalizing the total score of a developer.

ProbExperts frontend has been developed using Next.js on top of the React.js library to enable the server-side rendering to achieve better SEO results and better performance. Therefore, all the generated portfolio on the platform has been stored on the server fully rendered. This enables all the major content delivery networks (CDN) to cache these pages to provide faster access. GitHub OAuth2 service has been used to verify the users who generate their portfolios. Since the portfolio user model is a bit complex and consists of many dynamic data and most of them are not available for all the users, a non-relational database needed to be selected. Therefore MongoDB cloud

Atlas database has been selected as the central database for the ProbExpert platform. Each component of the High-level diagram will be thoroughly discussed in the coming up sections.

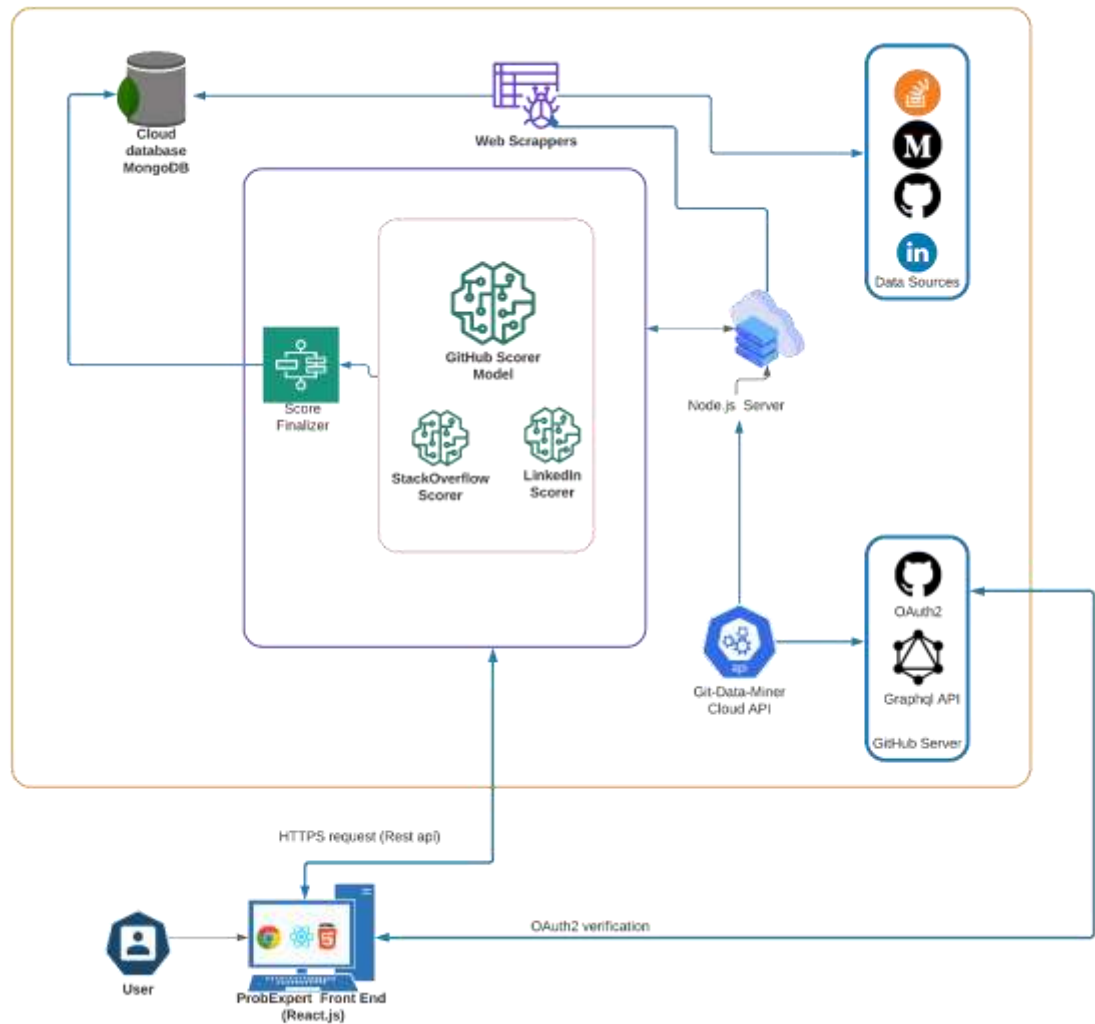


Figure 2.1: High-level architecture diagram

2.2 Data Gathering

Three different scrapping tools have been developed from the ground up to collect the developers' data from GitHub, Stack Overflow, and LinkedIn. Since Stack Overflow and LinkedIn do not provide a public API to gather their users' data, BeautifulSoup4 and Selenium were used to build these two scrappers. However, unlike others, GitHub provides a GraphQL API to query all of its public data. Since it has a strict rating limit (5000 requests for an hour), and these response query results not changing frequently, A tool called Git-Data-Miner has been developed from scratch to retrieve users' data and to stop duplicate requests within 24hrs in order to keep the request count below the GitHub rate limit.

2.2.1 Git-Data-Miner

Git-Data-Miner tool has been developed from the ground up to communicate with the GitHub Graphql API. As discussed in the above section, to keep the query amount below the limit and stop unnecessary duplicate requests, request caching feature has been enable using the cloud function of the Vercel cloud platform. More details about the implementation and the deployment of the application will be discussed in section 3.2

2.2.2 Dataset

To find out the useful features that help to determine users' proficiency, the first step is to collect all the available behaviors and activities via GitHub Graphql API. Then valid features have been selected by analyzing each feature thoroughly visualizing the

statistical relationship between them using Seaborn library based on matplotlib. Therefore using our Git-Data-Miner tool, we query the below data via GitHub GraphQL API.

- Name
- Account Created Date
- Is Hireable
- Number of Organizations worked at
- Total Commit Contribution
- Number of repositories contributed to
- Total Pull Request count
- Total Issues count
- Total Individual Repositories
- Total Followers
- Total Stargazers
- Total Discussion Comments
- Total approved comments as answered

The above-listed data can be query directly via the given GraphQL API. However, since this data is not enough to come to a good conclusion about a user's proficiency, few custom methods have been used to filter out valuable information through this API. These custom methods will be discussed in the implementation section in detail. These methods focused on finding the users opensource contribution thoroughly. So, user's open-source contributions and individual repositories have been divided into three sections to get more valuable information. Below is the list of details filtered out from these custom methods. In a GitHub repository, Stargazer is another GitHub user who has starred the repository to show their appreciation and bookmark it for later use. Therefore, the total number of stars received by a repository is considered as a solid metric to determine whether the project is well tested, mature and popular among the developer community [23].

- High Contributed Repositories (GitHub Stars \geq 10,000)
- Medium Contributed Repositories (10,000 $>$ GitHub Stars \geq 5000)
- Low Contributed Repositories (5000 $>$ GitHub Stars \geq 500)
- High Individual Repositories (GitHub Stars \geq 1000)
- Medium Individual Repositories (100 $>$ GitHub Stars $<$ 1000)
- Low Individual Repositories (GitHub Stars $<$ 100)

Then to determine the effectiveness of the above features to evaluate a developer skill, 50 GitHub profiles have been selected and extracted using the Git-Data-Miner. Since the main purpose is to find out how these features behave according to the users' proficiency level, a balanced dataset based on the stratified sampling method has been chosen. This balanced dataset consisted of users who belong to the below categories.

- Top Opensource Contributors
- GitHub Star Developers
- Software Engineers
- Software Engineering undergraduates
- Beginners (profiles which aged less than one year)

Here, as Top Opensource contributors, chosen the best-renowned developers in the opensource software field, Dan Abramov (Author of Redux), Taylor Otwell (Author of Laravel), and Evan You (Author of Vue.js). Since their projects are available on GitHub as open-source repositories, their profiles have received the top attentions of the Opensource contributors. Therefore, analyzing these profiles, easily find out how

these features behave in the experts' profiles. However, edge cases such as Linus Torvalds (Author of Linux) have been removed from the dataset to improve the accuracy of the analysis.

GitHub Star Developer is a program that GitHub offers to find out the best developers and open-source contributors using community feedback. In there, developers can nominate the best impactful developers who contribute to the community constantly. Therefore, only the top developers got selected as a GitHub Star in a year. So, after the top renowned opensource authors, GitHub stars have the most impactful accounts in GitHub.

Then to represent the other user categories, Software Engineers who contribute to and maintain their GitHub account selected as 3rd class of our dataset. As the 4th class of the dataset well maintained 4th year Software Engineering undergraduates accounts have been selected. Finally, to identify the beginner GitHub users' behaviors, accounts aged less than one year have been selected. So each class consists of 10 GitHub profiles.

2.2.3 Data visualization and feature extraction

To analyze each feature behavior against the selected classes, Jupyter notebook has been created from the ground up using the Seaborn library to visualize the statistical relationship of the data.

- **Number of followers**

GitHub provides followee, and follower features like most social media platforms do today. There, developers can follow the other users as they want but cannot control the

number of followers. Therefore, this metric is an excellent source to identify how the follower counts change according to each class.

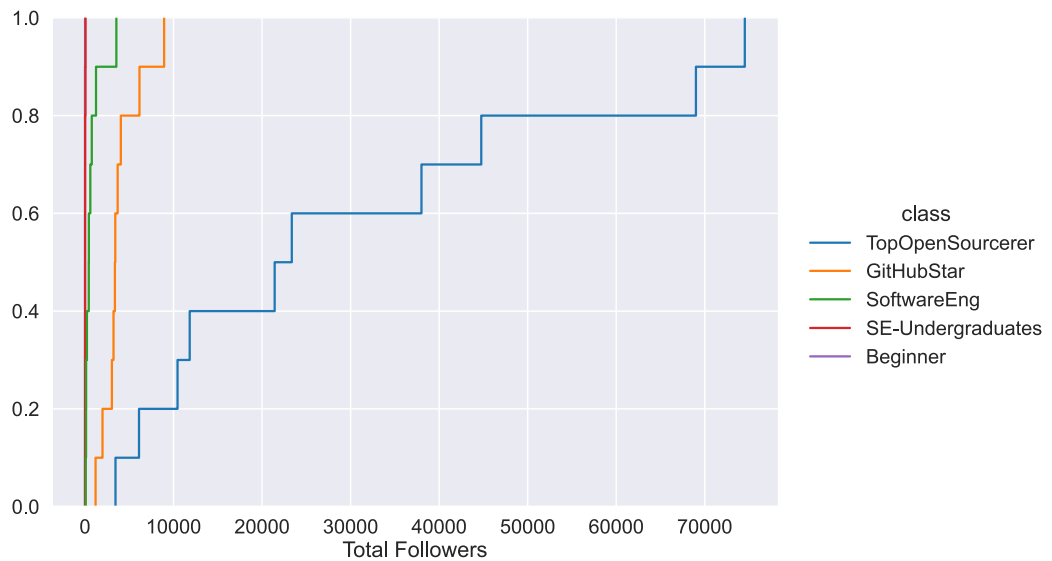


Figure 2.2: ECDF distribution of total followers count

Figure 2.2 illustrate the cumulative distribution of the total followers against each class. According to the distribution, more than 40% of Top-Opensource contributors have 40,000 or more followers. On the other hand, undergraduates and beginner developers do not have many followers at all. Therefore, this metric has been selected as a valid feature to identify and evaluate GitHub users' proficiency.

