



**islington college**  
(इस्लिङ्टन कलेज)

**Module Code & Module Title**  
**CS6PO5NI Final Year Project**

**Assessment Weightage & Type**  
**40% Final Project Report**

**Year and Semester**

**2022 Spring**

**Project Title: Sentimento (Social media assisting platform with sentiment analysis)**

**Student Name: Prashant Ghimire**

**London Met ID: 19031371**

**College ID: np01cp4a190008**

**Internal Supervisor: Bibek Khanal**

**External Supervisor: Subash Basnet**

**Assignment Due Date: 2022 April 27**

**Assignment Submission Date: 2022 April 26**

**Word Count: 8800 approx.**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.*

## **Acknowledgement**

I would like to take this occasion to express my gratitude to Mr. Subash Basnet and Mr. Bibek Khanal, our esteemed Project Supervisors, for their unwavering support and advice throughout our final year project. Their continuous supervision and monitoring have been really beneficial in completing this project.

I would also like to express my gratitude to our university and college for allowing us to learn and work on a self-project. I am grateful to my helping teachers, faculty members, and the entire Islington community for allowing us to grow with knowledge and wisdom.

## Abstract

The following report is a compilation of final year project work that tracks the project's overall progress. The introduction, background information, development progress documents, project outcome, and future possible engagement are all included in this report. The research culminates in a fully functional mobile application that performs sentiment analysis on YouTube comments, tweets, and custom texts. Spam and sarcasm detection are also included for a more reliable analysis. Being a social media assisting platform, there is also a feature for connecting job seekers with providers. This project employs a Multinomial Naive Bayes Probabilistic Classifier. Flutter and Flask are utilized to create a user-friendly application. The project's backend API is hosted on the Heroku platform.

**Keywords:** Sentimento, Social Media Sentiment Analysis, Multinomial Naive Bayes, Flutter, Flask

## Table of Contents

Chapter 1: Introduction .....	1
1.1 Project Description .....	1
1.2 Problem Domain .....	2
1.3 Current Scenario .....	3
1.4 Project as a solution.....	4
1.5 Aims and Objectives .....	5
1.6 Structure of the report .....	6
1.6.1 Background .....	6
1.6.2 Development .....	6
1.6.3 Testing and Analysis .....	7
1.6.4 Conclusion.....	7
Chapter 2: Background .....	8
2.1 About end/target users.....	8
2.2 Understanding solution .....	10
2.2.1 Sentiment Analysis Part.....	11
2.2.2 Backend API Development.....	15
2.2.3 Frontend Mobile Application Development .....	15
2.3 Similar projects.....	16
2.4 Comparison and uniqueness in my project .....	19
Chapter 3: Development .....	20
3.1 Considered Methodologies.....	21
3.2 Selected Methodology.....	23
3.3 Phases of methodology.....	25

3.4 Survey Results .....	28
3.4.1 Pre-survey results.....	29
3.4.2 Post-survey results .....	35
3.5 Requirement Analysis .....	41
3.6 Design and Logical Diagrams .....	44
3.6.1 Entity Relationship Diagram .....	44
3.6.2 System Architecture .....	45
3.6.3 Overall Use Case Diagram .....	46
3.6.4 Overall Context Level Diagram.....	47
3.6.5 Flowchart of sentiment analysis module .....	48
3.6.6 Communication diagrams.....	49
3.6.7 Sequence Diagrams .....	51
3.6.8 Structure chart for deleting vacancy .....	54
3.6.9 Activity Diagram to view saved sentiment reports .....	55
3.6.10 Data flow diagrams.....	56
3.7 Implementation following Scrum .....	58
3.7.1 Sprint 1: 1 <sup>st</sup> and 2 <sup>nd</sup> Increment.....	58
3.7.2 Sprint 2: 3 <sup>rd</sup> , 4 <sup>th</sup> , 5 <sup>th</sup> Increment .....	60
3.7.3 Sprint 3: 6 <sup>th</sup> and 7 <sup>th</sup> Increment .....	64
3.7.4 Sprint 4: 8 <sup>th</sup> , 9 <sup>th</sup> and 10 <sup>th</sup> Increment .....	68
3.7.5 Sprint 5: 11 <sup>th</sup> , 12 <sup>th</sup> and 13 <sup>th</sup> increment.....	70
3.7.6 Sprint 6: 14 <sup>th</sup> and 15 <sup>th</sup> increment.....	72
Chapter 4: Testing and Analysis .....	74
4.1 Test Plan .....	74
4.1.1 Unit testing plan.....	74

4.1.2	System testing plan .....	77
4.2	Unit Testing.....	78
4.3	System Testing .....	105
4.4	Critical Analysis.....	110
Chapter 5:	Conclusion .....	111
5.1	Legal, Social and Ethical Issues.....	111
5.2	Advantages .....	112
5.3	Limitations .....	114
5.4	Personal experience and Future work.....	115
5.4.1	Self-reflection or personal experience: .....	115
5.4.2	Here's how I will be further engaged on this project in future: .....	115
Chapter 6:	References .....	116
Chapter 7:	Bibliography.....	119
Chapter 8:	Appendix .....	123
Appendix 1:	Additional Diagrams/Figures .....	123
i.	Gantt chart .....	123
ii.	Work breakdown structure .....	124
iii.	Milestone chart.....	125
iv.	Layers in the designed sentiment analysis algorithm.....	126
v.	Product Backlog.....	127
vi.	Sprint Backlogs.....	128
Appendix 2:	Reasons to use consumed libraries/framework .....	134
i.	Using BLOC for state management .....	134
ii.	Using Flask over fully featured framework like Django.....	134
iii.	Multinomial Naive Bayes algorithm over available package like TextBlob ..	134

Appendix 3: Software Requirement Specification .....	135
Appendix 4: User Feedbacks.....	144
i. Google Play Store Reviews.....	144
ii. User Feedback Survey Report.....	145
Appendix 5: Weekly/Increment task information for development.....	148
Appendix 6: Essential code snippets .....	150
i. Backend.....	150
ii. Frontend.....	157

## List of Figures

Figure 1: Snapshots of built mobile application .....	1
Figure 2 Use case scenario of sentiment analysis (Liu, 2016) .....	3
Figure 3 Type of potential users of the developed platform.....	8
Figure 4 Built solution work breakdown.....	10
Figure 5 General approach of sentiment analysis (Binita Verma, 2018) .....	11
Figure 6 Various methods of sentiment analysis (Walaa Medhat, 2014) .....	12
Figure 7 Bayes rule with labelling (Jha, 2017).....	14
Figure 8 Sentiment analysis outcome between Taylor Swift and Jake Paul.....	16
Figure 9 Analysis of tweets on NEPSE .....	17
Figure 10 Searching social media related jobs in LinkedIn .....	18
Figure 11 Scrum methodology (scrum.org, 2020) .....	23
Figure 12 Sample of daily scrum board.....	26
Figure 13 Social media user's familiarity with sentiment analysis .....	29
Figure 14 User's awareness on precise social media activity.....	30
Figure 15 Chart showing user's interest to comment .....	31
Figure 16 Chart showing people's interest to tweet.....	32
Figure 17 Chart showing user's interest to use sentiment tool .....	33
Figure 18 Chart showing people's interest looking for LinkedIn alternatives .....	34
Figure 19 Chart showing social platform demand for analysis .....	35
Figure 20 Chart showing user's interest to use their own third-party credentials.....	36
Figure 21 Chart showing user's login preferences .....	37
Figure 22 Chart showing what sentiment labels user want .....	38
Figure 23 Chart showing user's interest on report saving feature .....	39
Figure 24 Chart showing user's opinion on limiting vacancy post at a time.....	40
Figure 25 ERD of Sentimento's database .....	44
Figure 26 System architecture of Sentimento platform.....	45
Figure 27 Use Case Diagram of overall system .....	46
Figure 28 Context level diagram .....	47
Figure 29 Flowchart of sentiment analysis module .....	48

Figure 30 Communication diagram for performing sentiment analysis.....	49
Figure 31 Communication diagram for posting vacancy.....	50
Figure 32 Sequence diagram for logging in user.....	51
Figure 33 Sequence diagram for tweet sentiment analysis .....	52
Figure 34 Sequence diagram for viewing freelancers .....	53
Figure 35 Structure chart for deleting vacancy .....	54
Figure 36 Activity diagram to view saved reports .....	55
Figure 37 Level 0 DFD for login/registration.....	56
Figure 38 Level 1 DFD for login/registration.....	56
Figure 39 Level 2 DFD for login/registration.....	57
Figure 40 Initial user authentication interface.....	58
Figure 41 Change in meaning of a sentence while removing stop words.....	59
Figure 42 Function to clean sentence .....	60
Figure 43 Function to make updates in serialized training model.....	61
Figure 44 Function to predict probability of each sentence to fall in 0 or 1 category .....	62
Figure 45 Single class to access all algorithmic feature of system.....	63
Figure 46 Function to extract tweets of chosen topic .....	64
Figure 47 Function to extract YouTube comments of chosen video.....	65
Figure 48 Function to handle login request .....	66
Figure 49 User login request task .....	67
Figure 50 API handling sentiment analysis request on chosen YouTube video .....	68
Figure 51 Viewing user's saved sentiment reports .....	69
Figure 52 Git commit showing front end user authentication pages task done .....	69
Figure 53 Bloc state management architecture (Aqeel, 2020) .....	70
Figure 54 Git commit showing tasks of fifth sprint .....	71
Figure 55 API hosted in Heroku platform .....	72
Figure 56 Application published to Google Play Store .....	73
Figure 57 Trying to register with already used email.....	78
Figure 58 Trying to login without being registered.....	79
Figure 59 Registering user in the platform .....	80
Figure 60 Logging inside the platform .....	81

Figure 61 Trying to log in with incorrect password .....	82
Figure 62 Showing profile icon to enter profile page .....	83
Figure 63 User navigated to profile screen after loading completes.....	84
Figure 64 Trying to perform sentiment analysis without providing tokens .....	85
Figure 65 Entering coupons and video information .....	86
Figure 66 Showing sentiment report after loading completes.....	87
Figure 67 Polarity check of manual sentence.....	88
Figure 68 Providing invalid coupon .....	89
Figure 69 Trying to perform analysis using invalid coupon.....	89
Figure 70 Toast message about invalid request.....	89
Figure 71 Saved sentiment reports of user .....	90
Figure 72 Trying to view reports without performing any analysis.....	91
Figure 73 Dashboard in portrait view .....	92
Figure 74 Dashboard in landscape view .....	93
Figure 75 Posting vacancy in the platform .....	94
Figure 76 User having five open vacancies trying to add another new vacancy .....	95
Figure 77 User navigated to phone call window .....	96
Figure 78 Deleting previously added post .....	97
Figure 79 Logging out from the platform .....	98
Figure 80 Turning off Wi-Fi and Mobile data .....	99
Figure 81 Trying to view reports without internet connection .....	99
Figure 82 Dashboard after logging in .....	100
Figure 83 Dashboard after opening for second time .....	100
Figure 84 Trying to access profile without providing authentication header .....	101
Figure 85 Viewing vacancies information from web browser.....	102
Figure 86 Trying to delete another user's vacancy.....	103
Figure 87 Trying to perform sentiment analysis on more than 1000 data.....	104
Figure 88 Checking hosted backend API on Heroku.....	105
Figure 89 Count before performing analysis .....	106
Figure 90 Performing polarity check.....	106
Figure 91 Count after performing analysis .....	106

Figure 92 Performing sentiment analysis on tweets .....	107
Figure 93 Viewing recently saved report .....	108
Figure 94 Flutter Application Compilation.....	109
Figure 95 Distinct use cases of sentiment analysis (Samadhan Engineering, 2021) ..	112
Figure 96 Snapshot of a LinkedIn learning course for Flask .....	116
Figure 97 Snapshot of Flask documentation web page .....	117
Figure 98 Snapshot of Flutter package manager .....	117
Figure 99 Snapshot of scikit-learn library .....	118
Figure 100 Gantt Chart of the project work division .....	123
Figure 101 Work breakdown structure of overall project .....	124
Figure 102 Milestone chart of overall project.....	125
Figure 103 Layers showing how sentiment analysis works .....	126
Figure 104 Product backlog design .....	127
Figure 105 Sprint 1 Backlog .....	128
Figure 106 Sprint 2 Backlog .....	129
Figure 107 Sprint 3 Backlog .....	130
Figure 108 Sprint 4 Backlog .....	131
Figure 109 Sprint 5 Backlog .....	132
Figure 110 Sprint 6 Backlog .....	133
Figure 111 User feedback from Google Play Store reviews.....	144

## List of Tables

Table 1 Comparison of other projects and sentimento .....	19
Table 2 General principles of SDLC (Prokopisko, 2019) .....	20
Table 3 Survey volunteer's information .....	28
Table 4 Functional requirements .....	42
Table 5 Unit testing for Flutter application .....	75
Table 6 Unit testing for Flask API .....	76
Table 7 System testing for built platform .....	77
Table 8 Unit test 1 .....	78
Table 9 Unit test 2 .....	79
Table 10 Unit test 3 .....	80
Table 11 Unit test 4 .....	81
Table 12 Unit test 5 .....	82
Table 13 Unit test 6 .....	83
Table 14 Unit test 7 .....	85
Table 15 Unit test 8 .....	86
Table 16 Unit test 9 .....	88
Table 17 Unit test 10 .....	89
Table 18 Unit test 11 .....	90
Table 19 Unit test 13 .....	92
Table 20 Unit test 14 .....	94
Table 21 Unit test 15 .....	95
Table 22 Unit test 16 .....	96
Table 23 Unit test 17 .....	97
Table 24 Unit test 18 .....	98
Table 25 Unit test 19 .....	99
Table 26 Unit test 20 .....	100
Table 27 Unit test 21 .....	101
Table 28 Unit test 22 .....	102
Table 29 Unit test 23 .....	103

Table 30 Unit test 24 .....	104
Table 31 System test 1.....	105
Table 32 System test 2.....	106
Table 33 System test 3.....	107
Table 34 System test 4.....	108
Table 35 System test 5.....	109

## **Table of Abbreviations**

- NLTK: Natural Language Toolkit
- API: Application Programming Interface
- BLOC: Business Logic Component
- CRUD: Create Read Update Delete
- ML: Machine Learning
- AI: Artificial Intelligence
- SDLC: Software Development Life Cycle
- FR: Functional Requirement
- SR: System Requirement
- ERD: Entity Relationship Diagram
- DFD: Data Flow Diagram
- HTTP: Hypertext Transfer Protocol
- Git: Global Information Tracker
- UI: User Interface
- UT: Unit Testing
- ST: System Testing
- SRS: Software Requirement Specification
- Sentimento: Name of the application/platform of this project

## Chapter 1: Introduction

### 1.1 Project Description

Sentimento, a social media assisting platform with sentiment analysis feature is a mobile application built using Flutter for front end, Flask for back end and Multinomial Naive Bayes Theorem for the algorithmic part.