- **Number of Programming languages**

Graphql query has been created to loop through each contribution to gather the programming language to determine the number of languages a developer uses for their coding. Since GitHub API only allows to query contribution details of last 4 years of a developer, It is impossible to identify all the languages used by a developer with older accounts.

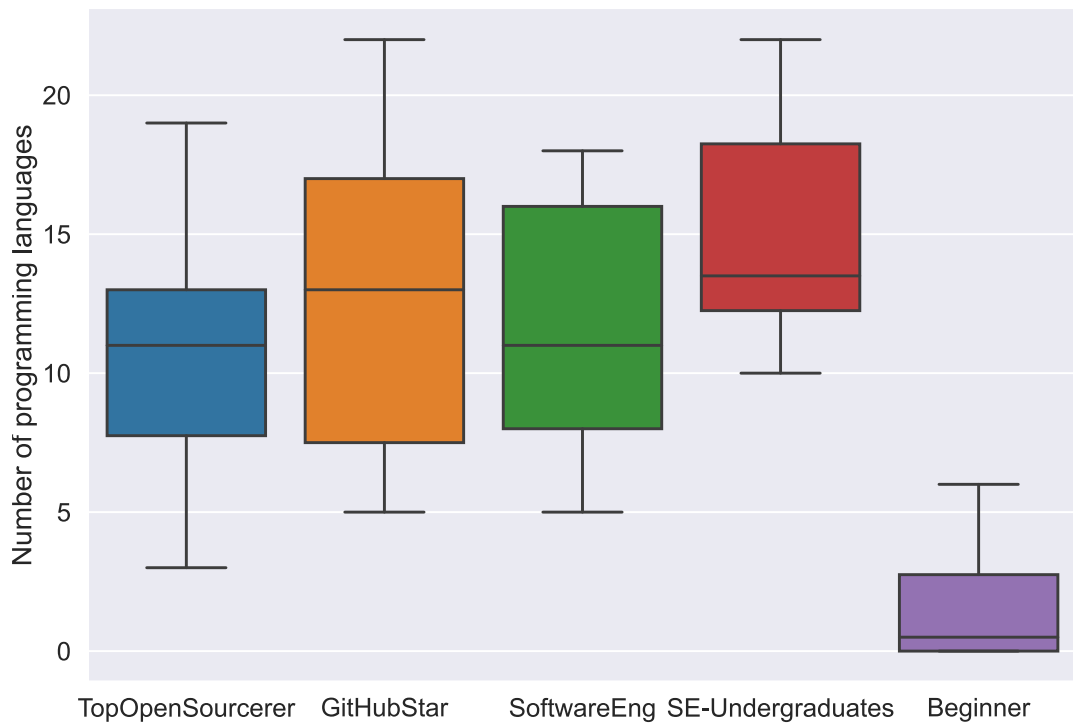


Figure 2.3: Boxplot graph of the usage of programming languages

According to Figure 2.3 software engineering undergraduates have the maximum average for programming language usage. It makes sense because an undergraduate has to go through many programming languages in their four years of academic life. Furthermore, this graph shows that when a programmer gets more experience, they tend to stick with fewer programming languages. Since there is no possible way to query language records more than five years deep, this metric has been rejected from our scorer model.

- **GitHub Profile Age**

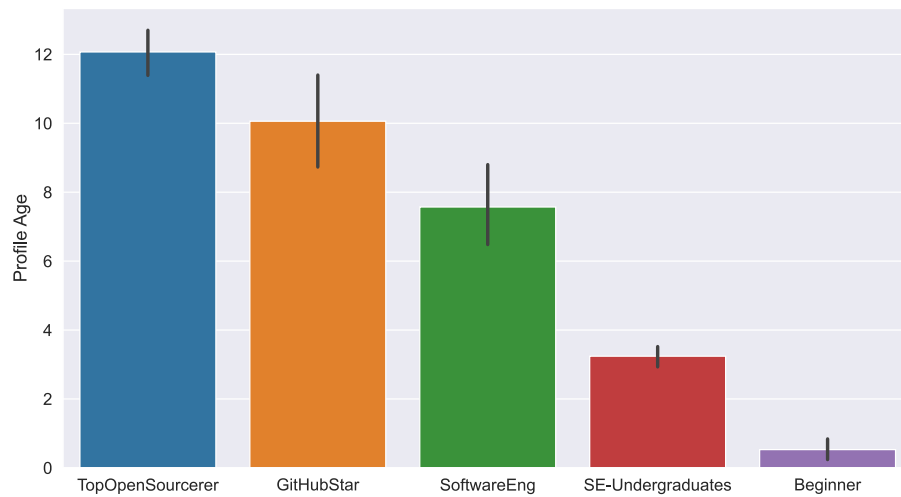


Figure 2.4: GitHub profile age distribution

Figure 2.4 Bar plot graph shows the average age distribution against the classes. Here we can clearly see that age gradually rise from beginner to top opensource contributors. Therefore, this metric has been considered as a useful feature to determine user proficiency.

- **Availability to work**

Graph of Figure 2.5 illustrates the availability status for work of each class. According to the bars of top opensource contributors and beginner developers, we can identify they do not prefer to work. However, in GitHub, the default status of this option is false. Therefore, beginners might be not aware of it to turn it on. However, since there is no valid relation between these data, this feature was rejected from the scorer model.

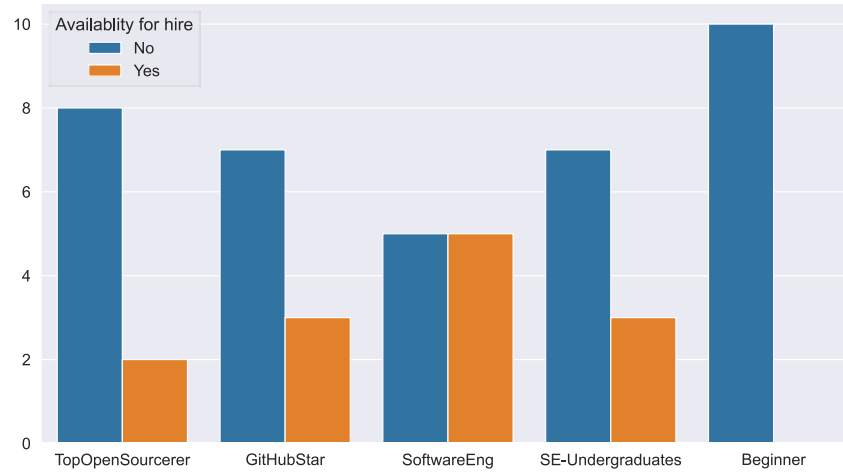


Figure 2.5: Availability for hire

- **Contribution to the Opensource repositories**

Since GitHub GraphQL API does not provide a direct way to query this information, a custom query has been implemented to retrieve the contributed opensource repositories data. Then in the Git-Data-Miner, these repositories have been filtered into three different arrays according to their significance, which are respectively discussed below.

Repositories that have more than 10,000 stargazers have been selected as High Opensource repositories. Since repositories that have over 10,000 stargazers are well known and maintained up to the community standard. Therefore, contributions made to these kinds of repositories are considered as significant contributions. Furthermore, it requires lots of experience and knowledge.

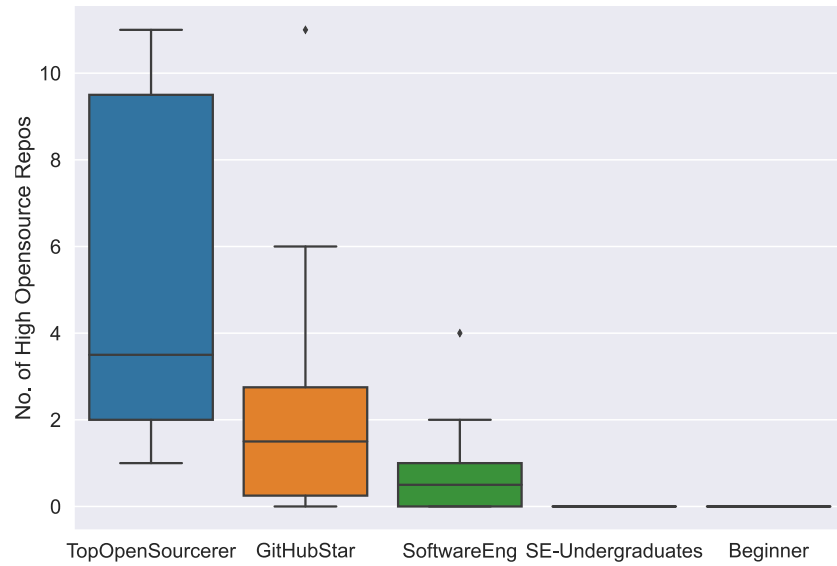


Figure 2.6: Contributed Repositories (GitHub Stars \geq 10,000)

Figure 2.6 depicts a gradual increase from beginner developers to top opensource contributors. Therefore this feature has been considered as a valid metric for the scorer model.

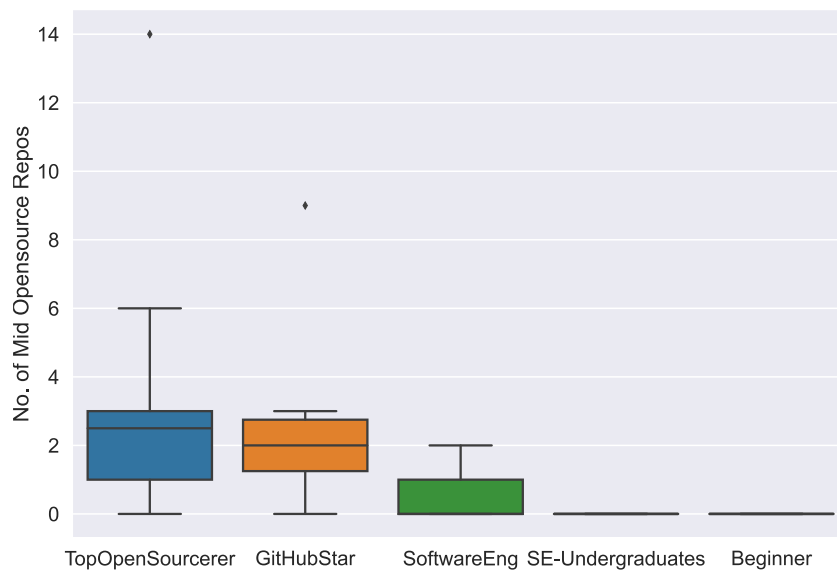


Figure 2.7: Contributed Repositories (10,000>GitHub Stars \geq 5,000)

Figure 2.7 also illustrates a gradual increase from beginner developers to top opensource contributors. However, in this graph, the gap between each class has reduced. Considering the clean relation between the data, this feature has been selected as a valid metric for determining user expectancy.