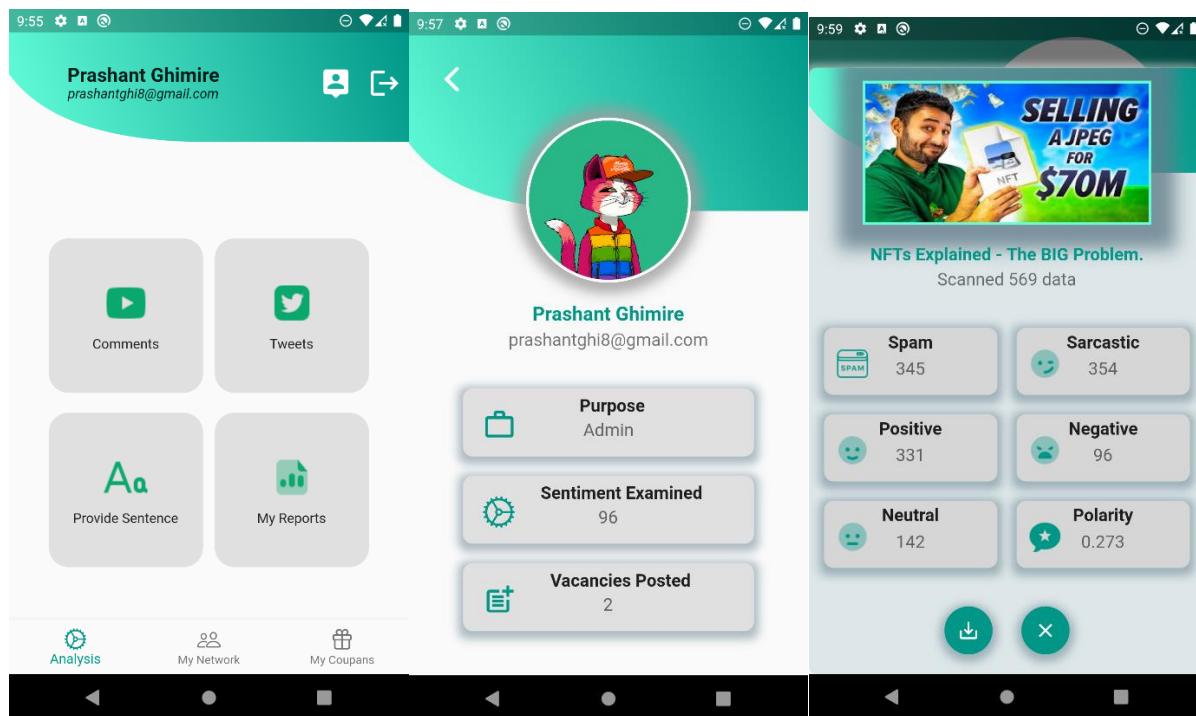


Figure 1: Snapshots of built mobile application

This application is built for folks who need social media in their professional lives in some way. Users can perform sentiment analysis on chosen YouTube video comments and tweets using the developed platform. This platform allows social media-related job recruiters and job seekers to connect with one another.

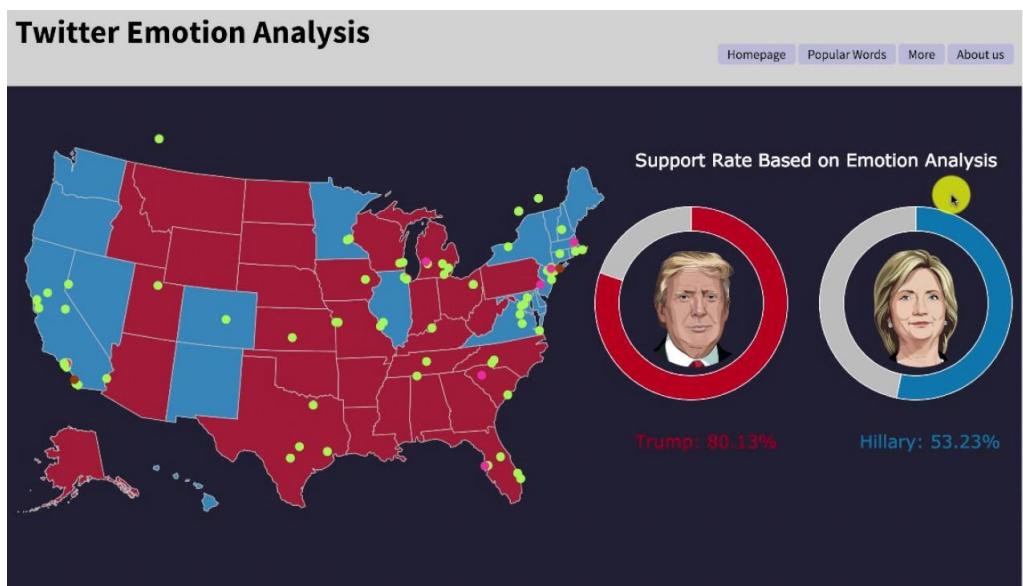
## 1.2 Problem Domain

The comments, feedback and public opinion on social medias can be really applicable for various purposes to companies, content creators and for similar professions. User's opinion data is available on the internet but there is lack of optimal utilization of it. The prime difficulties I found in the social media engaged professionals are:

- i. The feedbacks of any range of customers can be truly important to sharpen the qualities to a company, content creators and similar professionals but the lack of proper tools, the power of feedbacks cannot be optimally utilized (John Hattie, 2007).
- ii. The Online Labour Index (OLI) provides data related to gig economy (freelancers). In 2017, the index reported that skills from diverse ranges were growing from all around the world where USA was ahead for writing/translation and India was ahead for software development and technology. Even with the availability of multiple job searching platforms, the freelancers engaged in social media works were found to have difficulties in connecting with their specific type of jobs (Rajat Kathuria, 2017).

### 1.3 Current Scenario

In the early days of telecommunication, digital communication was a means of luxury to the very specific numbers of people. Coming to the present days, Social Medias have established themselves as a necessity in every day human's life. The social media penetration took place instantaneously within a generation period. With the advancement of social communication technology, concepts such as mass sentiment analysis, targeted advertising, digital economy concepts like Metaverse are also growing in a consistent and well-grounded pace.



*Figure 2 Use case scenario of sentiment analysis (Liu, 2016)*

Contemplating the potential use cases of strong human presence in social medias and my genuine interest in this, I was self-motivated to build my final year project and future career in this sector. Sentimento is a portable platform or mobile application for professionals who are engaged in any form of social media work. It is built primarily for social media influencers and digital content creators who has presence in the social media in a professional life. The professionals engaged in digital marketing/advertising, mass psychology, political campaigns and similar field professionals can be benefited from my built platform.

#### 1.4 Project as a solution

Sentimento, the platform I have developed is a tool for professionals who are engaged in multiple forms of social media works like content creators, freelancers, marketing personals, etc. The solutions that I have come up with to the previously stated challenges are as follows:

- i. To keep track of audience feedbacks, content providers can't look through every single comment on their YouTube videos. This takes a long time and is frequently unfeasible. The Sentimento platform may be used to scan comments in bulk and content creators can be provided with a summary feedback report.
- ii. Not every company or workforce affords expensive cost of data science team, the platform can be utilized by marketing and advertisement agencies to find what people are saying to particular things. At this stage of prototype level, YouTube comments and Twitter tweets are available as means of data.
- iii. Freelancers related to social media work can be connected with relevant work vacancies in the platform. I considered this feature to include to solve the second problem I mentioned in previous heading.
- iv. Political Campaigns or similar events has their own objectives. To monitor what people are saying about the particular event can be monitored from the platform using tweets sentiment analysis to specific hashtag/topic.

### 1.5 Aims and Objectives

Developing a prototype level platform for social media professionals where sentiment analysis can be performed and connecting freelancers with relevant work vacancies is the main aim of this project.

The activities of my involvement or the objectives to fulfil the aim of the project are as follows:

- i. To learn about Sentiment Analysis in Natural Language Processing which will be helpful for me to study Data Science after Bachelors level.
- ii. To learn Flutter, cross-platform application development framework which is also the interest of mine to engage in application development field.
- iii. To learn about backend and REST API development using Python's Flask.
- iv. To research and self-study about how can I implement the sentiment analysis part in my project.
- v. To perform additional research on documentation part for authentic facts/data and tackle difficulties I may face on development part.
- vi. To implement user interfaces and front-end development works on Flutter that I learned from my internship period.
- vii. To carry out database related works in real project that we have been learning from first year of the college.

## 1.6 Structure of the report

In this section, the chapters of the report are shortly introduced.

### 1.6.1 Background

This section of the documentation is devoted to a literature review. It includes information about other similar projects, as well as details on the differences and improvements made to the project. The mobile application is created to meet the needs of the possible consumers of the generated application/solution. This section of the report goes over the algorithmic part of the sentiment analysis feature utilizing Multinomial Nave Bayes in detail. The backend and frontend development work structures are also documented to ensure that the whole system is properly understood.

### 1.6.2 Development

This section of the report documents the technical work performed during the development of this project, including system design, database and system architecture, frontend and backend development, software development methodology. This section of the report also includes a survey done at various stages of project development, as well as diagrammatic representations of various components of the solution. The codes for various sections of the development, such as the algorithmic component, the backend/frontend side, and the cloud hosting work, are also attached here with sufficient elaboration.

### 1.6.3 Testing and Analysis

This section of the project documentation offers a summary of the testing procedures for the created system's various functionalities. The testing methods or plans for completing system and unit testing are first thoroughly outlined. Under system testing, each of the developed system's essential features is tested. Under unit testing, the smaller logical functions or the interface with which the user interacts are also tested. The testing procedure includes tabular data and screenshots of the interface depending on the expected and actual/output results.

Following the testing, there is a critical analysis portion in which the overall testing, whether the development went as intended, and other components of the development work are critically examined, as well as personal experience.

### 1.6.4 Conclusion

The report's final but crucial section offers information on the project's legal, social, and ethical difficulties. This section also includes the project's various use cases/benefits as well as its limitations. This section of the documentation work also discusses additional work that can be done to improve the project, how further studies in this topic can be done in further studies of my academic and career pathway.

## Chapter 2: Background

### 2.1 About end/target users

Sentimento, a social media assisting platform, can be used by various social media related professionals. Primarily, we can categorize the end users based in following tree structure:

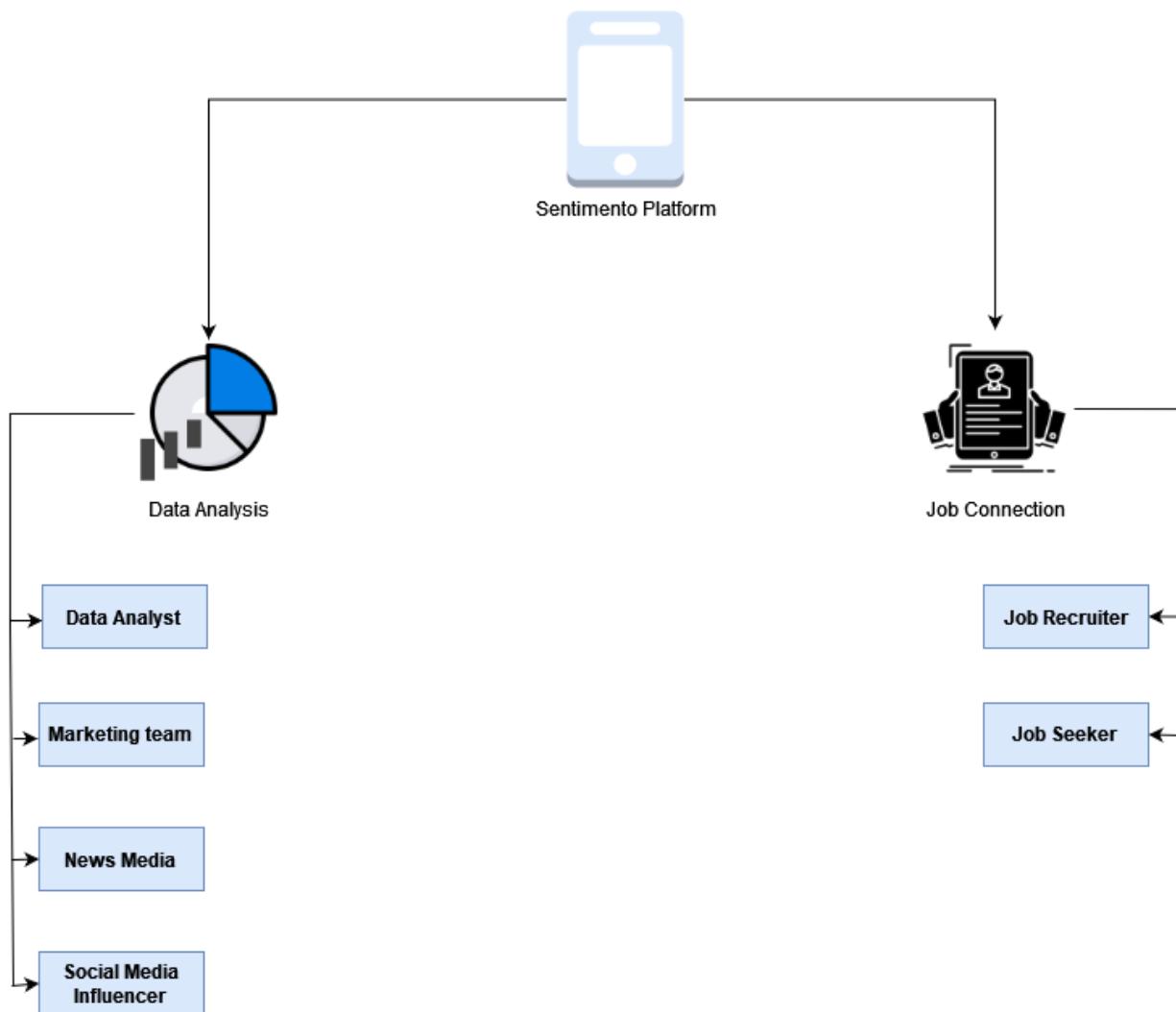


Figure 3 Type of potential users of the developed platform

**Specific client of sentimento platform**

The platform is designed and developed to fit all the previously mentioned potential users. It clarifies that this project is not made for any particular client. The platform is developed keeping in mind to fulfil the professional needs of potential users. The user interface and database architecture are structured for all the general users of the platform.

Because this is a prototype mobile application, no specific clients or corporations have been approached to create a toolkit for a specific company. However, with a few tweaks, the sentiment analysis tool may be tailored to meet the specific and detailed demands of any given client.

## 2.2 Understanding solution

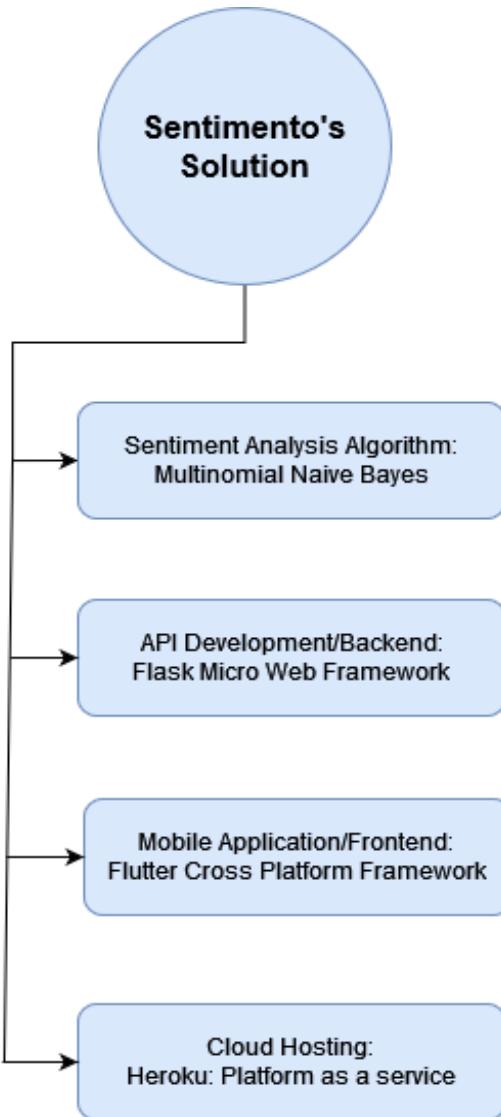


Figure 4 Built solution work breakdown

The project was finished with the help of a variety of tools and technologies that were already available in the community. The total solution is broken into numerous sections in this part, and each module is well explained.