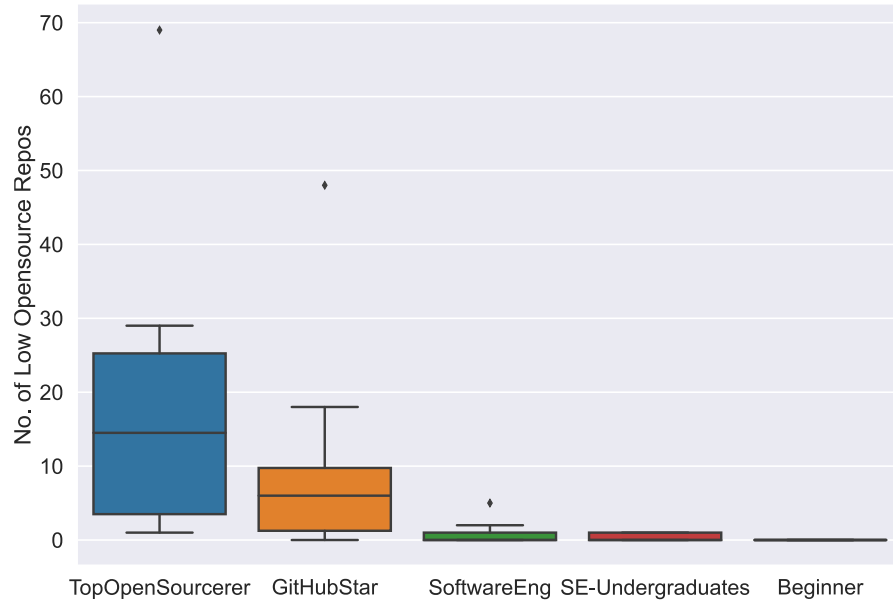


Figure 2.8: Contributed Repositories ($5,000 > \text{GitHub Stars} \geq 500$)

According to Figure 2.8, undergraduates also have made contributions to the low range opensource repositories. Considering the data distribution between the each class, this feature also have been selected as a valid feature for the scorer model.

- **Total Pull Request Count**

Figure 2.9 depicts the boxplot graph of the pull request made by each class. Considering the gradual increase of the pull request count from beginners to top opensource contributors, this feature has been selected as a helpful indicator to determine the user's proficiency.

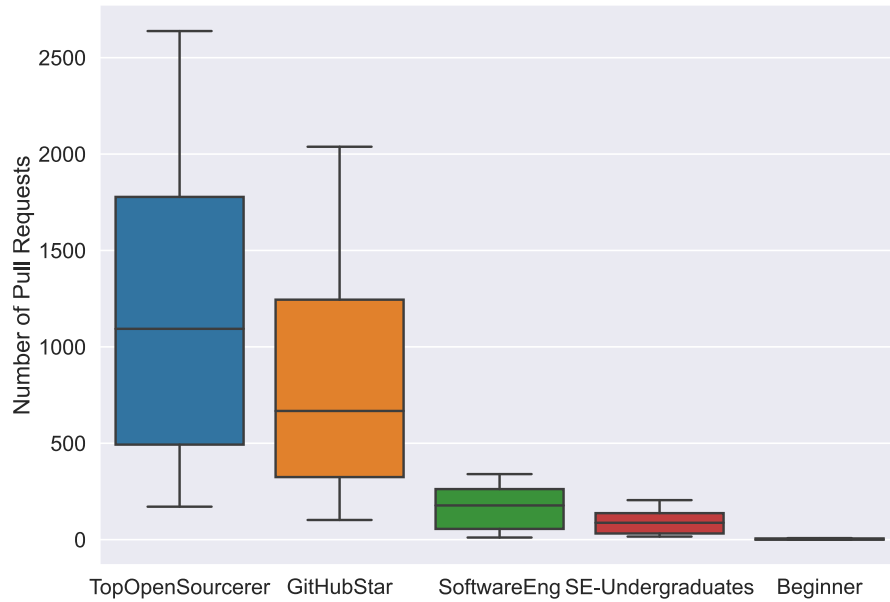


Figure 2.9: Total Pull Request Count

- **GitHub Discussion Statics**

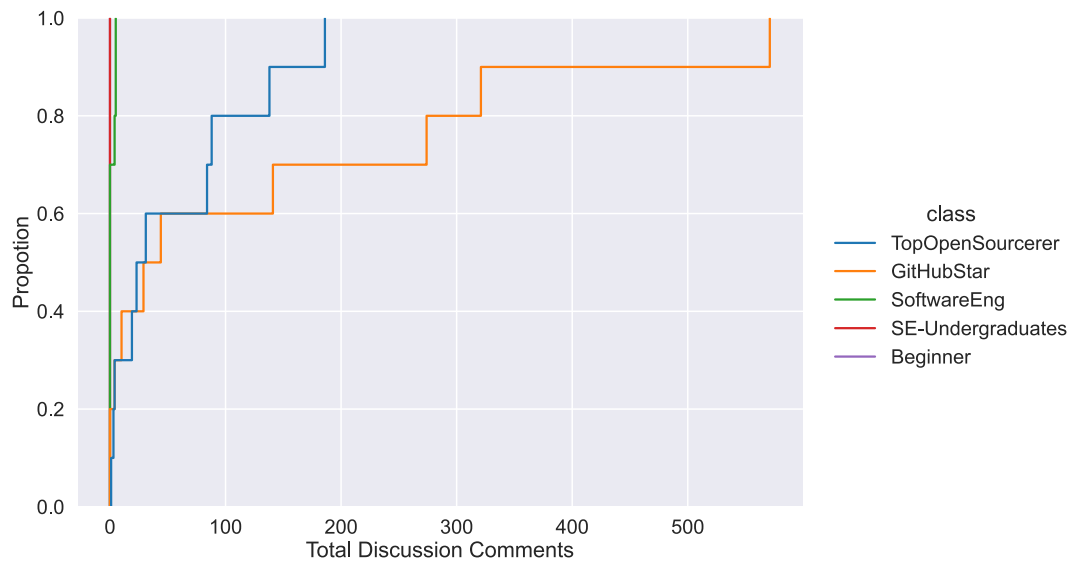


Figure 2.10: ECDF graph of the discussion data

GitHub Discussions is a collaborative communication forum for the community around an open-source project. Developers can ask questions about the issues, bugs,

improvements, technologies, etc. The experts who know the answers can comment in these discussions, and maintainers of the repositories can approve them as valid answers. Figure 2.10 shows the empirical cumulative distribution function of each class. According to the graph, GitHub-Star developers has participated in these discussions more than any other class. Furthermore, Figure 2.11 confirms the above theory by showing the ECDF of the approved answers. Therefore, this feature also has been considered as a valid metric to identify experts developer in GitHub.

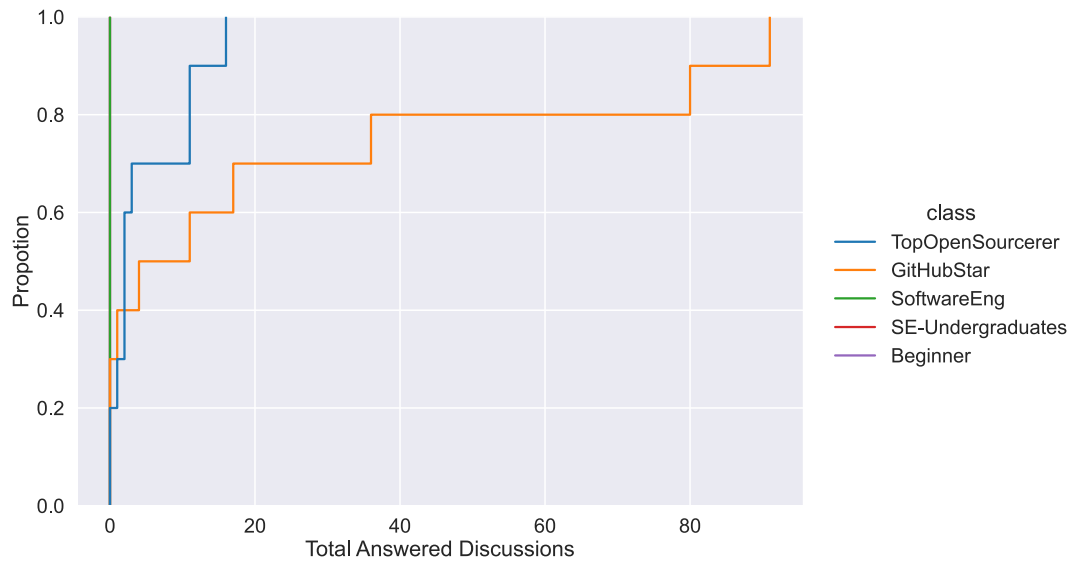


Figure 2.11: ECDF graph of the approved comments as answers

- **Total Received Stars**

Since GitHub GraphQL does not provide a direct way to find the number of stars a developer received for their individual repositories, a custom method has been developed to loop through all the individual repositories to gather the total amount of received stars. According to Figure 2.12 there is a significant relationship between each class. Therefore, this feature has been considered as a valid metric for measure developer proficiency.

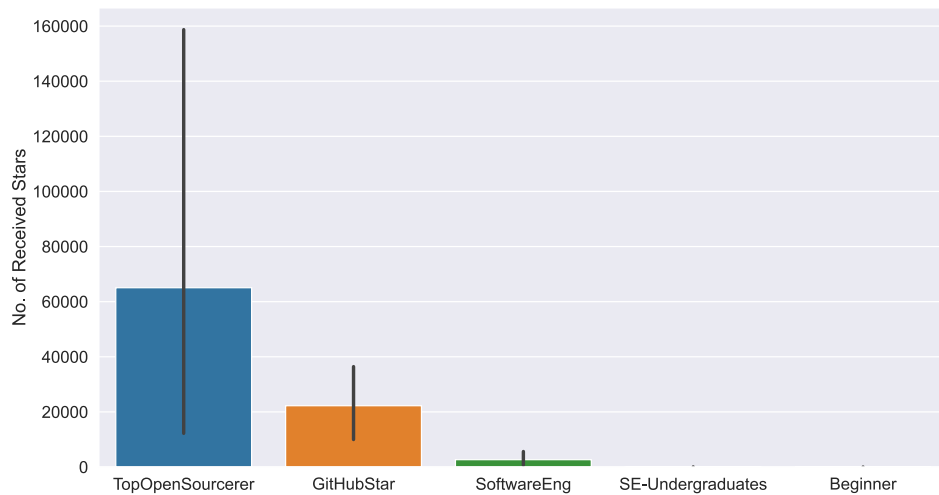


Figure 2.12: Distribution of the received stars

- **Created Issues**

GitHub Issue a solution to keep track of tasks, enhancements, and bugs for a repository. Most software projects have a bug tracker of some kind, and GitHub's tracker is called Issues, and has its own section in every repository.

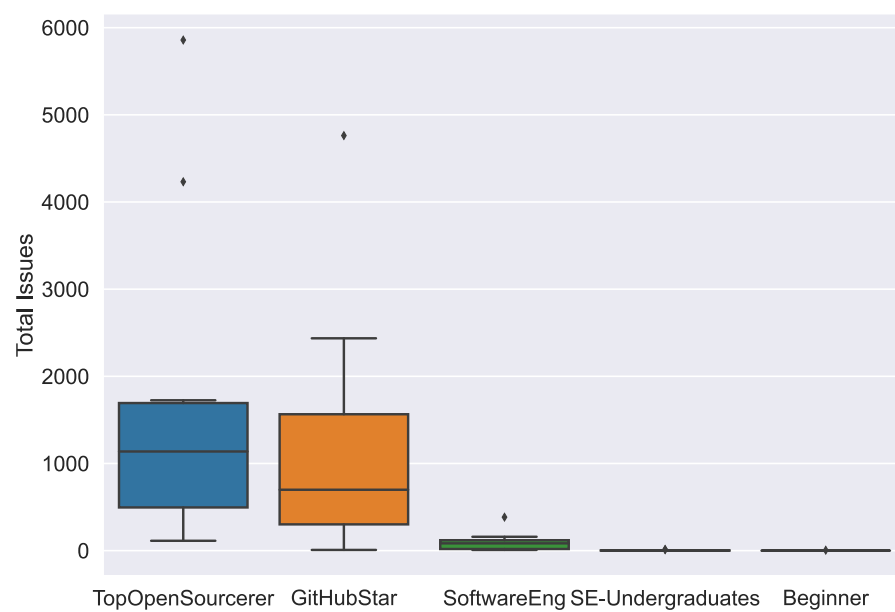


Figure 2.13: Box plots of the created issues

Figure 2.13 depicts the boxplot graph of the total issues created by each class. Considering the gradual increase of the total issue count from beginners to top opensource contributors, this feature has been selected as a helpful indicator to determine the user's proficiency.

2.3 GitHub Scorer Model

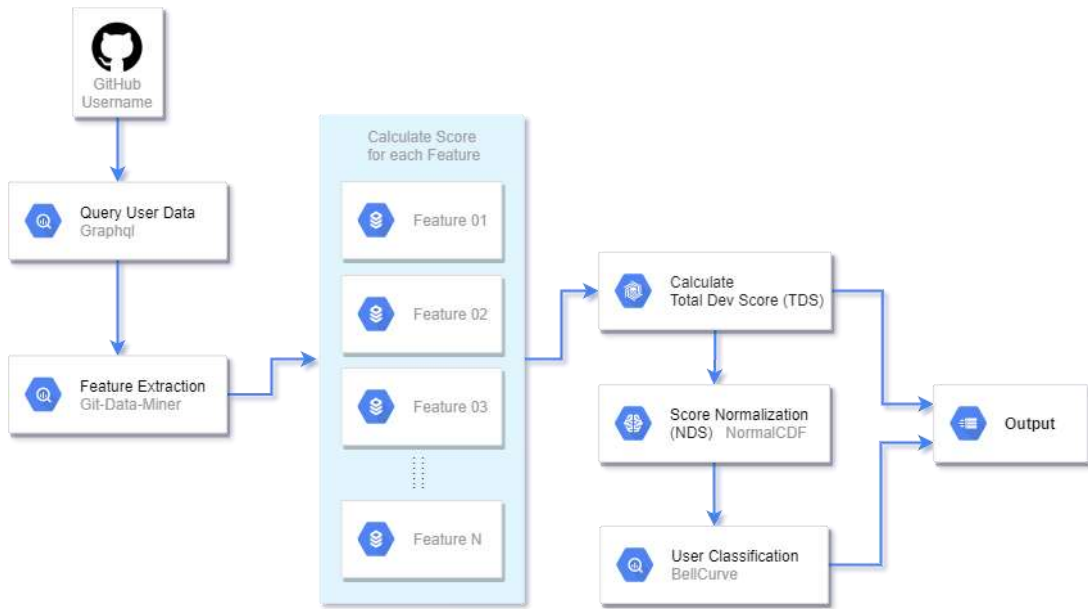


Figure 2.14: High level diagram of the GitHub Scorer

The primary scorer model, which is GitHub scorer consists of 4 stages. In stage one, it gathers user data via GitHub GraphQL API for the given username. Then inside the Git-Data-Miner, it filters all the required features discussed in section 2.2.3 for scoring. Then by multiplying each feature's weights with given user's feature values, the individual feature score is calculated. As mentioned in section 2.2.3, if the values user gained for the mentioned specific features are less than their lowest threshold, they are considered as penalty activities. Therefore the penalty score will be added to the total score as well. After calculating the total dev score, In stage 3, to determine the rank of the selected developer, the normalized score is calculated based on the Bell-curve[25] by consuming the derived TDS value from the previous step. The normal cumulative

distribution function (normalCDF) has been used to derive the normalized dev score from the generated score instead of using empirical rule. since empirical rule does not give the precise value for the result, ranking users based on those values will not be accurate. However, it is impossible to calculate the cumulative distribution directly in JavaScript because it does not provide any built-in method to calculate the error function (erf) [26]. Therefore, by using the erf method of math.js opensource library, the normalCDF function was implemented.

$$NDS = \frac{1}{2} \operatorname{erf} \left(\frac{TDS - TW}{\sqrt{2} TR} \right) \quad 2.1$$

According to equation **Error! Reference source not found.**, NDS gives the normalized developer score using TDS, summed values of assigned weight for each factor (TW), and the summed value of the weight of each rank (TR).

2.3.1 User Classification based on Bell Curve

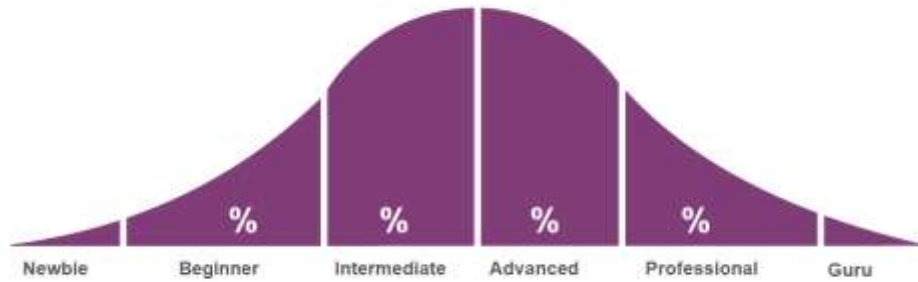


Figure 2.15: Classes distribution on a bell curve

Figure 2.15 illustrate the classes distribution on a typical bell curve graph, but in ProbExpert's algorithm a separate bell curve graph generated for each developer

according to their NDS which is derived from TDS value. Since endpoint value and the standard deviation do not change, each bell curve will be created according to the TDS values which consider as the mean of the above equation **Error! Reference source not found.** Therefore, distribution of the class ranges has been flipped to reverse the effect. Table 2.1 shows the updated ranges of the distribution. Finally, by considering the left tailed p-value from the generated bell curve, a user is classified into the related class according to the values of Table 2.1

Table 2.1: Distribution of the Classes

Class	Range
Newbie	100-81
Beginner	80-56
Intermediate	55-46
Advanced	45-26
Expert	25-5
Guru	5-0

Below 3 test cases show how this classification works under the hood.

Table 2.2: User info - test case01

Test case	01
Username	prabhuignoto
TDS	62.58
NDS	39.0641

As the test case 01, an account of a Software engineer has been selected. Since TW and TR values do not change as explained earlier, below is the generated bell curve for this developer statistics.

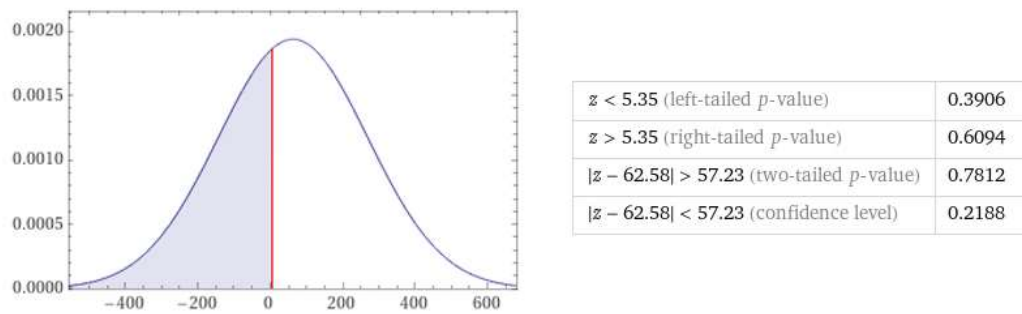


Figure 2.16: Bell curve - test case01

According to the bell curve of Figure 2.16, there is about 39.06% of area has covered from the full curve when considered the left tailed p -value. Therefore, this developer will be classified as an Advanced developer according to the Table 2.1

Table 2.3: User info - test case02

Test case	02
Name	Alex Ellis
Username	alexellis
TDS	779.011
NDS	0.0086

As the test case 02, an account of a GitHub Star developer has been selected. Only few get selected as GitHub stars according to their contributions to the opensource community. below is the generated bell curve for this developer statistics. Since this user has very high value for TDS, left tailed p value of the generated bell curve has really small value, which is close to 0.0086%. Therefore, according to the Table 2.1 this developer will be classified into the Guru class.

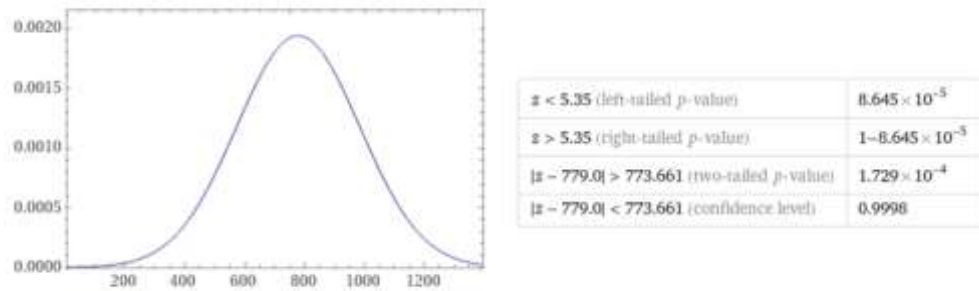


Figure 2.17: Bell curve - test case02

Table 2.4: User info - test case03

Test case	03
Name	Sathsara Gajanayake
Username	sathsaragajanayake
TDS	-164.914
NDS	79.5746

As the test case 03, an account of an beginner, who currently studies the 2nd year of his IT degree has been selected. Since he does not have any significant activities yet in his profile, he has received a minus value for TDS. Below is the generated bell curve for his contributions.

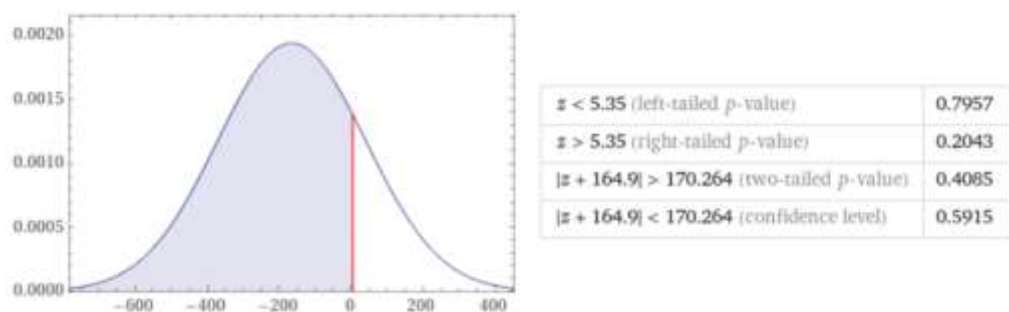


Figure 2.18: Bell curve - test case03

Table 2.1: Distribution of the Classes

Figure 2.18 illustrate that there is more than 79% of the bell curve has been covered from the left tailed p value in other words generated NDS value. Therefore, according to the Table 2.1 This student get classified into the Beginner class.