## 2.2.1 Sentiment Analysis Part

### Generic Introduction to Sentiment Analysis

Sentiment Analysis in the context of computer science is the phenomenon of extracting human sentiment from the provided texts. It is the concept of mining human opinion regarding particular topic, event, individual/party, issues or similar aspects. Sentiment/Opinion Analysis is a part of Natural Language Processing. The outcome of a sentiment analysis project is a system which helps to understand the human sentiments like positive, negative and neutral expressions regarding any topic (Ludvig Persson, 2018).

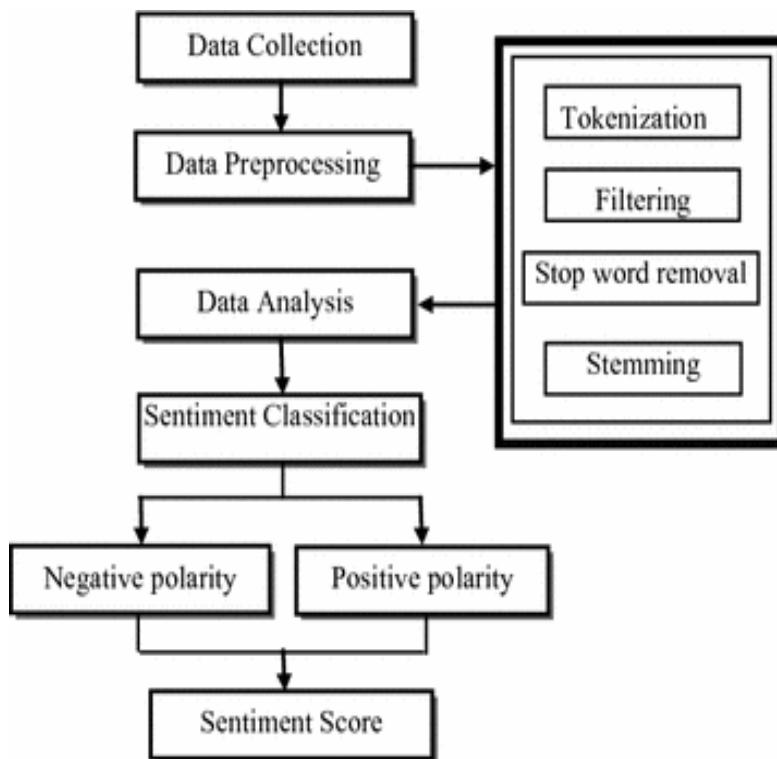


Figure 5 General approach of sentiment analysis (Binita Verma, 2018)

The above diagram depicts a graphical summary of the sentiment analysis workflow. The flow of the process is depicted in the diagram above, from the beginning phase of gathering data to the processing phases and finally to the finalization of sentiment scores.

### Distinct approaches of performing sentiment analysis:

There are multiple approaches of performing sentiment analysis. Depending on the use cases, either machine learning approaches or lexicon-based approaches can be implemented. More details on these distinct approaches are described below.

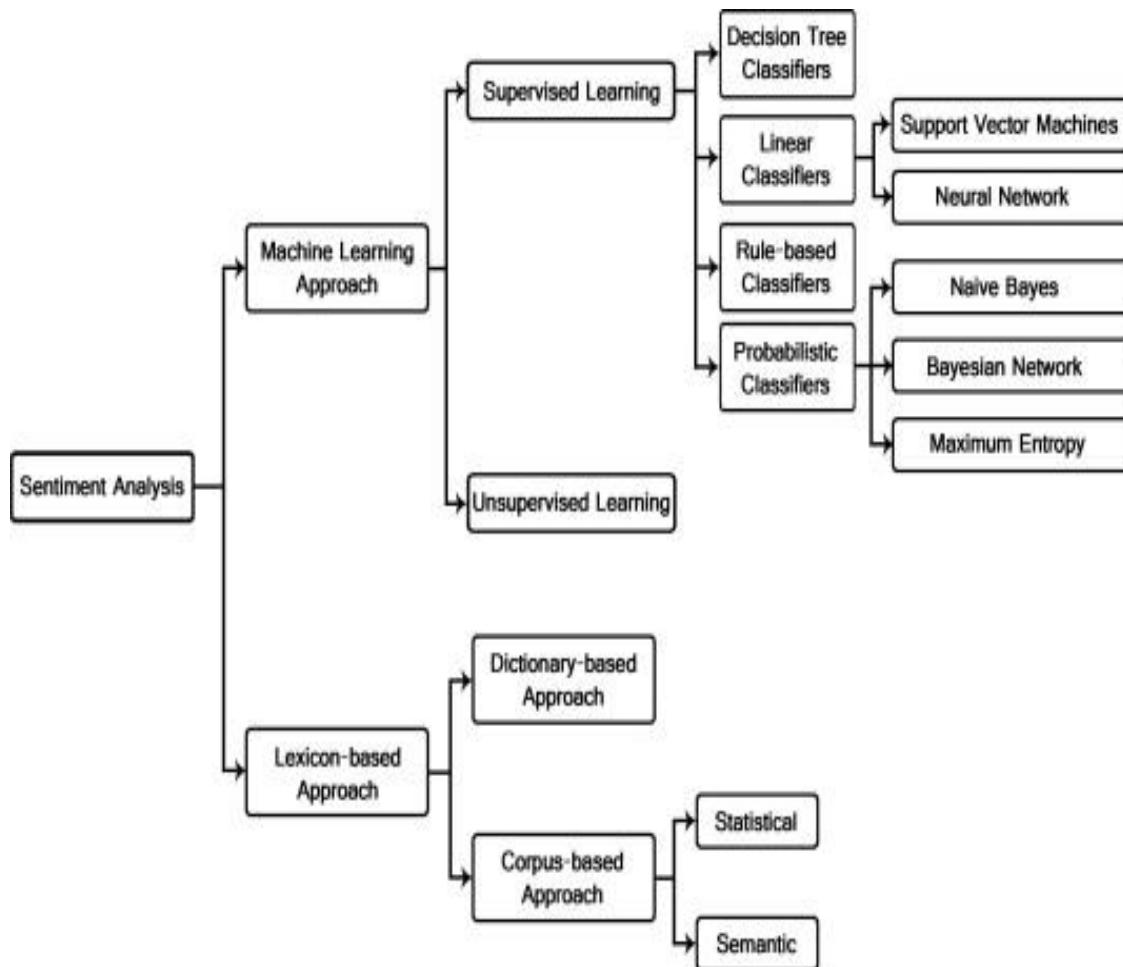


Figure 6 Various methods of sentiment analysis (Walaa Medhat, 2014)

### Lexicon-based Approach

In this approach, a collection or dictionary of positive and negative labelled words are required. For calculating the overall sentiment/polarity of the data, a function to sum the average polarity of words is defined. The final average polarity is determined as the overall sentiment (positive or negative) score for a sentence or a piece of testing data (Mulvenna, 2015).

### Machine Learning Approach

Supervised and Unsupervised learning are two categories in machine learning approach. In unsupervised learning approach, a model or machine is trained consuming the unclassified/unlabelled information. The developed algorithm is solely responsible to act on those unclassified data without guidance. The model classifies the information based on the observed similarities, differences and patterns from the provided information. Clustering and Association are two categories of unsupervised learning. Clustering is about finding patterns in collection of unlabelled data. For finding the relation or link among data in large unclassified data, association techniques are used (Shimon Ullman, 2014).

In supervised learning, there are multiple classification techniques like Decision Tree Classifier, Linear Classifier, Rule Based Approach, Probabilistic Classifier. Each of these classification approaches have their own certain variations and multiple algorithms. Decision Tree Classifier can be used for both classification and regression problems. It is a tree-structured classifier where there are decision node and leaf node. Decision nodes are used to perform decision, have multiple branches whereas leaf nodes are output of those decisions (Lior Rokach, 2015).

Naive Bayes is popular probabilistic classification technique derived from Bayes Theorem. This theorem is a mathematical formula used for calculating conditional probabilities. Naive Bayes Classifier is easy and simple to implement. It can work with less amount of training data too. It is fast and can be used for real-time predictions (Stanford Edu, 2009).

$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$

Likelihood                      Class Prior Probability  
 $P(c | x) = \frac{P(x | c)P(c)}{P(x)}$   
 ↓                                  ↓  
 Posterior Probability           Predictor Prior Probability

Figure 7 Bayes rule with labelling (Jha, 2017)

#### Elaborating the above rule with an example

Let's assume 'c' as shopping and 'x' as raining. The posterior probability denotes the conditional probability of shopping when it's already raining.  $P(x | c)$  denotes probability of occurrence of rain while shopping.  $P(c)$  denotes probability of raining. When there is occurrence of discrete features (e.g., word counts for text classification), the multinomial naive bayes classifier is suitable option. It requires integer feature counts and based on frequency of the features and its label or classification. It can be used in multinomially distributed data (Scikit-learn, 2021).

### 2.2.2 Backend API Development

Flask, a micro web framework has been utilized for the API development of the sentimento project.

Reasons to use Flask over other backend tools for this project:

- i. Flask is lightweight, only useful modules to our system can be imported, less space consuming comparing to Django or other full-featured framework.
- ii. Previous experience of working with Flask framework.
- iii. Only API development is necessary with no need of user-interacting web pages.  
So, there is no need to use web page templates. In such case, Flask is one of the best tools.

### 2.2.3 Frontend Mobile Application Development

Flutter, cross platform framework is used for the user interface development work for the project.

Reasons to use Flutter instead of React Native for this project:

- i. Previous experience of working with Flutter during internship.
- ii. Easy to learn if there is clear understanding of object-oriented programming as Flutter is focused on widget-based interface development.
- iii. React Native uses middleware to translate developer's written JavaScript code to native components. Such code translation can create difficulty for smooth interaction (run time animations) on user interfaces. On the other hand, Flutter compiles code with support of C++ engine and with native iOS and Android platform. This makes Flutter to contain lesser performance issues. (Doshi, 2021)  
In Sentimento project, there is possibility to render user's saved reports in larger quantity. For such purpose, Flutter is suitable on this project.

## 2.3 Similar projects

Similar types of projects are sought from the project's research stage to requirement analysis and development, and inspiration is collected to produce a better solution for Sentimento's project.

### i. TSentiment (Mobile Application for tweet sentiment analysis)

It is an iOS application for tweets analysis based on hashtag. User has to choose a topic to which analysis is performed. The tweets analysed and percentage distribution of positivity/negativity found is shown in this platform (C9 Apps Desenvolvimento de Software Ltda ME, 2017).

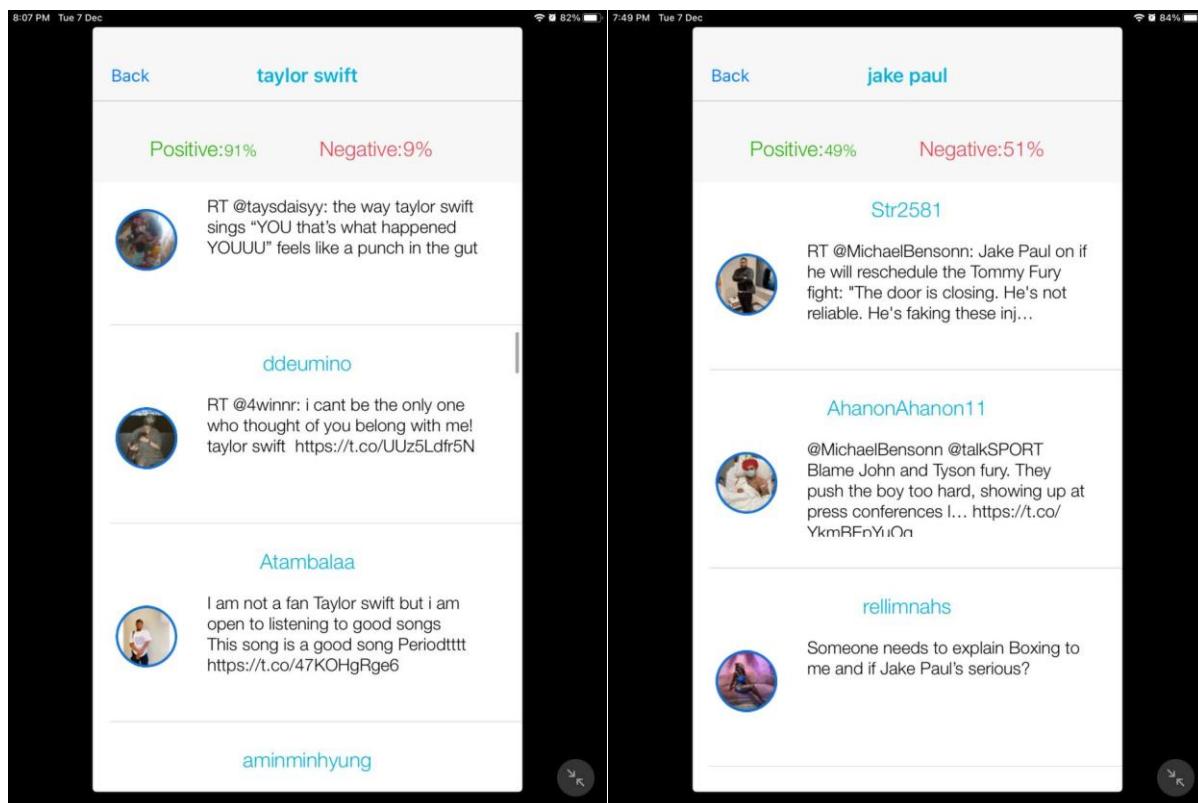


Figure 8 Sentiment analysis outcome between Taylor Swift and Jake Paul

Publisher of the application: C9 Apps Development de Software Ltd. ME

Platform: iOS and MacOS

## ii. Sentiment Analysis of Twitter (Mobile Application)

It is an android application for tweets analysis. Users can choose the topic and number of total tweets they want to analyse. The type of visualization output can also be chosen whether they want to view in graph or pie-chart. There is option to choose between polarity visualization and emotion. There is also feature to show recent searches by a user. Maximum 100 recent tweets would be analysed and tweets would be 7 days old. The application works with Naive Bayes Classifier and generates the sentiment in positive and negative polarity (Jay Khatri, 2020).

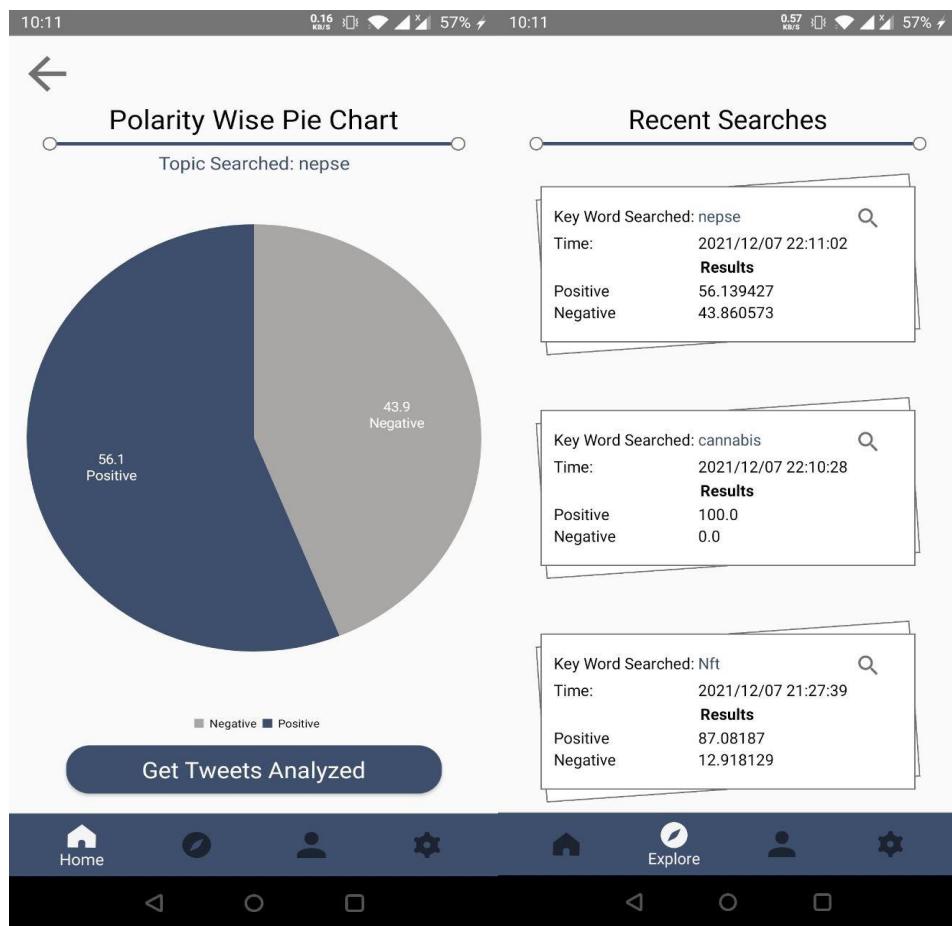


Figure 9 Analysis of tweets on NEPSE

Developers: Jay Khatri, Isha Khimsurya

Platform: Android

### iii. LinkedIn (Professional Networking Platform)

It is a successful and well-established platform which serves in professional networking, job portals and similar services. The vision of this platform is to create economic opportunity for every member of the global workforce (LinkedIn, 2021).

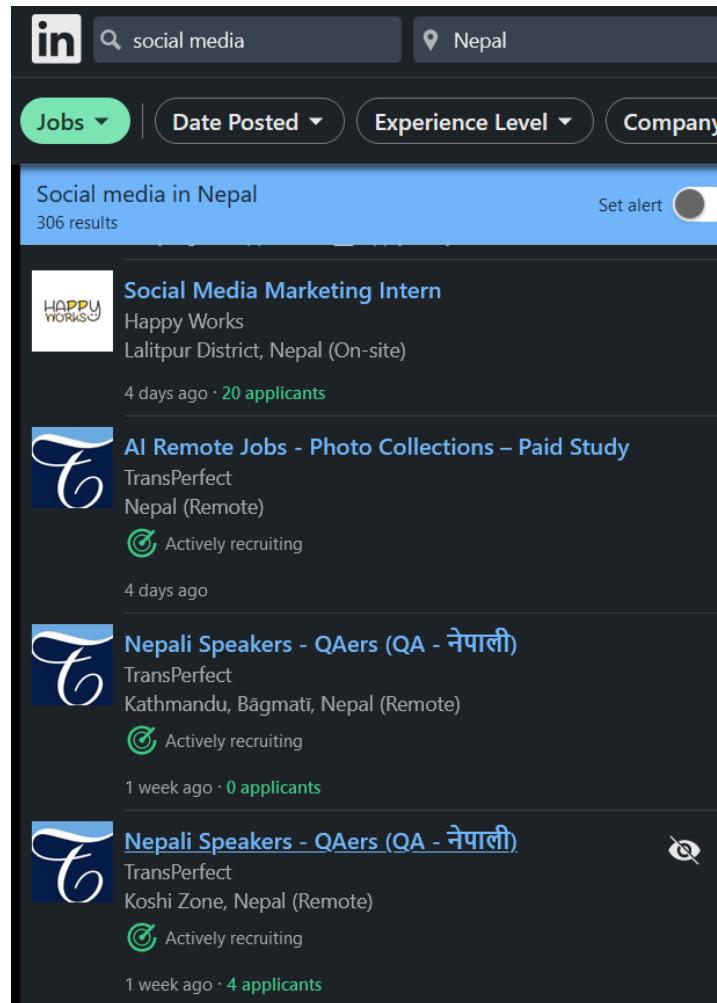


Figure 10 Searching social media related jobs in LinkedIn

LinkedIn is considered as similar projects to mine because there will be social media related job networking feature in my ‘Sentimento’ platform.

## 2.4 Comparison and uniqueness in my project

S. N	Comparison topic	Project 1	Project 2	Sentimento
1	User Authentication System	✗	✓	✓
2	YouTube Comments Analysis	✗	✗	✓
3	Tweets Analysis	✓	✓	✓
4	Custom Sentence/Paragraph Analysis	✗	✗	✓
5	User's defined maximum data to pick	✗	✓	✓
6	Report Saving Feature	✗	✗	✓
7	Custom API Keys	✗	✗	✓
8	Social Media Job Networking	✗	✗	✓

Table 1 Comparison of other projects and sentimento

According to the information supplied in the Google Play application description, the second project uses the Naive Bayes Classifier. Multinomial Naive Bayes is used in my research to classify data as sarcastic or not. The frequency of an event occurring is represented by Multinomial Nave Bayes. Because the goal is to increase the accuracy of the analysis, improvements to the algorithm may be made in future sprints. However, the Naive Bayes algorithm is set to be used in the classification section.

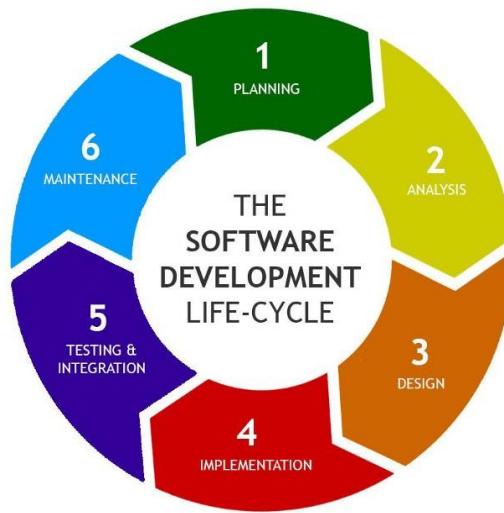
### Personal view on how the developed solution is better than similar projects

Sentimento platform isn't just for analysing tweets. YouTube comments and custom sentence/paragraphs are also supported. The sentiment report saving function allows users to access reports in the future and compare sentiment discrepancies between past and present sentiment changes on the same topic.

## Chapter 3: Development

### Software Development Life Cycle (SDLC):

It is a defined collection of rules/protocols for the creation of production-ready or large-scale projects. Following any software development life cycle, the project's integrity, general documentation, and security are the critical aspects for project success. Following software development principles, the project involves technical, management, finance, and every department of personnel (Young, 2013).



*Table 2 General principles of SDLC (Prokopisko, 2019)*

Software Development can be carried out following several methodologies based on each project's requirement and conditions. Waterfall, Agile, Rapid Application Development are popular methodologies. These methodologies have smaller branches with certain differences.

### 3.1 Considered Methodologies

#### **Waterfall**

It is a linear and traditional approach of software development. Each phase of process is carried out in sequential order. Next task is not executed without finishing the previous one (Young, 2013).

Good points to considerate:

- i. The straight goal focused approach makes the project well planned.
- ii. Minimizes cost for changing requirements and project is estimated to be well documented.

Bad points to considerate:

- i. The initial task plan limits further feature to implement in the middle of development.
- ii. The poor flexibility in making changes make essential new updates to come very late in the production. This is not feasible for current market scenario where rapid changes are essential to make if needed.

#### **Spiral Model**

This model is appropriate for unique projects where there is high risk of project failure.

Good points to considerate:

- i. Better for complicated projects where task to accomplish is very less clear.
- ii. Strong documentation control over approval of development stages.

Bad points to considerate:

- i. Not suitable for project with smaller scope and low risk projects.
- ii. End of the project is difficult to estimate.

### **Unified Process**

It is an interactive approach of development for larger projects with bureaucratic development teams. The principles of this methodology stand between waterfall and agile principles.

Good points to considerate:

- i. Highest risk containing tasks are prioritized first to maintain early break point. This allows easy cancellation of project in case of high failure possibility.
- ii. Performing several tasks like documentation, development and testing in parallel, labour pool can be utilized efficiently (Young, 2013).

Bad points to considerate:

- i. Documentation, artifact making is heavier than Agile.
- ii. The system to be developed for this project is moderate with less complexity, in such case, agile is more appropriate than unified process.

### **Agile**

Agile principles are very adaptive towards change in requirements and plans. The minimal cost and quality software are the key principles of agile methodology. It is similar to unified process but iterations are much short period of time in agile.

Good points to considerate:

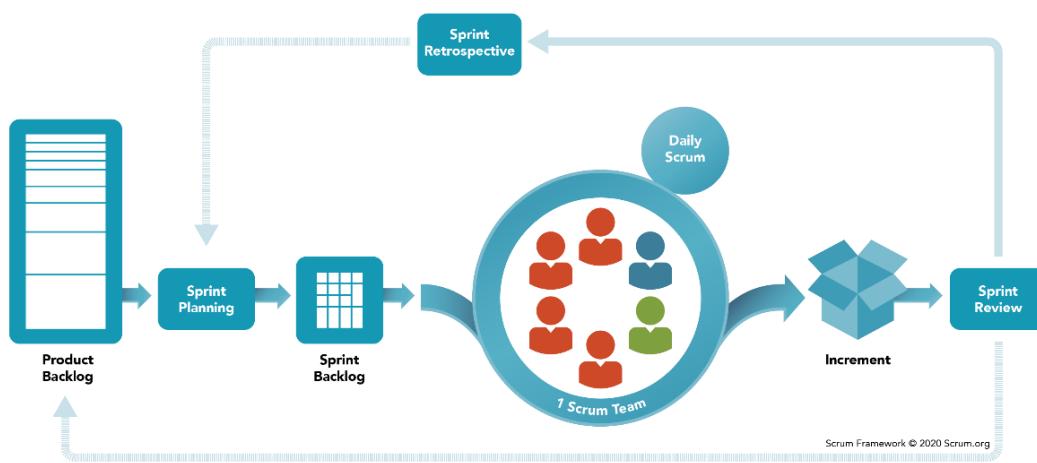
- i. The agile frameworks bring strong bond in team work as it promotes fail-early-fail-fast approach in the work. This empowers team to be self-managed.
- ii. The quality of the product is significantly better developed with agile frameworks as it focuses result over excessive documentation.

Bad points to considerate:

- i. The lack of work-in-progress control can result too many running tasks. This can generate long queues (Perisic, 2014).

### 3.2 Selected Methodology

The Scrum framework of agile methodology will be utilized for this project because it is appropriate for this project type. Agile characteristics such as flexible work divisions for individuals, working software over exhaustive documentation, and adapting to changes are all part of the Scrum methodology. Each sprint of the scrum is dedicated to completing specific tasks.



*Figure 11 Scrum methodology (scrum.org, 2020)*

The Scrum methodology is about carrying out the project based on multiple sprints. In each sprint, we carry out certain feature. Each sprint can be of one to two week long. The tasks to be done, the tasks to be reviewed are maintained in scrum. There is no hard rule for task assignment. Instead, the tasks can be chosen by each individual and primary focus is for completion of tasks in a sprint. Products are built within certain time frame and improves on each iteration from the previous one.

Three points to consider on sprint planning are as follows (Star Agile, 2020):

- i. What can be delivered in increment resulting from upcoming sprint?
- ii. How can the work needed to deliver an increment be achieved?
- iii. When is word considered 'done'?

Reasons to choose Scrum for this project ‘Sentimento’ are as follows:

- i. This is an individual project; all the tasks are self-done. Roles are not specified in scrum but main focus is on task completion. This provides flexibility to carry out multiple tasks in self-easiness.
- ii. Previous experience of scrum during my internship helped me to manage or breakdown work structure following scrum principles.
- iii. Technical challenges in my project are higher than documentation and scrum itself is effective to develop product more than artifacts focus.

Roles for ‘Sentimento’ project are as follows:

Product Owner, Scrum Master (Project Manager), Development Team are three roles in Scrum project (Glamazdina, 2021). As this is an individual final year project, all the three roles are allocated to myself as a sole performer of the project.

### 3.3 Phases of methodology

Scrum phases of Agile methodology are as follows:

i. Product Backlog Creation

The functional and non-functional requirements, as well as product features, are recorded at this phase for later implementation. The task description provided by the client is broken down into smaller technical components to complete. The product owner or client party meets with the scrum master to discuss the product's development. This engagement can happen at any time during the development process.

ii. Sprint planning and Backlog creation

The set of product backlog items are selected for the next sprint in this phase. Usually, a sprint last about two weeks, the short duration sprints provide the opportunity to accept client's further feedback. This provides enough time to adapt new changes if required. There is an additional benefit of extra time for bug addressing and debugging. After breaking down the problem domains of the sprint into smaller tasks, scrum team is also responsible for prioritizing the works (Star Agile, 2020).

### iii. Working on sprint/ Daily Scrum

Following the planning and appropriate coordination, this is the actual working stage on the product. A task progress board is kept up to date. Task cards are divided into four categories: tasks to accomplish, current tasks, tasks to review, and completed tasks.

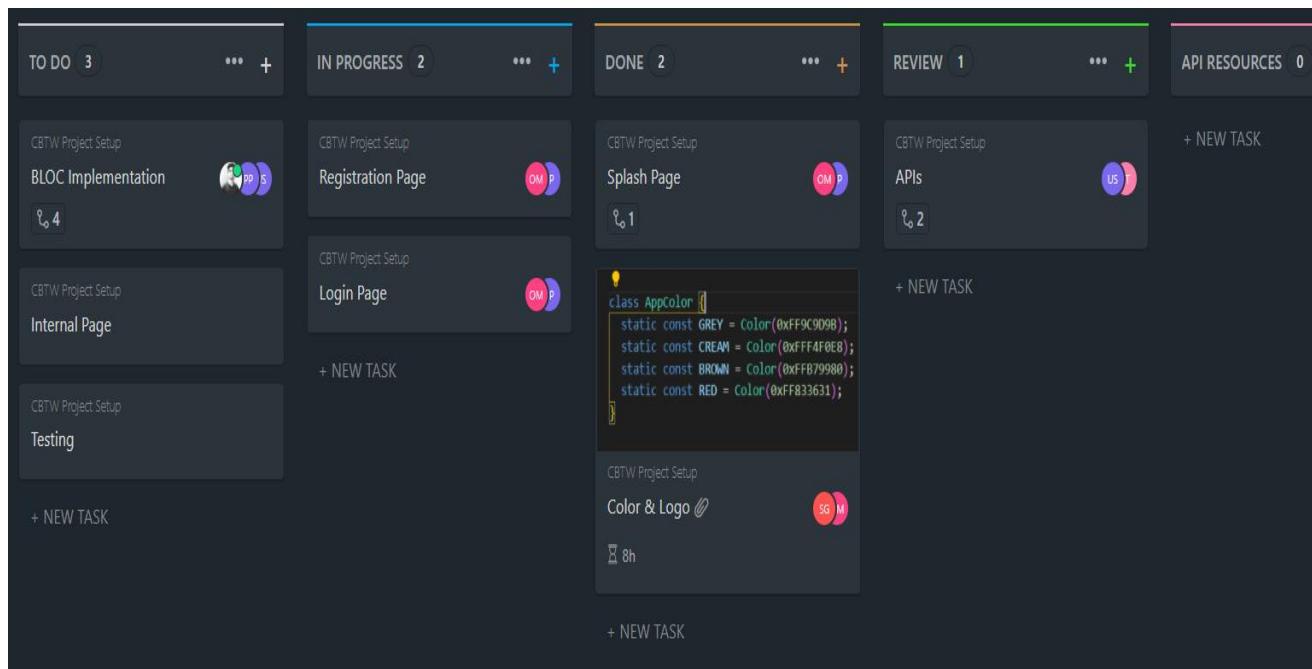


Figure 12 Sample of daily scrum board

Internal meeting of the developer team exists for less than 15minutes to synchronize the tasks for next 24hours. The work inspection and work forecast of team members are carried out for fine tuning of the product.