2.3.2 StackOverflow and LinkedIn Scorer Models

Stack Overflow and LinkedIn scorer models have also been developed using the same manner used to develop the GitHub scorer model. User Reputation, Total answered tags count, Total Number of Badges with their types, the number of endorsements, and the number of peer connections have been used as the features for these two models. Finally, the developer's overall score is calculated by giving 80% of the weight to the GitHub score and 10% each for the other two scorers.

2.4 Automated Portfolio Generation

By consuming the given usernames of GitHub, Stack Overflow, and LinkedIn of a developer, ProbExpert's portfolio system will first scrape and collect the developer activities and calculate the developer score as explained in the previous sections. Then it generates the developer portfolio by showcasing the developer score and other valuable and significant developer activities using timeline progress bars and graphs.

Furthermore, it includes a leaderboard that assists recruiters and other users in finding out the experts on the platform. Since these portfolios needed to be optimized for search engines, Next.js [30] has been used on the top React.js library to enable static rendering. Therefore, all the generated developer portfolios can be cached on all the major content delivery networks (CDNs), and also, they are 100% accessible by web spiders to rank them on major search engines. Also, Next.js hybrid-static page generation method is used for loading the portfolios without any server-side computation by serving the pre-rendered pages stored in the server.

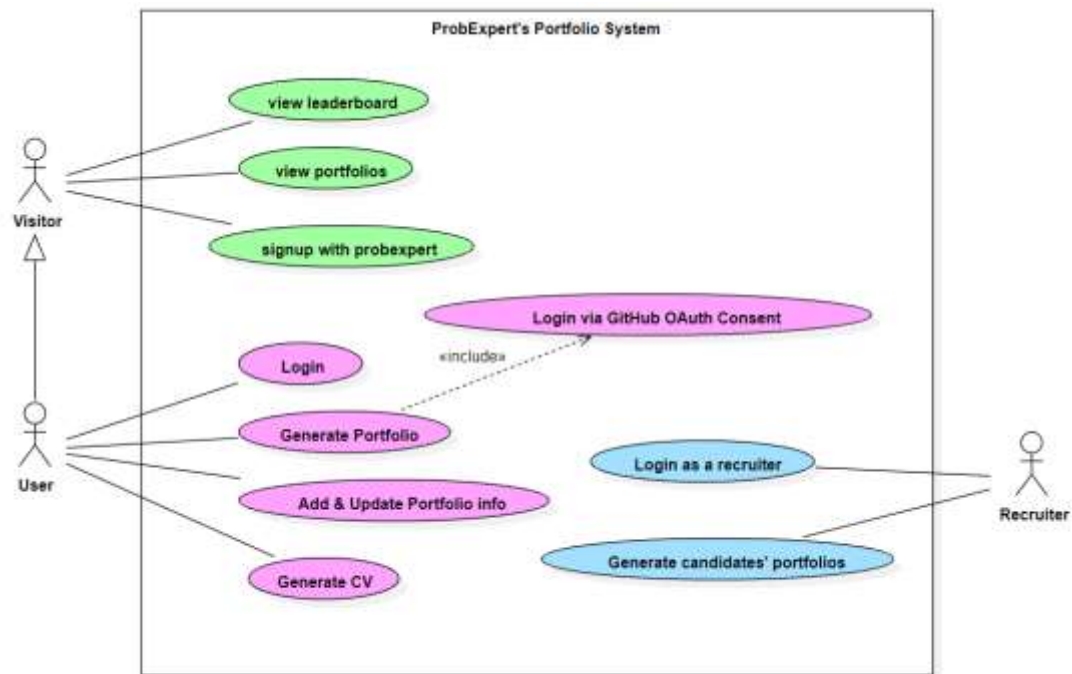


Figure 2.19: Use-Case Diagram of the Portfolio System

Figure 2.19 shows the use-case diagram of ProbExpert's portfolio system. Since need have verification system to identify users enters their own profile usernames. GitHub OAuth 2 service has been used. So, after they enter their usernames in the given dialog box and press generate button, they will redirect into the GitHub's consent screen. After they successfully enter their credentials into the consent screen, by validating the returned OAuth code from the GitHub, the user portfolio will generate and get attached to their main ProbExpert's profile. Then user has full access to its content and he or she can update the open details such as bio, profile image, etc. However, they do not get access to change the information generated from analyzing their social coding profiles. After making the necessary changes to their portfolios, finally, they can download the Online Portfolio in PDF format by clicking on the cv generate button.

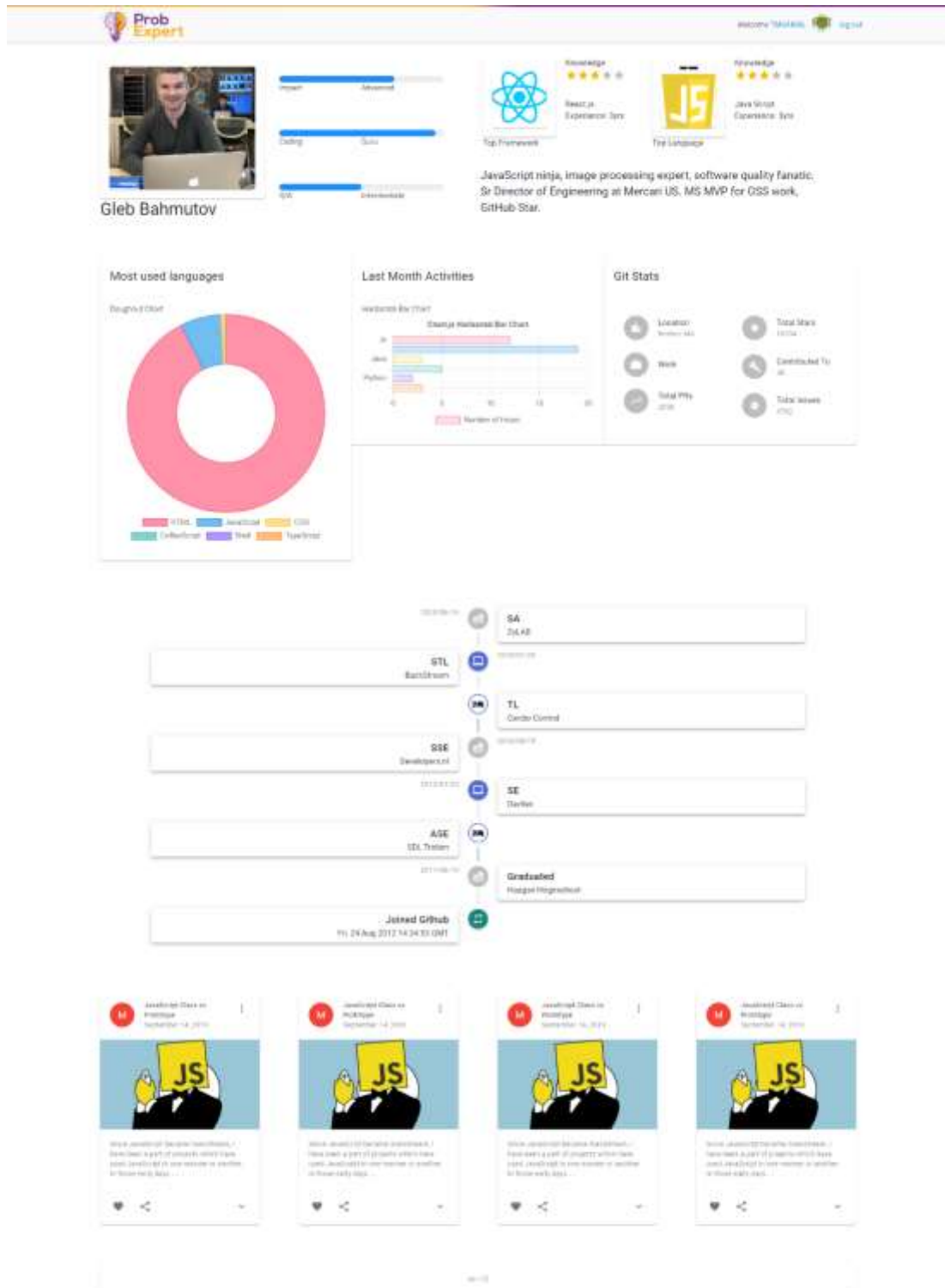



Figure 2.20: Portfolio UI (WIP)



[New](#)
[Sign up](#)

[To my portfolio](#)

[Generate portfolio](#)

LeaderBoard

Rank	Name	Username	Score
1	Loris Torvalds	lorvalds	208594.597
2	Taylor Otwell	taylorotwell	175281.266
3	Daniel Eisenberg	edgier	14457.837
4	Fabien Paternoster	fabpat	12774.286
5	Evan Yeh	yeh998883	10788.030
6	Don Alexander	gawron	9647.854
7	Sebastian Bork	sebastianbork	9197.657
8	Aditya Venkatesh	adityaVenkatesh	8884.243
9	Oliver Wink	olowink	3805.733
10	Oliver Sattmann	sattmann	2742.796
11	Katherine Pisch	katchp	2227.825
12	Madhey Muckham	madhey	2130.052
13	Andrey Stets	stet	1988.778
14	Oliver Heuliger	foeser	1889.888
15	Sebastian Raju	sebastianraju	1884.527

Show per page: 15 25 50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950 1000

15

25

50

100

150

200

250

300

350

400

450

500

550

600

650

700

750

800

850

900

950

1000

Figure 2.21: ProbExpert Leaderboard

2.5 Commercialization aspects of the product

Introducing ProbExpert into the market is a significant goal in this project. Below Figure 2.22 **Error! Reference source not found.** illustrate the marketing plan and strategy we made for ProbExpert commercialization purpose.

2.5.1 Premium Accounts for Hiring Managers

As discussed in the literature survey and research gap, recruiting managers still do not have a single platform to evaluate all their candidates in a single place. Therefore, they have to go into each platform like GitHub, StackOverflow, and LinkedIn to evaluate each candidate. They can get rid of all the time-consuming hard work by just creating a recruiters premium account in ProbExpert platforms. Using the recruiter premium, they can get all the advantages listed below. Utilizing the following perks, organizations can save countless minutes on CV shortlisting and find the top unemployed developers in the field in no time.

- Easily identify all the unemployed top developers using the leaderboard.
- Evaluate their candidates' proficiency and contributions to the software field via generating the profile within a single click.
- Compare all their candidates by looking at the generated score for each of them.