### iv. Sprint review or testing and product demonstration

This phase is about viewing the tasks done in this sprint. The quality of the sprint or deployment is evaluated and judged if any necessary changes are essential to make. The product owner describes what product backlog items have been ‘done’ and what not. The entire team collaborates on what to do next (Khristich, 2020).

v. Sprint retrospective

It is a final meeting and last phase of the scrum development process. It is attended by product owner/client, scrum master and development team. This is time boxed for forty-five minutes a week (Khristich, 2020).

Questions each team member answers in sprint retrospective are as follows (Linders, 2013):

- i. What worked well in previous sprint.
- ii. What did not work well, difficulties faced, requirements that were not fulfilled.
- iii. What should be done differently, changes a team member want to make in working process if any.

### 3.4 Survey Results

To obtain better and reliable data, the surveys were taken only from people who regularly uses social media (Instagram, Snapchat, LinkedIn) in their professional life.

Profession of the survey volunteers:

<b>Profession</b>	<b>Number of persons</b>
Student	12
Artist/Influencer	1
Data Analyst	1
Job Seeker	2
News Media Officer	2
University Professor	1

*Table 3 Survey volunteer's information*

### 3.4.1 Pre-survey results

Question 1: Are you familiar about sentiment analysis on user's activity?

19 responses

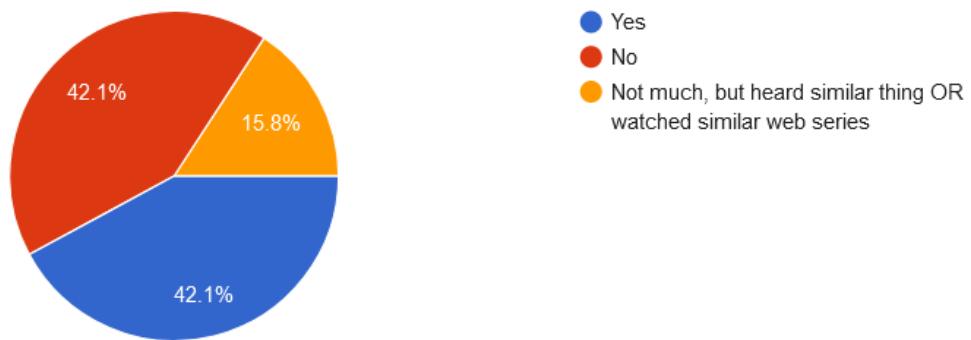


Figure 13 Social media user's familiarity with sentiment analysis

- i. The survey shows nearly half of the social media users are aware about how companies can use our social media activity data.
- ii. Forty two percent approximate users not knowing about their user activity indicates companies do not take sufficient steps to highlight how do they use their information for multiple purposes.
- iii. Sixteen percent of approximate users getting to know about how social media companies use our data indicates movies and web series are helping to make people understand about critical information.

Question 2: Do you think any valuable information can be extracted using user's comment, tweets, and posts?

19 responses

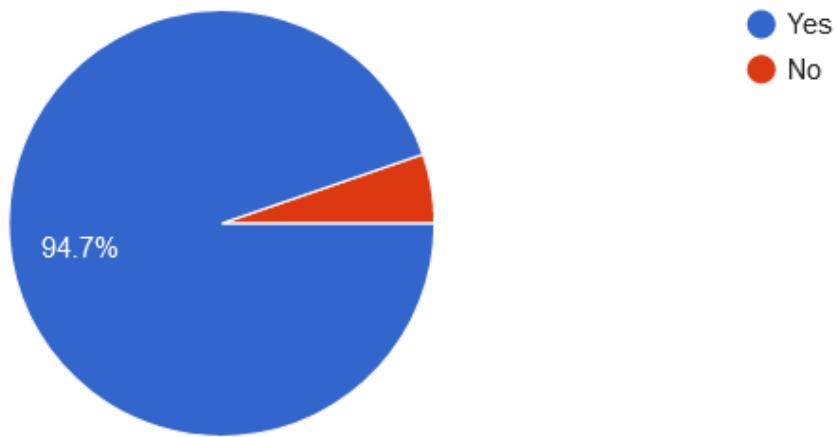


Figure 14 User's awareness on precise social media activity

- i. The answers from volunteers shows majority of the social media users who are aware about their online data are also precisely confident on what data can be useful for third factor use cases.

Question 3: Do you comment your opinion on YouTube videos?

19 responses

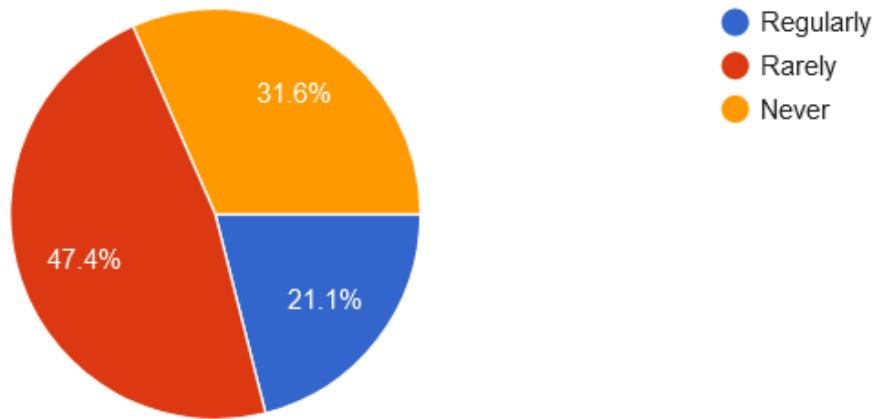


Figure 15 Chart showing user's interest to comment

- i. Almost half of the volunteers chooses to rarely comment on YouTube videos. This indicates that there is high probability for most of the comments found on YouTube are spam, unwanted promotions or sometimes scam initiations.
- ii. The answers suggest to make a user's opinion filtration system from bulk of spam comments. This will help to perform sentiment analysis on reliable data excluding senseless spam comments.

Question 4: Do you often tweet about ongoing internet topics, events or news?

19 responses

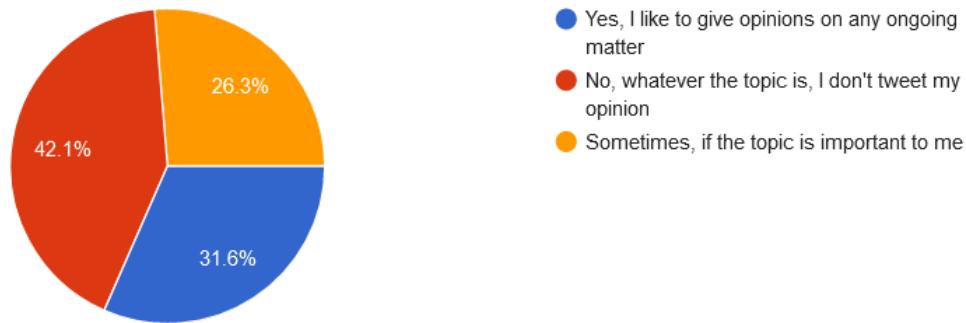


Figure 16 Chart showing people's interest to tweet

- i. The survey answer tells people are getting less interested to tweet about any particular topics than in past. Before few years, the tweeting culture was booming significantly.
- ii. People's awareness on data privacy has made people more conscious on where to express themselves.
- iii. Twenty six percent approximate users ready to tweet on critical social, global issues indicates people are ready to unite and raise voice on any important issues.

Question 5: If there was a tool that will give you summary report of public sentiment on any internet topics, will you use such tool?

19 responses

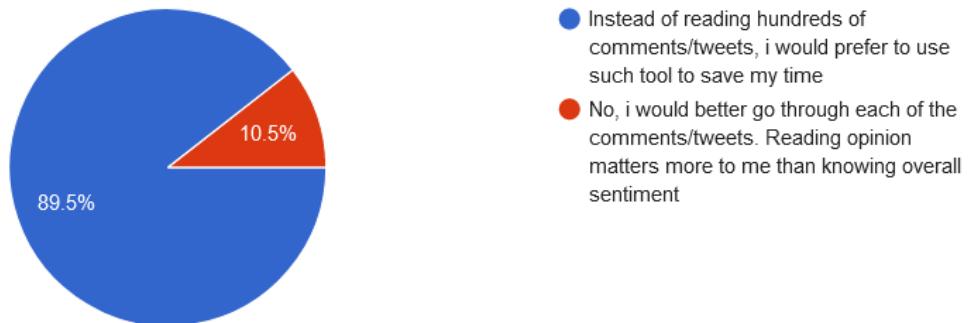


Figure 17 Chart showing user's interest to use sentiment tool

- i. Majority of the survey volunteers are interested in using tools that saves time to understand mass sentiment on particular topic.
- ii. One slice of a pizza or ten percent people are more interested to go through each of the comments/tweets by themselves.

Question 6: Do you think, a separate platform to connect just social media related job recruiter and job seeker is a better idea?

19 responses

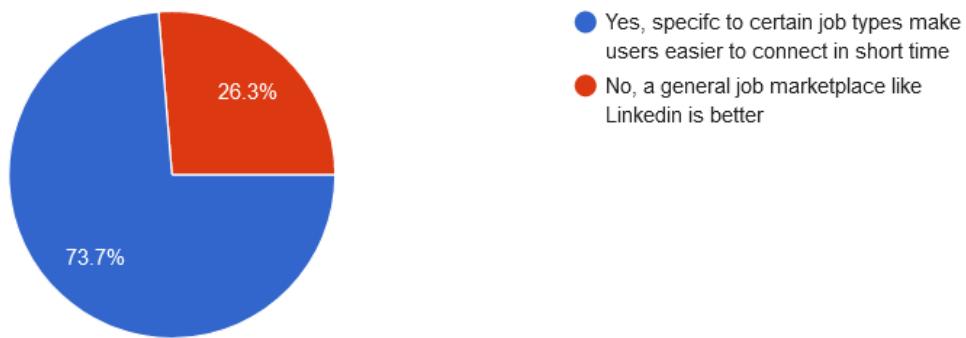


Figure 18 Chart showing people's interest looking for LinkedIn alternatives

- i. Seventy three percent is a significant number of people interested to use their profession specific job platform instead of generic job platforms like LinkedIn.
- ii. The user's interest rate makes job seeker and provider connecting feature to add to the platform requirements.

### 3.4.2 Post-survey results

Question 1: Which social media platforms do you want to use for obtaining mass sentiment/opinion?

19 responses

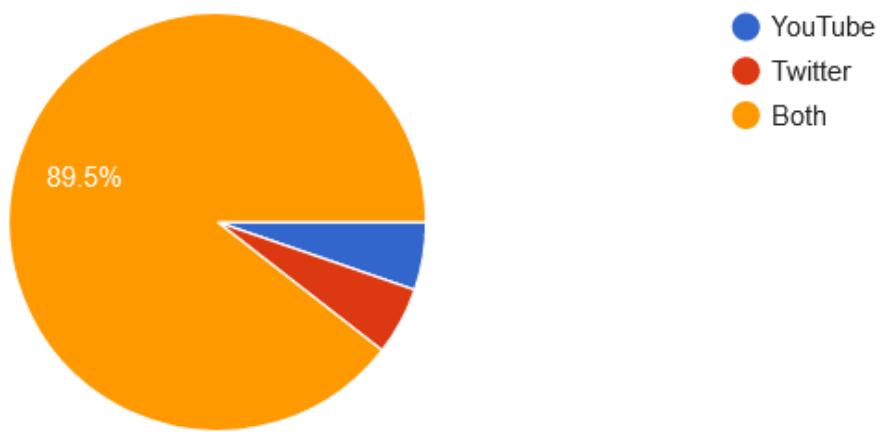


Figure 19 Chart showing social platform demand for analysis

- i. Majority of the answers from post-survey showed their interest to use both YouTube and Twitter platform for sentiment analysis.
- ii. Both platforms are now considered to integrate on Sentimento's system after requirement analysis.

Question 2: Do you want to use your own third-party access tokens or will it be easier for you to not care about it?

19 responses

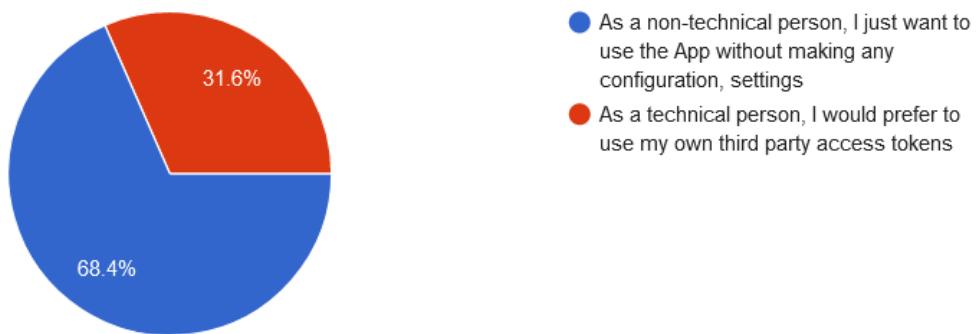


Figure 20 Chart showing user's interest to use their own third-party credentials

- i. The survey shows the platform will be used by both technical and non-technical users.
- ii. There is strong need to implement a system where user can either use their own third-party access tokens or can be dependent with platform itself to serve both type of users.

## Question 3: About user login preferences

19 responses



Figure 21 Chart showing user's login preferences

- i. Survey report shows all post survey volunteers want to be logged in to the platform unless they want to logout by themselves.
- ii. All the potential users find logging in every time quite tedious. So, considering this, bearer token mechanism will be used with no date expiry to keep user logged in.