2.5.2 CV generate option as a premium service

By using ProbExpert portfolio system, developers can maintain their online portfolio as they want. To attract more users into the system and for the platform's popularity, this service will be offered to developers as a completely free service. However, downloading their online portfolio as PDF can be provided to the users as a premium option by considering users' feedback.

2.5.3 Section for display Open job Opportunities

In ProbExpert, we could provide a separate space for organizations to post their job opportunities. This implementation also has a positive effect on getting more

developers to the system and turning them into daily users. This service could provide to the organizations as a pay-as-you-go service.

2.5.4 Display Sponsored Advertisements

Since this platform has a specific set of users, Organizations that want to promote their new technologies and services can post their advertisements on this platform. Since advertising has a negative effect on the platform's reputation, this is not a finalized strategy yet.

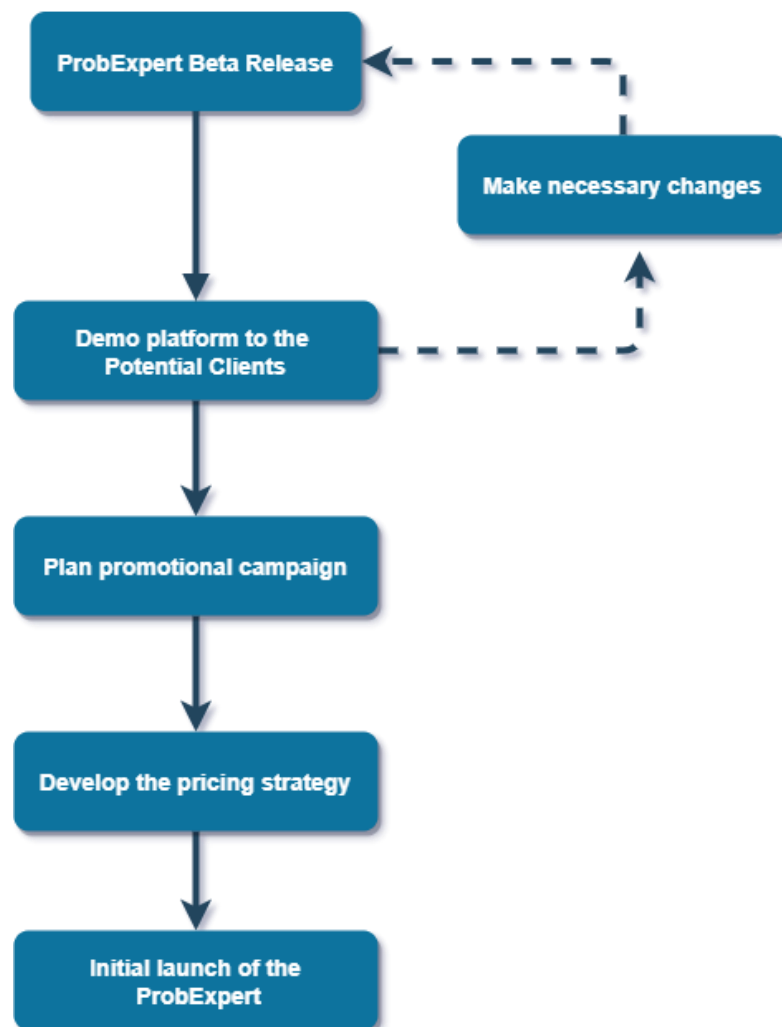


Figure 2.22: ProbExpert Marketing Strategy

3 TESTING & IMPLEMENTATION

This section describes how the system was implemented and tested under several testing techniques to minimize the issues.

3.1 Testing

For any application, testing plays a significant role in the success of the application. Quality testing procedures can reduce the number of bugs in the application and identifying them before the release is essential and cost-effective.

```
kamal@LAPTOP-4J67MM59 MINGW64 /e/Experiment_Ground/probexpert/Git-Data-Miner (development)
$ npm test

> GIT-DATA-MINER@1.0.0 test E:\Experiment_Ground\probexpert\Git-Data-Miner
> jest --coverage

PASS tests/calculateScore.test.js
PASS tests/userInfo.test.js
PASS tests/repoInfo.test.js
PASS tests/featureWeights.test.js
PASS tests/classifyUser.test.js

-----
File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----
All files | 80.46 | 63.03 | 70.27 | 81.14 |
api | 88.64 | 65.52 | 100 | 88.64 |
  get-user-info.js | 88 | 68.42 | 100 | 88 | 35,39,61
  get-user-repos.js | 89.47 | 60 | 100 | 89.47 | 39,43
features | 100 | 100 | 100 | 100 |
  index.js | 100 | 100 | 100 | 100 |
src | 73.13 | 46.43 | 25 | 75.38 |
  calculateFeatureScore.js | 31.25 | 0 | 0 | 35.71 | 2-9,13,26,54
  calculateScore.js | 90.24 | 65 | 100 | 90.24 | 69,75,81,87
  featureWeights.js | 70 | 100 | 0 | 70 | 4-5,302
src/common | 75.97 | 62.11 | 77.14 | 75.59 |
  actions.js | 75.41 | 67.35 | 71.43 | 75 | 27,46,48,56,64,72,162,190-192,202-230
  classifyUser.js | 84.38 | 55.56 | 80 | 83.87 | 57,99-109
  customCalculations.js | 50 | 45.45 | 100 | 50 | 5,21-24,32-38
  filterFeatures.js | 77.78 | 75 | 100 | 77.78 | 10,14
  queryGitInfo.js | 100 | 100 | 100 | 100 |
  validateUser.js | 100 | 100 | 100 | 100 |
  weights.js | 100 | 100 | 100 | 100 |
src/gitRepos | 79.8 | 55.1 | 58.82 | 80.41 |
  get-top-repos.js | 69.49 | 44.44 | 58.33 | 68.97 | 27-30,41-51,95,127-159
  repo-info.js | 95 | 61.29 | 60 | 97.44 | 48
src/info | 94.34 | 86.49 | 91.67 | 96.15 |
  repo-info.js | 95.24 | 91.67 | 100 | 95.24 | 48
  top-languages-info.js | 93.75 | 76.92 | 90 | 96.77 | 54
-----

Test Suites: 5 passed, 5 total
Tests: 16 passed, 16 total
Snapshots: 0 total
Time: 11.124 s
Ran all test suites.
```

Figure 3.1:Jest code coverage results

Therefore, we introduced the Jest testing framework to our repositories to maintain the error-free codebase. Also using Jest, we can collect all the code coverage information of the entire project. Figure 3.1 shows the tested test cases and the code coverage information of the Git-Data-Miner repository.

Also, Postman API testing tool has been used to test all the API endpoints, and its collections feature was used to manage all the endpoints in one place. Figure 3.2 shows testing backend API request via the Postman application.

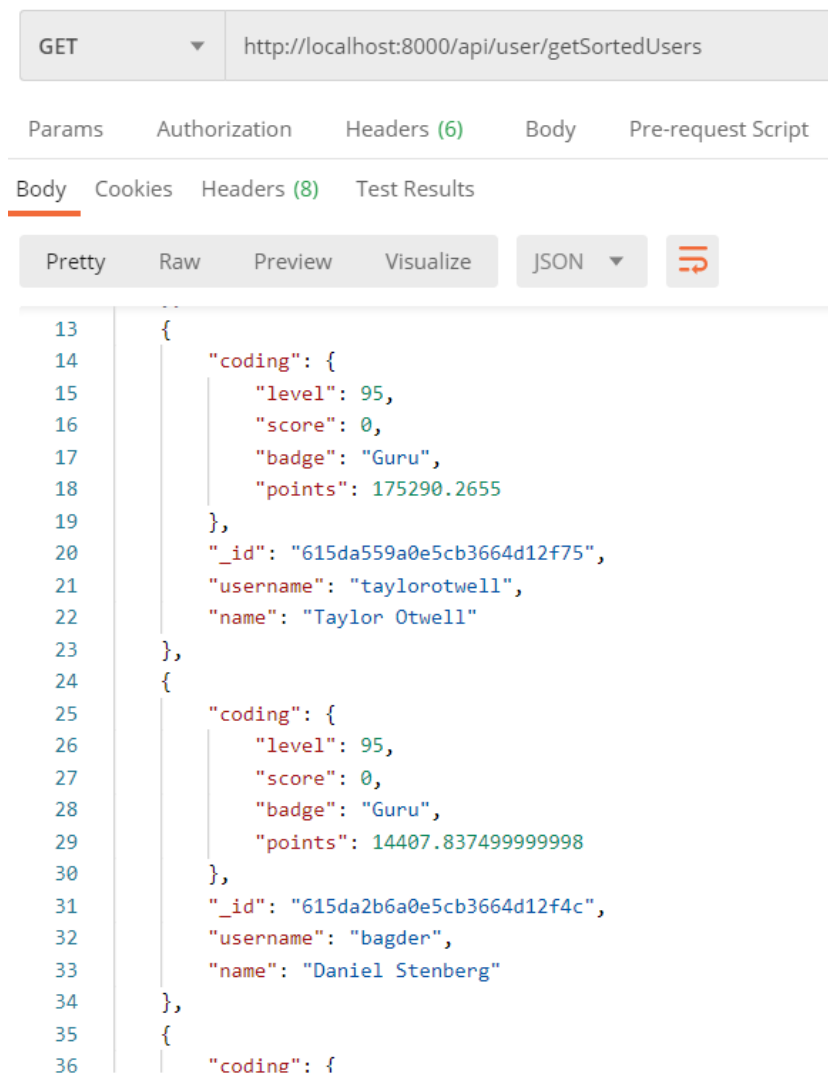


Figure 3.2: Testing API endpoints via Postman

3.2 Implementation

3.2.1 Portfolio Front-end

ProbExpert portfolio front end was developed using Next.js on top of React.js. Material-Ui component library has been used to develop the frontend components and stylings. The main reason to select Next.js to implement the frontend is that it provides server-side rendering (SSR) feature out of the box[27]. This technology has been used to render all the generated portfolio on the server side. This means that once the HTML has been delivered to the client (the user's browser), nothing else needs to happen for the user to be able to read the content on the page. This makes page loading times appear much faster to the user. SSR also make it easier to make the pages indexable and crawlable. Therefore, the web crawler can easily access all the pages for indexing and ranking them on major search engines like Google, Bing, etc. The below figure shows the user interface of an actual portfolio in our application.

3.2.2 Portfolio Back-end

The portfolio backend was developed as a RESTful API service to easily communicate with the ProbExpert frontend. Express.js library was used on top of Node.js to build the backend. Since Express.js provides a highly advanced routing mechanism, It helps to reduce the development time massively. Also, its built-in debugging mechanism helps pinpoint the exact part of the code that contains the bugs.

Since the portfolio user model consists of many data and most of them are not available for all the users, a database with a strict schema is harder to maintain. Therefore, the MongoDB Atlas database has been used as the central database of the portfolio system.

```

_id: ObjectId("615de483d4586252b4c72358")
coding: Object
  level: 95
  score: 0.017514309298222175
  badge: "Guru"
  points: 741.803
> contribute_high_repos: Array
> contribute_mid_repos: Array
> contribute_low_repos: Array
  0: "lerrua/remote-jobs-brazil"
  1: "wesbos/awesome-uses"
  2: "prest/prest"
  3: "ossf/scorecard"
  4: "editor-bootstrap/vim-bootstrap"
  5: "openshift/osin"
  6: "ergochat/ergo"
  7: "mattn/goveralls"
  8: "golang/gofrontend"
> individual_high_repos: Array
> individual_mid_repos: Array
> individual_low_repos: Array
class: "GitHubStar"
> git_langs: Array
  username: "avelino"
  name: "Avelino"
  avatar_url: "https://avatars.githubusercontent.com/u/31996?u=77bedf6b738458f691ee6e..."
  bio: "open source engineer"
  company: "@decodebuzz"
  location: "Kailua-Kona, Hawaii"
  created_at: "2008-10-31T14:06:07Z"
  total_prs: 553
  total_issues: 910
  total_stars: 70111
  total_commits: 11334
  contributed_to: 54
  total_repos: 170
  total_followers: 3223
  is_hireable: true
  profile_age: 12.9
  total_discussions: 44
  answered_discussions: 4
  latest_update: 2021-10-06T18:01:39.934+00:00
__v: 0

```

Figure 3.3: Sample user document on MongoDB

3.2.3 Web Scrappers

Two separate scrappers have been developed to scrape user information from StackOverflow and LinkedIn. BeautifulSoup 4 python library was used as the main technology to them. Since LinkedIn block unauthorized visitors from viewing user profiles, Selenium was used to automate the Login process in LinkedIn. Since the portfolio backend has been built on the Node server, the Node.js child process feature was used to run the python scripts.

3.2.4 GitHub OAuth2 flow

To validate whether the users generate their portfolio using their own GitHub accounts, OAuth2 verification has been applied to the portfolio generation flow. Below **Error! Reference source not found.** illustrate the message flow between ProbExpert and the GitHub OAuth2 server.

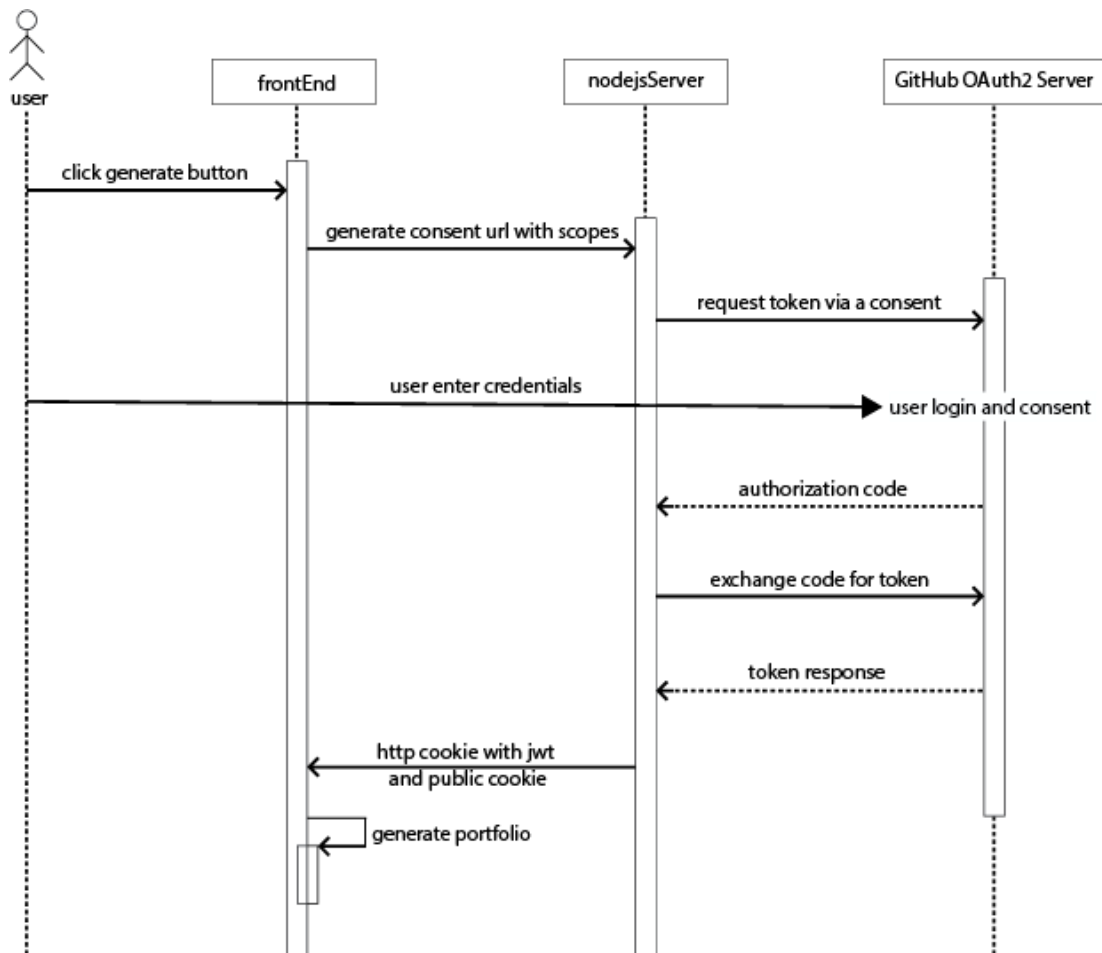


Figure 3.4: OAuth implementation

3.2.5 CV generation

Puppeteer opensource javascript library was used for PDF generation purposes. However, puppeteer allows generating PDF documents both on the client-side and

server-side. To keep the frontend simplified and these kinds of tasks easier to handle from the Node.js, puppeteer implementation was done from the server-side.

3.2.6 Deployment

Since the portfolio frontend was developed using Next.js, it has been deployed on their native Vercel's cloud platform. Vercel is a deployment and collaboration platform for deploy client-side applications and web services. These servers scale automatically according to the traffic, and they have an inbuilt caching system for all the applications. Therefore, Vercel is the optimal solution to host the frontend of the ProbExpert platform.

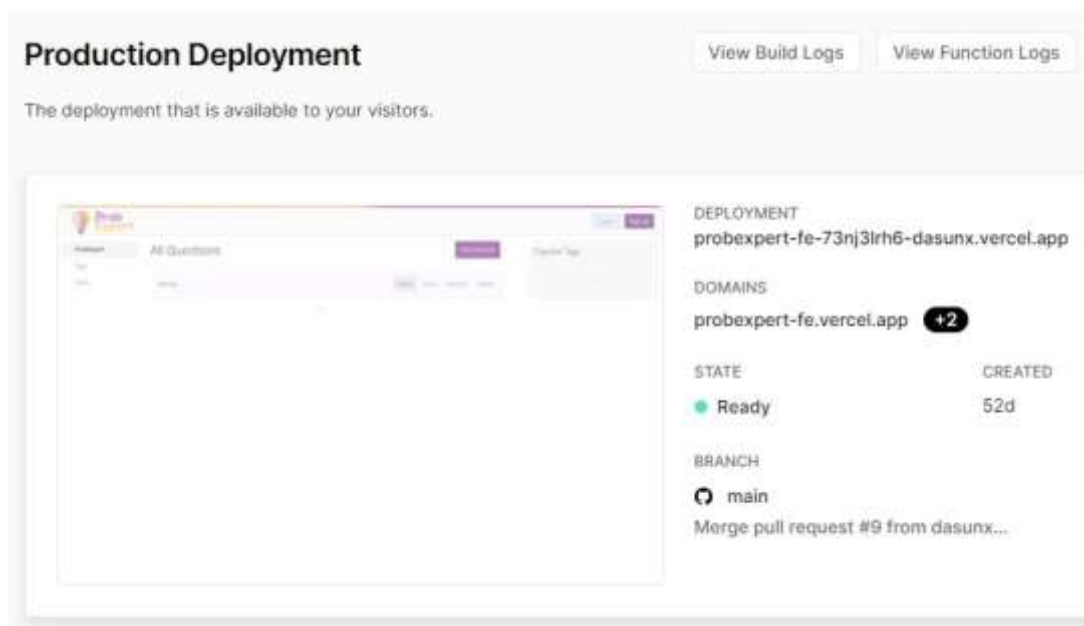


Figure 3.5: Frontend deployment on Vercel

the Git-Data-Miner tool that is used to communicate with GitHub GraphQL API has also been deployed on Vercel. Since all the methods of Git-Data-Miner were made as serverless functions, Vercel's serverless service has been used for the deployment.

ProbExpert's backend has been deployed on Heroku using a free dyno. Since ProbExpert's Node.js server consisted of python scripts and Selenium, three external build packs: python, chrome-driver, and chrome-browser build pack, have been added to the dyno.

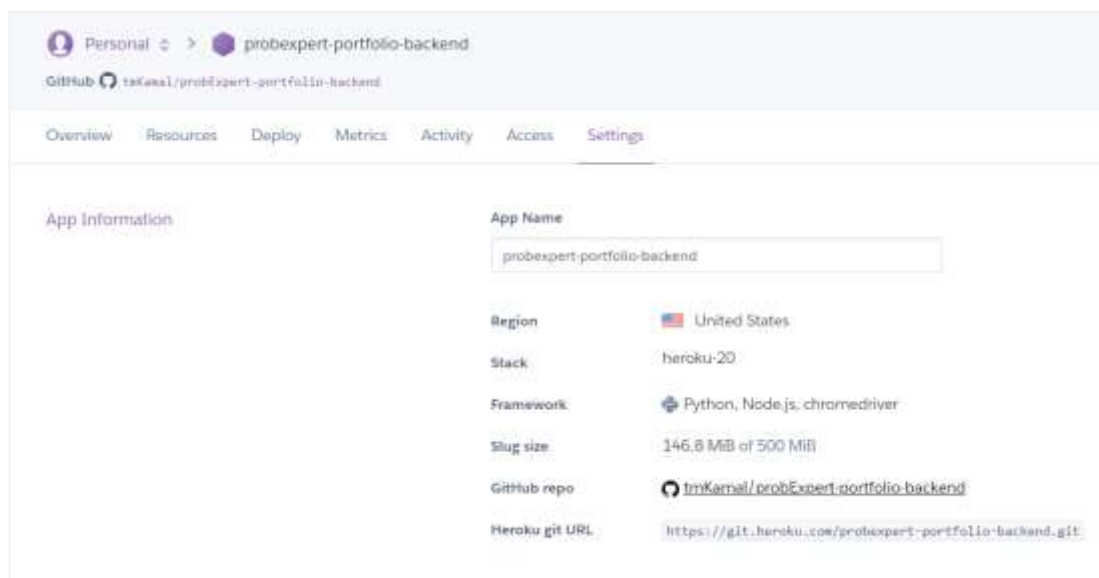


Figure 3.6: Backend deployment on Heroku

4 RESULTS & DISCUSSION

This section discusses the results and findings of the ProbExpert developer ranking algorithm and automated Portfolio generation implementation.

4.1 Results

Using the Git-Data-Miner, a balanced dataset contains 50 GitHub user profiles based on the stratified sampling have been selected and extracted. Since the main purpose is to find out which user activities has the effect of determining the proficiency of a developer, we gathered 18 sets of features as discussed in the 2.2.2. Then after performed the feature analysis, we found that only 8 features have a valid effect for evaluating developer proficiency.