Question 4: Choose one or many sentiment categories you want in the platform

19 responses

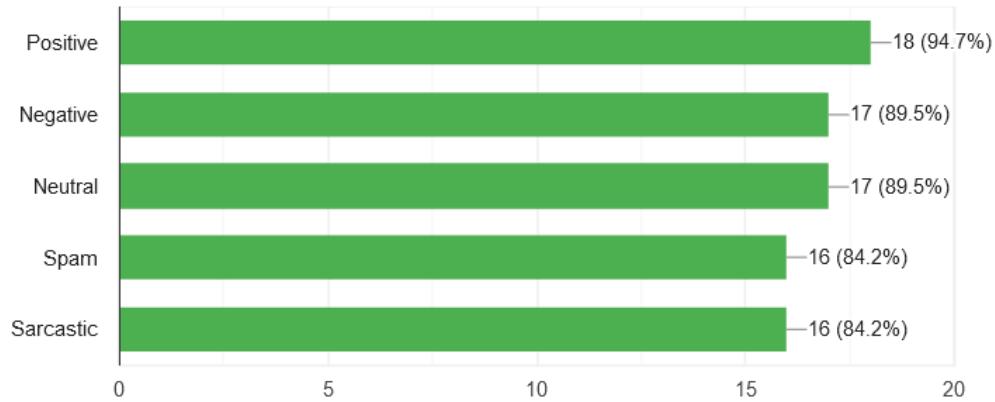
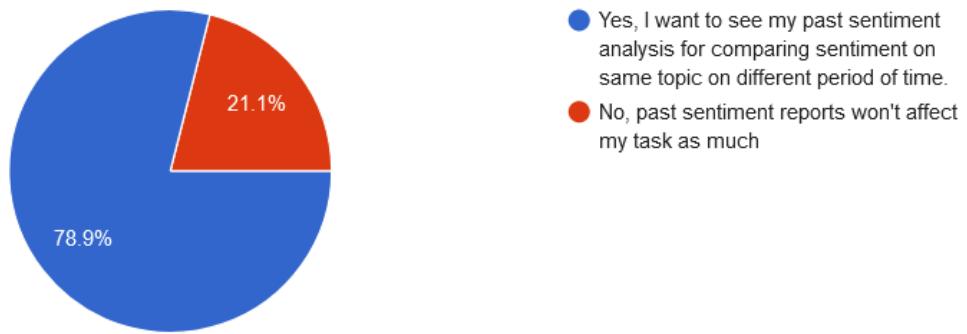


Figure 22 Chart showing what sentiment labels user want

- i. Out of total nineteen responses, all the five sentiment categories are strongly chosen.
- ii. All the five sentiment labels will be considered for development work.

Question 5: Do you want to see what analysis you did the last time?

19 responses

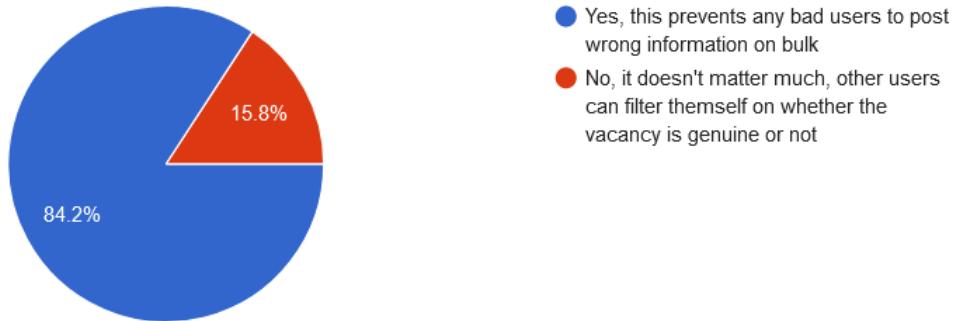


*Figure 23 Chart showing user's interest on report saving feature*

- i. Nearly eighty percent of survey volunteers want to view their past sentiment reports.
- ii. Twenty one percent of people not agreeing to record their past reports is also a considerable number of users. Considering this, instead of storing all the past reports, the platform will have feature where users can save the sentiment report if they want to.

Question 6: Do you think users should be limited on how many job vacancies they can post at a time?

19 responses



*Figure 24 Chart showing user's opinion on limiting vacancy post at a time*

- i. Eighty four percent of people are conscious about chances of false vacancies post if users can post any number of vacancies.
- ii. Near to 16 percent of people desires to post any number of vacancies they want. Considering both type of users, a threshold limit of five posts can be made at a time. Any previous post can be deleted to other new posts. This feature satisfies both kind of user preferences.

### 3.5 Requirement Analysis

After doing research about social media platforms, analysing what can be done for the project with supervisors and evaluating the conducted survey results, following requirement analysis has been done:

#### Functional requirements:

##### i. Primary Features

<u>Req.ID</u>	<u>Requirement Description</u>						
FR. 1	<p>User can register to system with unique and valid registration detail</p> <table border="1" data-bbox="352 853 1405 1085"> <tr> <td colspan="2" data-bbox="352 853 1405 908">System Requirement</td></tr> <tr> <td data-bbox="352 908 474 963">SR. 1</td><td data-bbox="474 908 1405 963">User can provide their correct registration information</td></tr> <tr> <td data-bbox="352 963 474 1085">SR. 2</td><td data-bbox="474 963 1405 1085">Backend responds with registration success or fail according to user provided information</td></tr> </table>	System Requirement		SR. 1	User can provide their correct registration information	SR. 2	Backend responds with registration success or fail according to user provided information
System Requirement							
SR. 1	User can provide their correct registration information						
SR. 2	Backend responds with registration success or fail according to user provided information						
FR. 2	<p>User can login to system with their credentials used on registration</p> <table border="1" data-bbox="352 1161 1405 1393"> <tr> <td colspan="2" data-bbox="352 1161 1405 1216">System Requirement</td></tr> <tr> <td data-bbox="352 1216 474 1271">SR. 3</td><td data-bbox="474 1216 1405 1271">User can provide their correct login credentials</td></tr> <tr> <td data-bbox="352 1271 474 1393">SR. 4</td><td data-bbox="474 1271 1405 1393">Backend responds with login fail in case of invalid information or else user will be forwarded to dashboard</td></tr> </table>	System Requirement		SR. 3	User can provide their correct login credentials	SR. 4	Backend responds with login fail in case of invalid information or else user will be forwarded to dashboard
System Requirement							
SR. 3	User can provide their correct login credentials						
SR. 4	Backend responds with login fail in case of invalid information or else user will be forwarded to dashboard						
FR. 3	<p>User can perform sentiment analysis on YouTube comments and tweets</p> <table border="1" data-bbox="352 1465 1405 1774"> <tr> <td colspan="2" data-bbox="352 1465 1405 1520">System Requirement</td></tr> <tr> <td data-bbox="352 1520 474 1664">SR. 5</td><td data-bbox="474 1520 1405 1664">User can provide YouTube video URL or topic of tweet to perform sentiment analysis</td></tr> <tr> <td data-bbox="352 1664 474 1774">SR. 6</td><td data-bbox="474 1664 1405 1774">Backend responds with sentiment report if valid input is given by user.</td></tr> </table>	System Requirement		SR. 5	User can provide YouTube video URL or topic of tweet to perform sentiment analysis	SR. 6	Backend responds with sentiment report if valid input is given by user.
System Requirement							
SR. 5	User can provide YouTube video URL or topic of tweet to perform sentiment analysis						
SR. 6	Backend responds with sentiment report if valid input is given by user.						

FR. 4	<p>User can view open job vacancies and skill offering from other users of platform</p> <table border="1" data-bbox="349 297 1405 582"> <thead> <tr> <th colspan="2" data-bbox="349 297 458 361">System Requirement</th></tr> </thead> <tbody> <tr> <td data-bbox="349 361 458 477">SR. 7</td><td data-bbox="458 361 1405 477">User can choose either job vacancies or freelancing services to view</td></tr> <tr> <td data-bbox="349 477 458 582">SR. 8</td><td data-bbox="458 477 1405 582">Backend responds with currently available vacancies posted by other users of platform</td></tr> </tbody> </table>	System Requirement		SR. 7	User can choose either job vacancies or freelancing services to view	SR. 8	Backend responds with currently available vacancies posted by other users of platform
System Requirement							
SR. 7	User can choose either job vacancies or freelancing services to view						
SR. 8	Backend responds with currently available vacancies posted by other users of platform						
FR. 5	<p>User can post social media related job vacancies and their skill offering in the platform</p> <table border="1" data-bbox="349 737 1405 1022"> <thead> <tr> <th colspan="2" data-bbox="349 737 1405 800">System Requirement</th> </tr> </thead> <tbody> <tr> <td data-bbox="349 800 458 916">SR. 9</td><td data-bbox="458 800 1405 916">User can provide information about their vacancy or freelancing skill.</td></tr> <tr> <td data-bbox="349 916 458 1022">SR. 10</td><td data-bbox="458 916 1405 1022">Backend stores the data in the database that is accessible by other users later</td></tr> </tbody> </table>	System Requirement		SR. 9	User can provide information about their vacancy or freelancing skill.	SR. 10	Backend stores the data in the database that is accessible by other users later
System Requirement							
SR. 9	User can provide information about their vacancy or freelancing skill.						
SR. 10	Backend stores the data in the database that is accessible by other users later						
FR. 6	<p>User can view their detailed profile</p> <table border="1" data-bbox="349 1115 1405 1400"> <thead> <tr> <th colspan="2" data-bbox="349 1115 1405 1178">System Requirement</th> </tr> </thead> <tbody> <tr> <td data-bbox="349 1178 458 1294">SR. 11</td><td data-bbox="458 1178 1405 1294">Profile viewing REST API request can go from frontend to backend with user's authentication detail</td></tr> <tr> <td data-bbox="349 1294 458 1400">SR. 12</td><td data-bbox="458 1294 1405 1400">Backend responds with user's profile data if authentic request is provided.</td></tr> </tbody> </table>	System Requirement		SR. 11	Profile viewing REST API request can go from frontend to backend with user's authentication detail	SR. 12	Backend responds with user's profile data if authentic request is provided.
System Requirement							
SR. 11	Profile viewing REST API request can go from frontend to backend with user's authentication detail						
SR. 12	Backend responds with user's profile data if authentic request is provided.						
FR. 7	<p>User can save and view their past sentiment analysis reports</p> <table border="1" data-bbox="349 1469 1405 1776"> <thead> <tr> <th colspan="2" data-bbox="349 1469 1405 1533">System Requirement</th> </tr> </thead> <tbody> <tr> <td data-bbox="349 1533 458 1649">SR. 13</td><td data-bbox="458 1533 1405 1649">User can have the option to save sentiment report for further uses.</td></tr> <tr> <td data-bbox="349 1649 458 1776">SR. 14</td><td data-bbox="458 1649 1405 1776">Backend stores the report in the database.</td></tr> </tbody> </table>	System Requirement		SR. 13	User can have the option to save sentiment report for further uses.	SR. 14	Backend stores the report in the database.
System Requirement							
SR. 13	User can have the option to save sentiment report for further uses.						
SR. 14	Backend stores the report in the database.						

Table 4 Functional requirements

ii. Authentication

Bearer authentication tokens will be used by the system to authenticate user's request to backend.

iii. Reporting/Documentation requirements

Use case diagram, system architecture, flowchart and similar diagrams are designed before the start of development of the project.

Non-functional requirements:

- i. Performance: To reduce the memory usage and shorten sentiment analysis algorithm processing time, the machine learning model is to be serialized into pickle file.
- ii. Scalability: The backend of the project is to be hosted in cloud for larger scale accessibility of the project.
- iii. Security: For user's security reasons, the platform should not store third party access tokens. Considering it, the tokens will be locally saved in user's device.

A proper software requirement specification document made during the requirement analysis phase of the project is attached in the appendix section of the report.