By using the extracted features above discussed, the GitHub scoring model has been developed tested and implemented into the ProbExpert main platform. Figure 2.16, Figure 2.17, and Figure 2.18 illustrate the validity of the developed ranking algorithm by evaluating 3 different level users by classifying them correctly according to their proficiency. Since this scoring model uses the bell curve to classify users, extreme developers such as Linus Torvalds also will get selected into the ‘Guru’ class, regardless of their higher TDS value. Since the machine learning algorithm has some higher false results when predicting the edge cases, this approach is solid and consistent for directly implementing into the platform like ProbExperts. Also, this scoring model calculates the TDS value for each developer; it can be used to rank them on leaderboards without any confusion. Figure 2.21 shows how we use the TDS value to do the ranking on ProbExperts.

Below is the other list of results our portfolio system provide to the software developer community.

Expert Identification: Since all the developers are listed on the leaderboard according to their rankings, hiring managers and potential buyers can find the unemployed or free-lancing developers in no time by filtering the leaderboard by using the available to hire tag.

CV Shortlisting: Recruiters can utilize ProbExpert's portfolio generating section to quickly generate and determine developers' scores and activities. This will save a recruiter a significant amount of time, which they would otherwise spend manually going through a developer's social networks such as LinkedIn, GitHub, and Stack Overflow while evaluating a candidate.

As guidance for newbies: Since developer portfolios include significant developer activities, viewing an expert's portfolio will give novices a solid idea of how they should shape their path accordingly to pursue their dream profession or career goals

4.2 Research Findings

As discussed in the above section 4.1, after the data analysis, we found out that only eight features directly affect when determining a developer proficiency. Below is the list of the selected features after the analysis.

- Profile age
- Total Commit Contribution
- Number of repositories contributed to (Based on Stargazer count)
- Total Pull Request count
- Total Issues count
- Total Followers
- Total Stargazers
- Total approved comments as answered

It is not surprising to find out that the total number of the used programming language doesn't have a effect on evaluating a developer because GitHub Limited the number of years can go deep into individual commits in detail. The reason behind this issue was discussed thoroughly in the section.

Another fascinating fact we found out from this thesis is that GitHub-Star developers have a great impact on the community regarding knowledge sharing. According to the empirical cumulative distribution function of Figure 2.10, Only 10% of GitHub-Star developers has surpassed all other classes when answering the discussions on GitHub.

Also, the 'availability to work' feature doesn't have a possible effect when determining proficiency. Beginner users might not be aware of how to turn this option in Github can be the reason behind this issue, as discussed in the section 2.2.3.

4.3 Discussion

Since StackOverflow and LinkedIn do not provide a public API to collect the developer activities and behaviours unlike GitHub, two web scrappers has been developed. However LinkedIn has a strict set of rules for viewing users profiles as a unregistered user. Therefore Selenium has been used to automate the login process to collect user information as a registered user. However, this implementation also has some flaws due to the IP restriction rules of LinkedIn. It only allows viewing a limited number of user profiles from one IP address at a time. Therefore to make this Linked in web scrapper work in the production mode, It needed to be hosted on a Linux instance with dynamic DNS configured. Amazon EC2 machine will be a perfect solution for this deployment. Since the portfolio and scoring algorithm are mainly based on GitHub GraphQL API, LinkedIn web scrapper could be dropped from the future release of ProbExpert to improve the system stability.

4.4 Summary of the Student Contribution

Table 4.1: Summary of student contribution

Personal	Functionality	Description
Thennakoon T.M.K.H.B	Detecting users' proficiency level along with an auto-generated portfolio	<p>Serverless RESTful API to retrieve data from Github GraphQL service.</p> <p>Web Scrappers to gather necessary developer's statistics</p> <p>Scoring model to generate a score for developers activities and behaviours</p> <p>Ranking algorithm to classify developers into suitable classes</p> <p>Implement Automated portfolio generation functionality</p>

7 CONCLUSION

In this research, we have presented our approach for extracting developers' features, ranking them using the scorer model, and classifying them into related classes based on the Bell curve. As discussed in the methodology, the tool we developed to communicate with the GitHub GraphQL API can be used to collect data for any future research conducted on the GitHub platform. Since this portfolio system developed as a micro service, it can be implemented into any existing platform without needing much changes.

REFERENCES

- [1] C. Hauff and G. Gousios, "Matching GitHub developer profiles to job advertisements," in *IEEE International Working Conference on Mining Software Repositories*, Aug. 2015, vol. 2015-August, pp. 362–366, doi: 10.1109/MSR.2015.41.
- [2] E. Constantinou and G. M. Kapitsaki, "Identifying Developers' Expertise in Social Coding Platforms," in *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*, Oct. 2016, pp. 63–67, doi: 10.1109/SEAA.2016.18.
- [3] C. Hauff and G. Gousios, "Matching GitHub developer profiles to job advertisements," *IEEE Int. Work. Conf. Min. Softw. Repos.*, vol. 2015-Augus, pp. 362–366, 2015, doi: 10.1109/MSR.2015.41.

- [4] J. Yang, K. Tao, A. Bozzon, and G. J. Houben, “Sparrows and owls: Characterisation of expert behaviour in StackOverflow,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8538, pp. 266–277, 2014, doi: 10.1007/978-3-319-08786-3_23.
- [5] A. Dargahi Nobari, M. Neshati, and S. Sotudeh Gharebagh, “Quality-aware skill translation models for expert finding on StackOverflow,” *Inf. Syst.*, vol. 87, Jan. 2020, doi: 10.1016/j.is.2019.07.003.
- [6] A. Capiluppi, A. Serebrenik, and L. Singer, “Assessing technical candidates on the social web,” *IEEE Softw.*, vol. 30, no. 1, pp. 45–51, 2013, doi: 10.1109/MS.2012.169.
- [7] B. Vasilescu, V. Filkov, and A. Serebrenik, “StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge,” in *2013 International Conference on Social Computing*, 2013, pp. 188–195, doi: 10.1109/SocialCom.2013.35.
- [8] G. Zhou, J. Zhao, T. He, and W. Wu, “An empirical study of topic-sensitive probabilistic model for expert finding in question answer communities,” *Knowledge-Based Syst.*, vol. 66, pp. 136–145, 2014, doi: 10.1016/j.knosys.2014.04.032.
- [9] S. Patil and K. Lee, “Detecting experts on Quora: by their activity, quality of answers, linguistic characteristics and temporal behaviors,” *Soc. Netw. Anal. Min.*, vol. 6, no. 1, pp. 1–11, 2016, doi: 10.1007/s13278-015-0313-x.
- [10] S. Baltes and S. Diehl, “Towards a theory of software development expertise,” in *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Oct. 2018, pp. 187–200, doi: 10.1145/3236024.3236061.
- [11] Association for Computing Machinery. and SIGAPP., *Proceedings of the 25th*

annual ACM symposium on applied computing 2010 : Symposium on Applied Computing : Sierre, Switzerland, March 22-26, 2010. ACM Press, 2010.

- [12] A. Borodin, G. O. Roberts, J. S. Rosenthal, A. Borodin, and J. S. Rosenthal, “Link Analysis Ranking: Algorithms, Theory, and Experiments,” 2005.
- [13] J. Kleinberg and P. K. Pranav, “Analysis of Link Structure Authoritative Sources in a Hyperlinked Environment,” 1999.
- [14] “The PageRank Citation Ranking: Bringing Order to the Web,” 1998. [Online]. Available: www.yahoo.comm.
- [15] P. Jurczyk and E. Agichtein, “Discovering authorities in question answer communities by using link analysis,” in *International Conference on Information and Knowledge Management, Proceedings*, 2007, pp. 919–922, doi: 10.1145/1321440.1321575.
- [16] J. Zhang, M. S. Ackerman, and L. Adamic, “Expertise Networks in Online Communities: Structure and Algorithms.”
- [17] J. Quemada and ACM Digital Library., *Proceedings of the 18th international conference on World wide web.* ACM, 2009.
- [18] Y. Li, ACM Digital Library., Association for Computing Machinery. Special Interest Group on Knowledge Discovery & Data Mining., and Association for Computing Machinery. Special Interest Group on Management of Data., *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2008.
- [19] W. C. Kao, D. R. Liu, and S. W. Wang, “Expert finding in question-answering websites: A novel hybrid approach,” in *Proceedings of the ACM Symposium on Applied Computing*, 2010, pp. 867–871, doi: 10.1145/1774088.1774266.
- [20] E. Constantinou and G. M. Kapitsaki, “A study of the characteristics of developers’ activities in GitHub,” in *Proceedings - Asia-Pacific Software*

Engineering Conference, APSEC, 2013, vol. 2, pp. 7–12, doi: 10.1109/APSEC.2013.104.

- [21] J. E. Montandon, L. L. Silva, and M. T. Valente, “Identifying Experts in Software Libraries and Frameworks among GitHub Users Software Developers Expertise View project Software Defects View project Identifying Experts in Software Libraries and Frameworks among GitHub Users.” [Online]. Available: <https://stackoverflow.com/jobs>.
- [22] K. M. Perrone, G. Zanardelli, E. Worthington, and J. Chartrand, “Role Model Influence on the Career Decidedness of College Students,” *undefined*, 2002.
- [23] W. Huang, W. Mo, B. Shen, Y. Yang, and N. Li, “CPDScore: Modeling and Evaluating Developer Programming Ability across Software Communities.” [Online]. Available: <http://stackoverflow.com/>.
- [24] W.-Y. Ma, ACM Digital Library., and Association for Computing Machinery. Special Interest Group on Information Retrieval., *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 2011.
- [25] “The Bell Curve: Intelligence and Class Structure in American Life - Richard J. Herrnstein, Charles Murray - Google Books.” [https://books.google.lk/books?hl=en&lr=&id=s4CKqxi6yWIC&oi=fnd&pg=PR11&dq=inverse+bell+curve&ots=gcy0h2pfB6&sig=fXJ_uEOIrIKgYbzT3uV_ofy9lOc&redir_esc=y#v=onepage&q=inverse bell curve&f=false](https://books.google.lk/books?hl=en&lr=&id=s4CKqxi6yWIC&oi=fnd&pg=PR11&dq=inverse+bell+curve&ots=gcy0h2pfB6&sig=fXJ_uEOIrIKgYbzT3uV_ofy9lOc&redir_esc=y#v=onepage&q=inverse%20bell%20curve&f=false) (accessed Sep. 14, 2021).
- [26] A. Soranzo and E. Epure, “Simply Explicitly Invertible Approximations to 4 Decimals of Error Function and Normal Cumulative Distribution Function,” no. 2, pp. 3–5, Jan. 2012, Accessed: Sep. 14, 2021. [Online]. Available: <https://arxiv.org/abs/1201.1320v1>.
- [27] L. Hussein and S. Jubell, “Hur rendering påverkar prestanda

och sökmotoroptimering: – NextJS vs React,” 2021, Accessed: Sep. 14, 2021.
[Online]. Available: <http://urn.kb.se/resolve?urn=urn:nbn:se:hv:diva-17387>.