### 3.6 Design and Logical Diagrams

#### 3.6.1 Entity Relationship Diagram

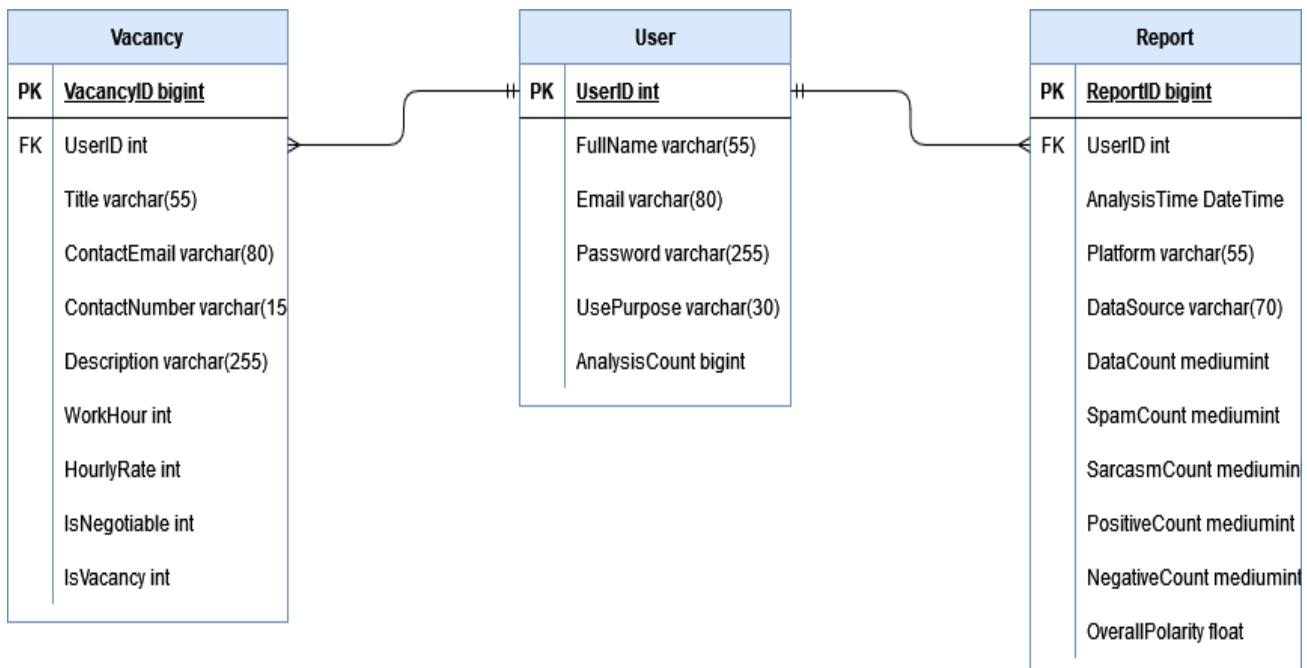


Figure 25 ERD of Sentimento's database

### 3.6.2 System Architecture

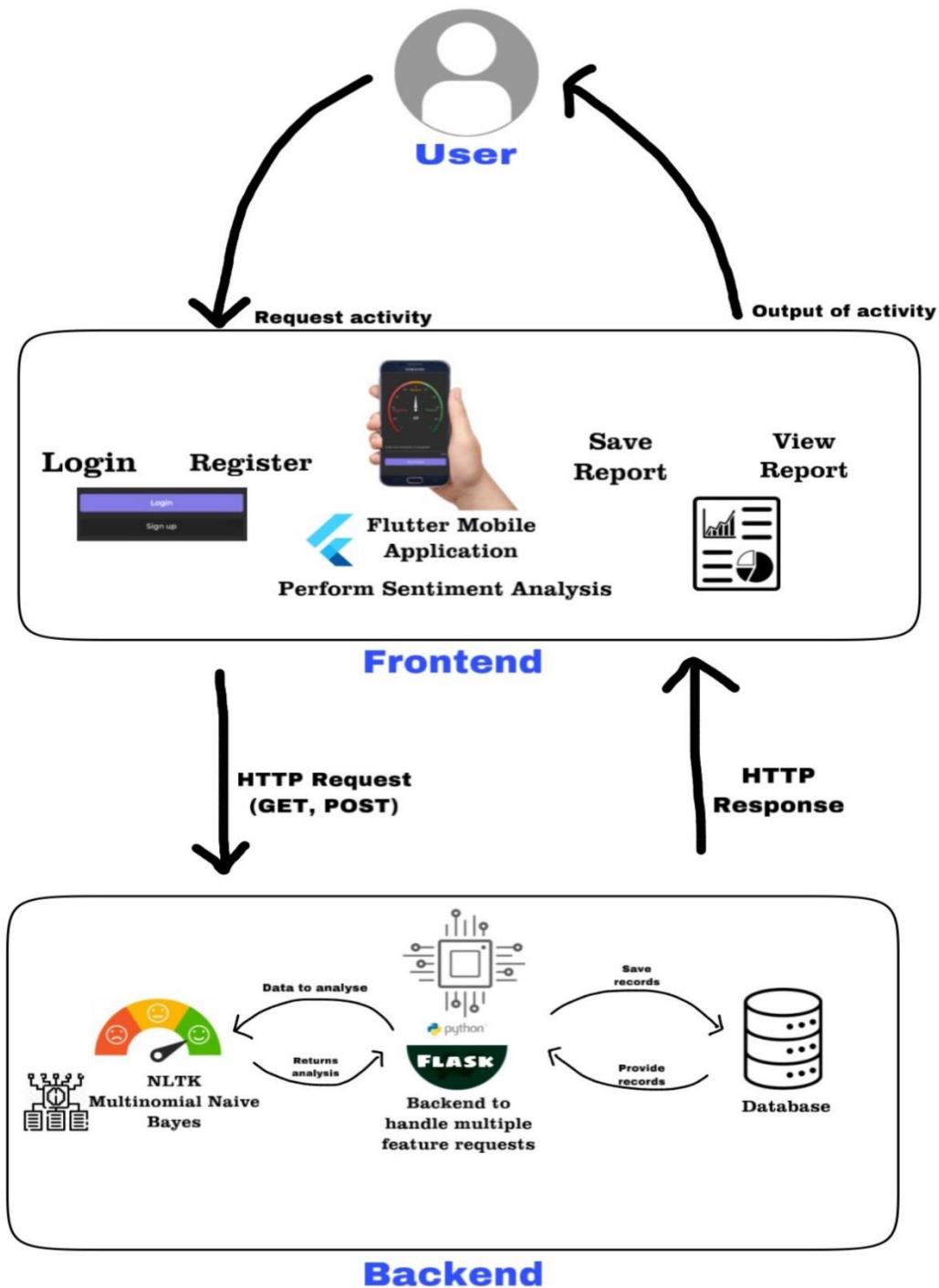


Figure 26 System architecture of Sentimento platform

### 3.6.3 Overall Use Case Diagram



Figure 27 Use Case Diagram of overall system

### 3.6.4 Overall Context Level Diagram

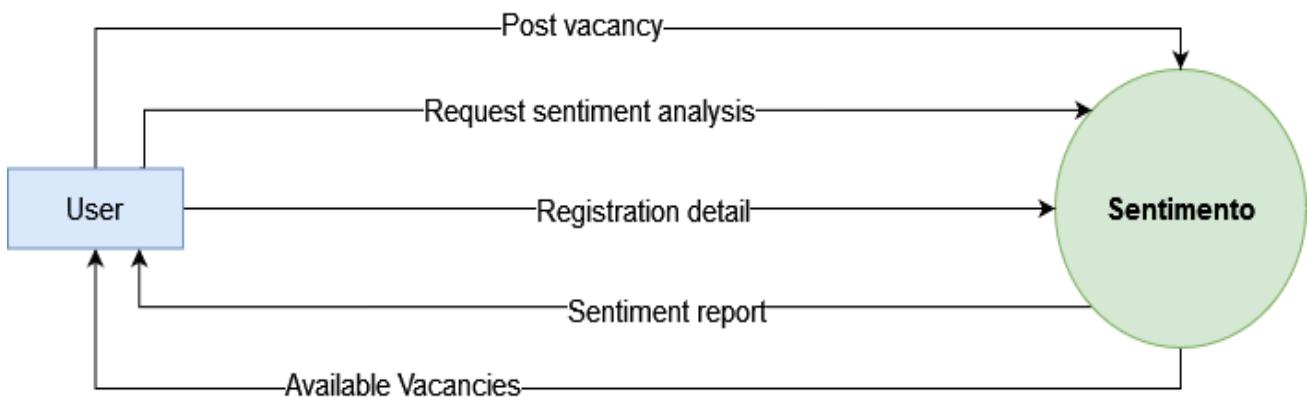


Figure 28 Context level diagram

### 3.6.5 Flowchart of sentiment analysis module

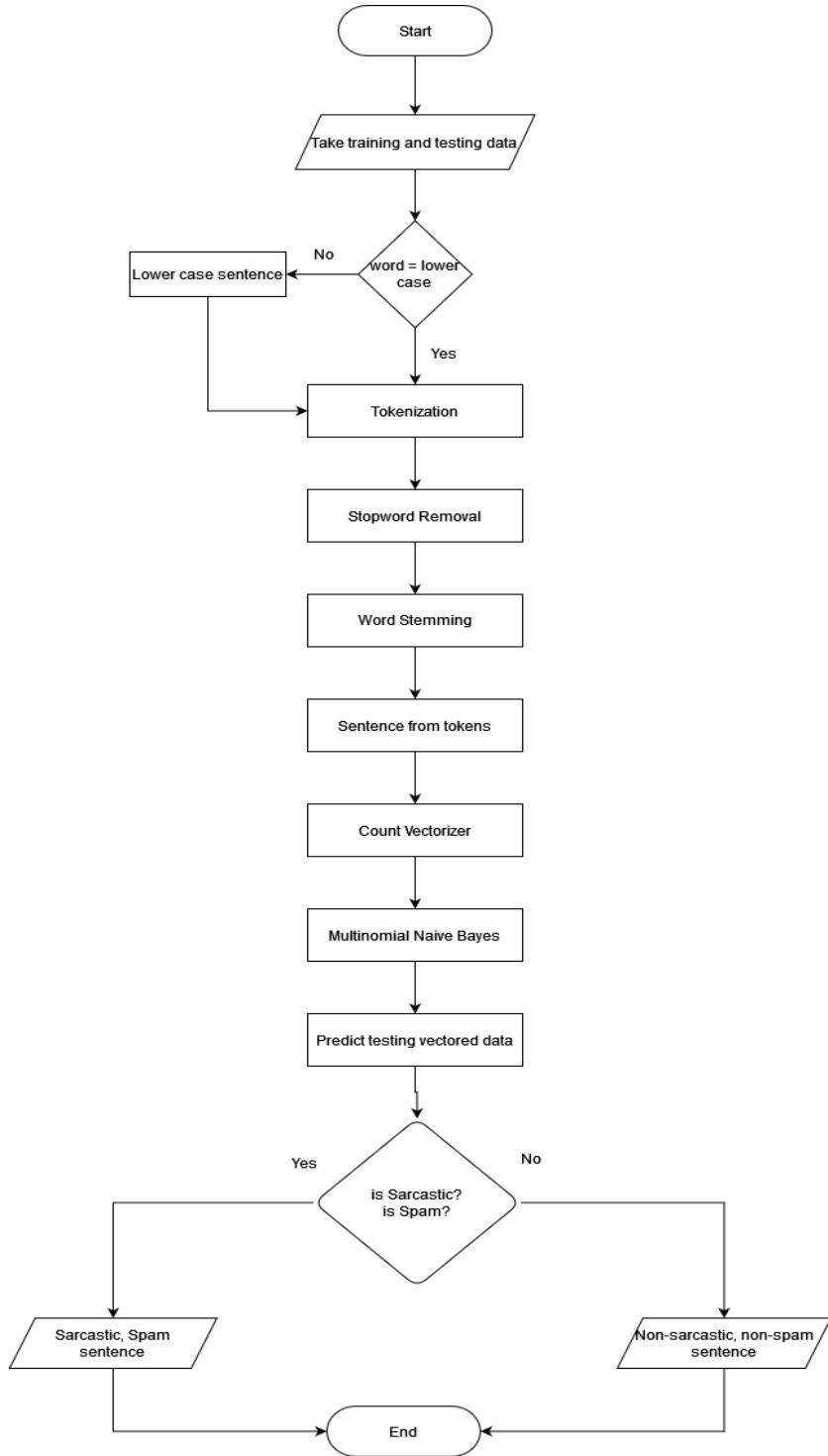


Figure 29 Flowchart of sentiment analysis module

### 3.6.6 Communication diagrams

#### 3.6.6.1 Communication diagram for sentiment analysis

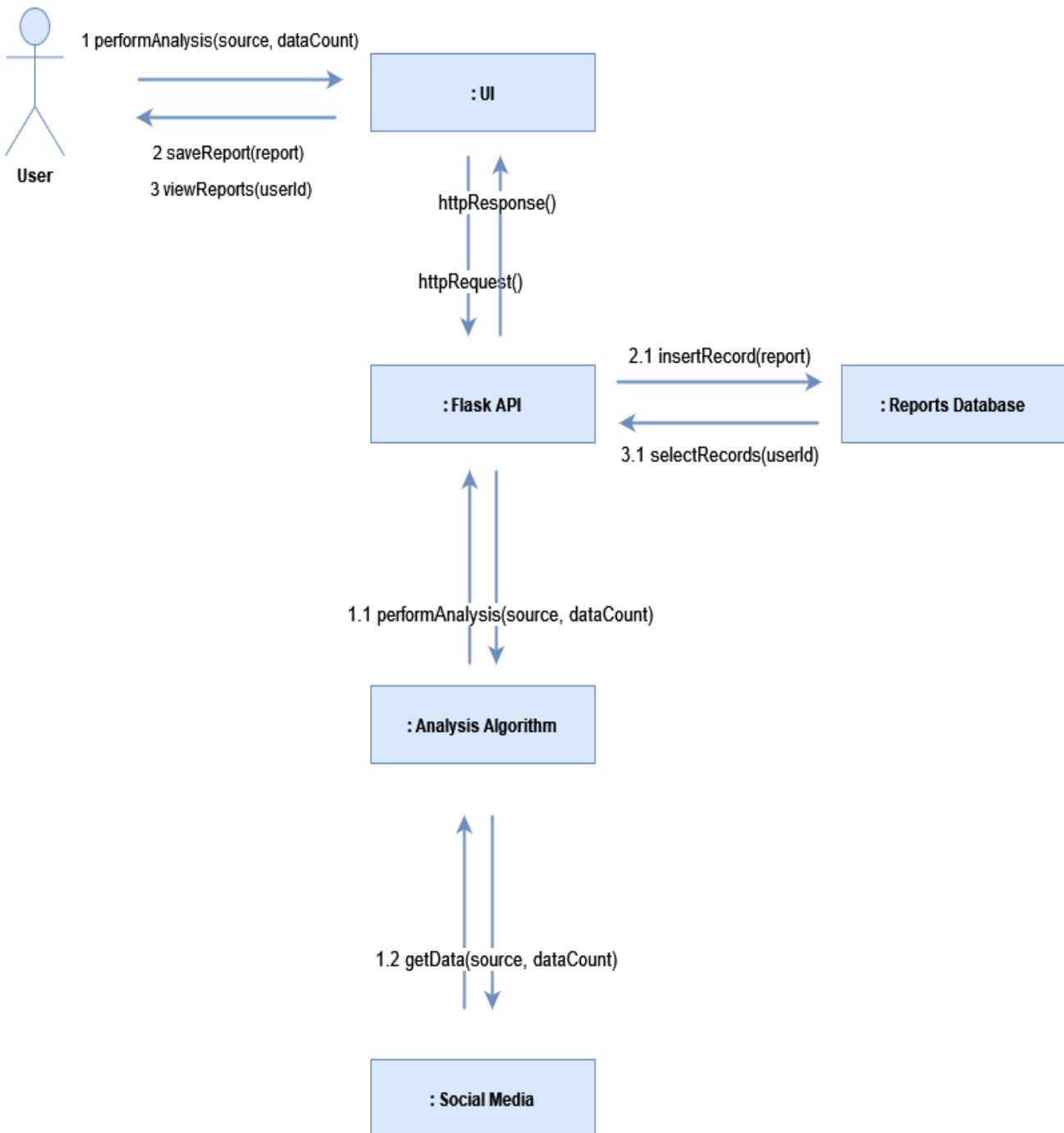


Figure 30 Communication diagram for performing sentiment analysis

## 3.6.6.2 Communication diagram for posting vacancy

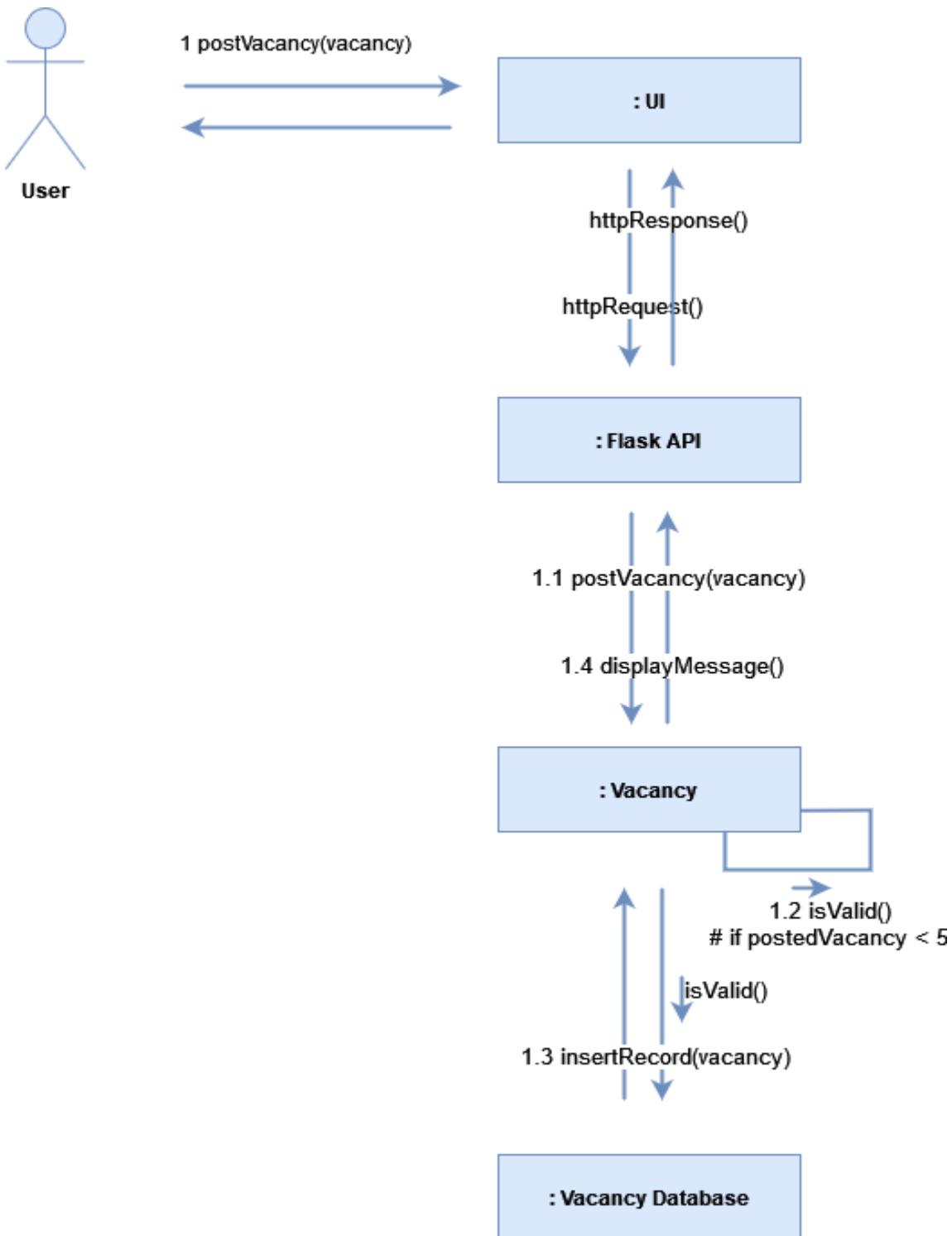


Figure 31 Communication diagram for posting vacancy

### 3.6.7 Sequence Diagrams

#### 3.6.7.1 Sequence diagram for user login

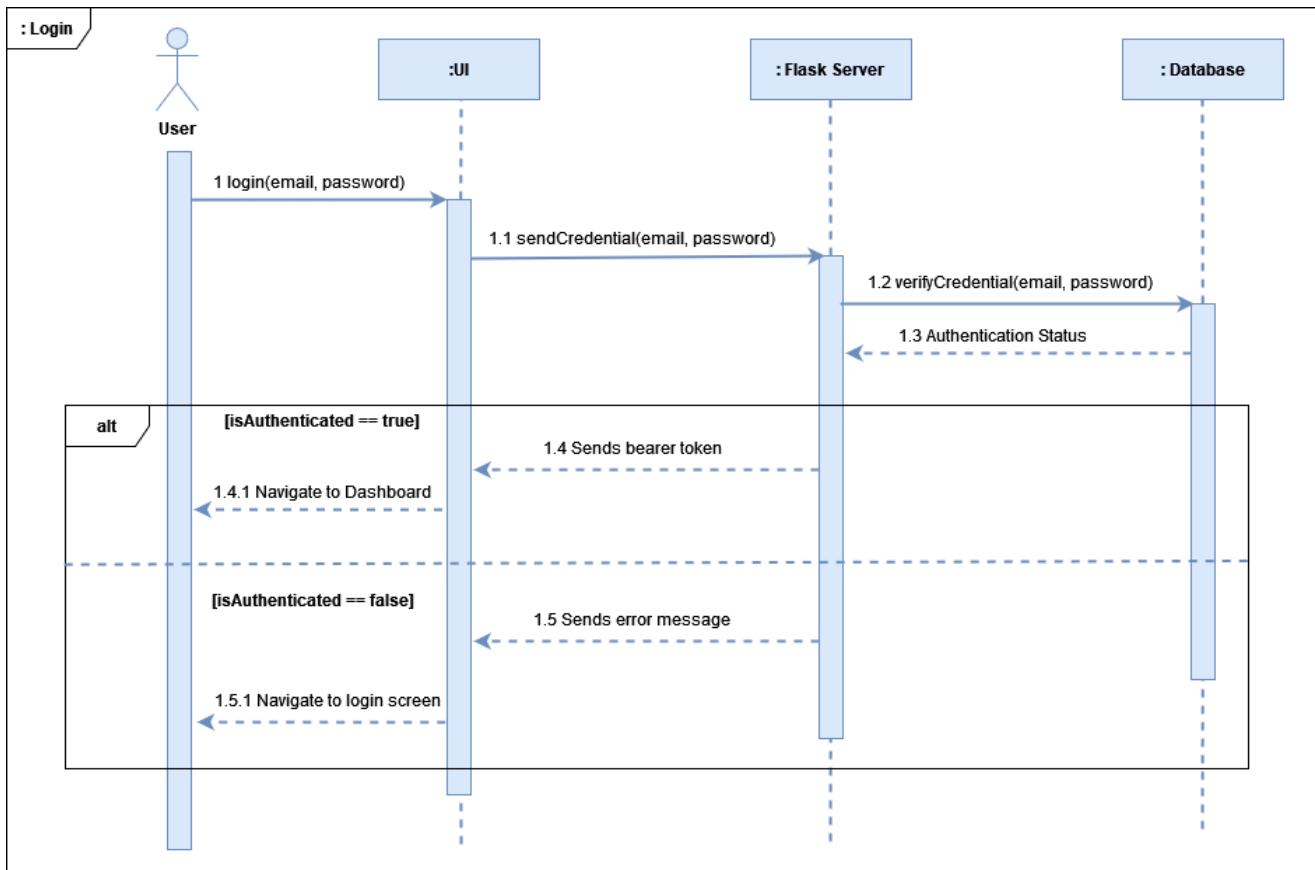


Figure 32 Sequence diagram for logging in user

### 3.6.7.2 Sequence diagram for performing sentiment analysis on tweets

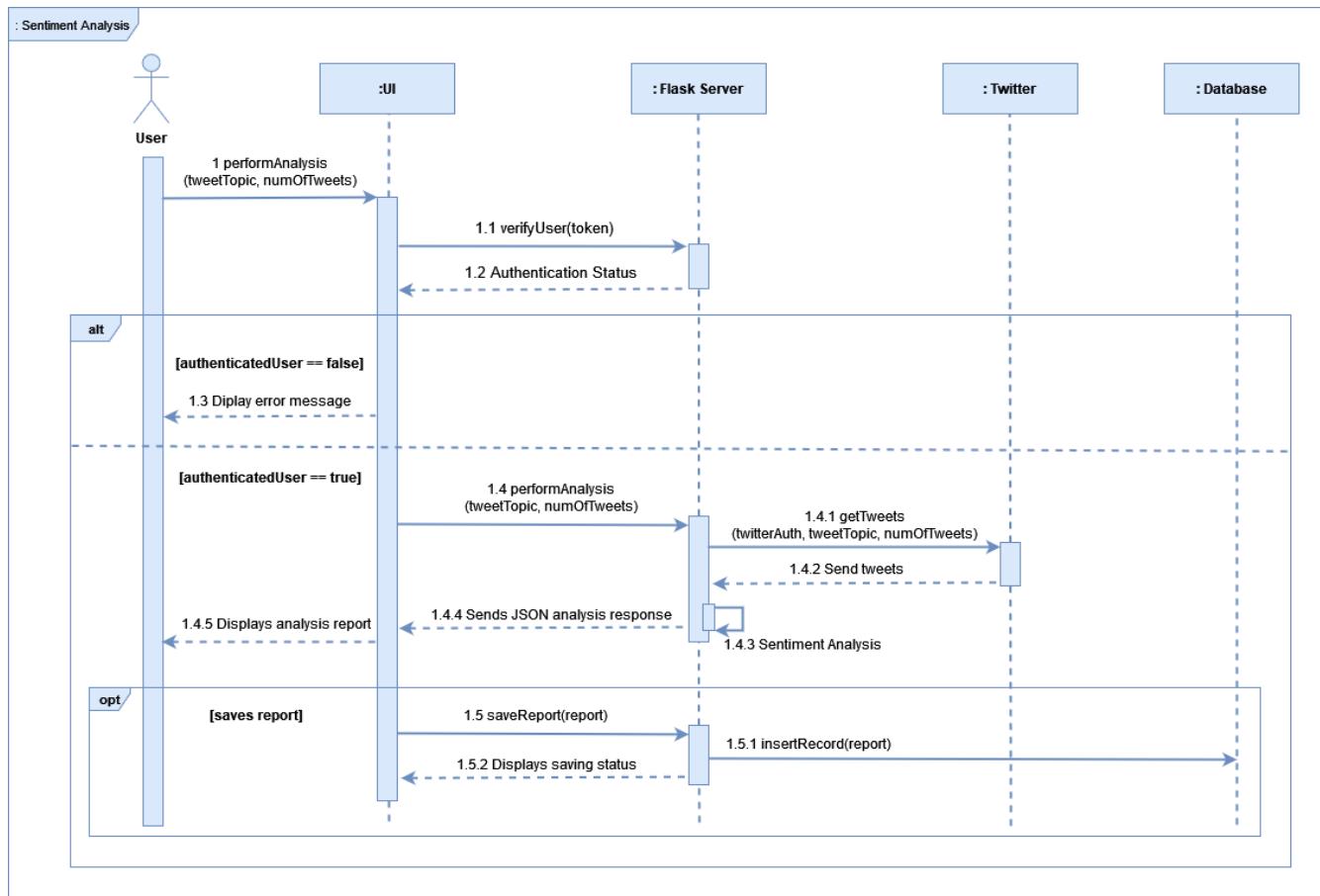


Figure 33 Sequence diagram for tweet sentiment analysis

## 3.6.7.3 Sequence diagram for viewing freelancers

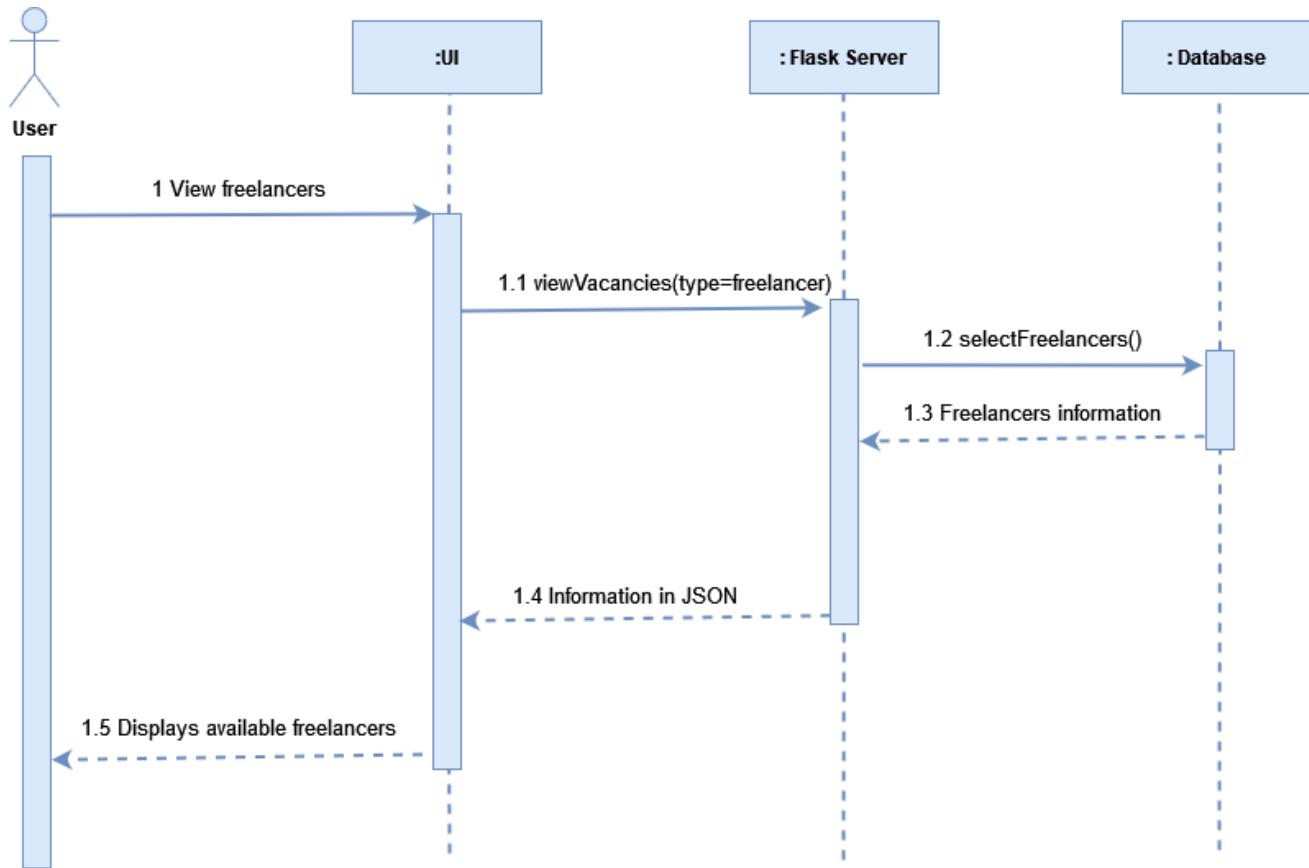


Figure 34 Sequence diagram for viewing freelancers

## 3.6.8 Structure chart for deleting vacancy

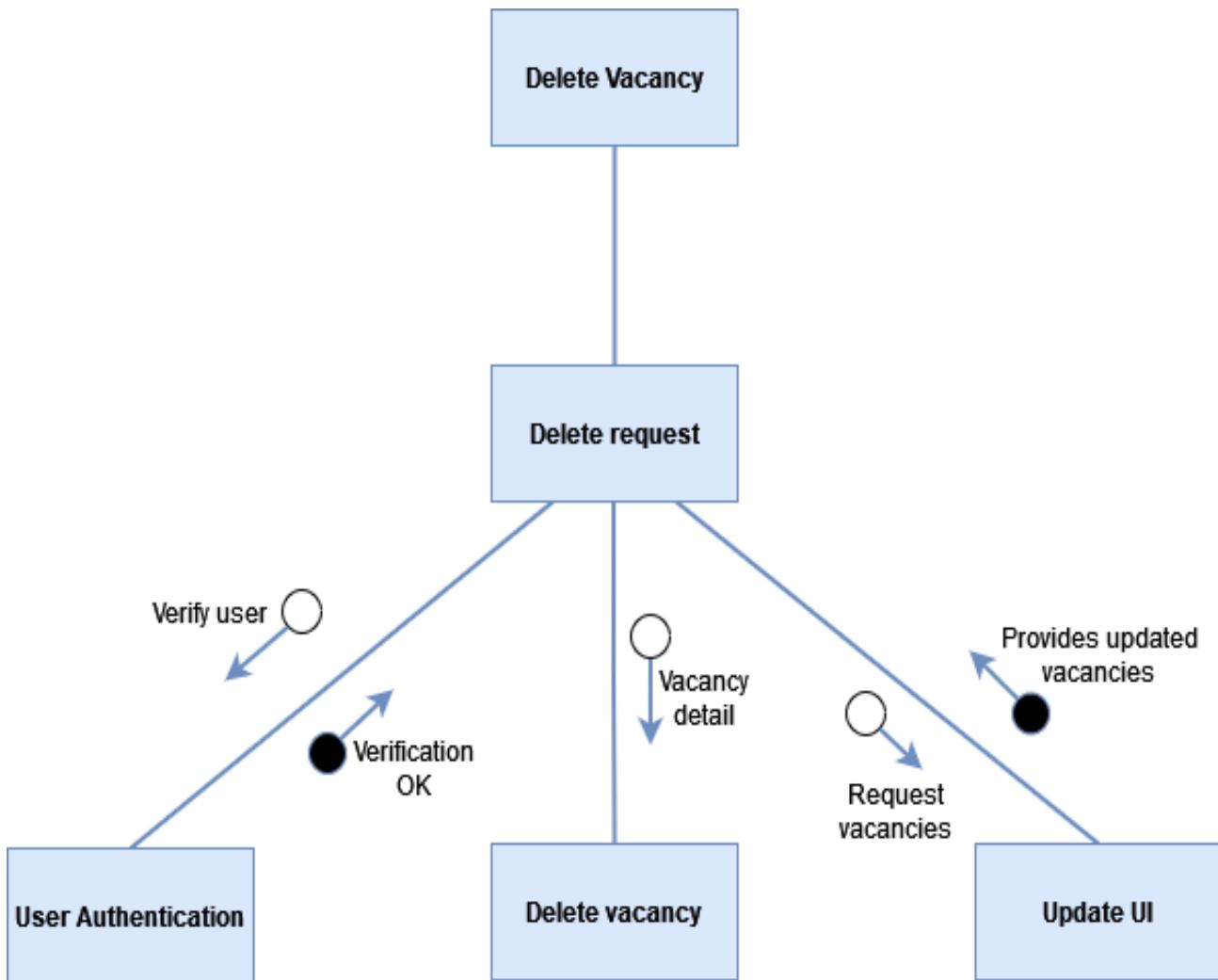


Figure 35 Structure chart for deleting vacancy

## 3.6.9 Activity Diagram to view saved sentiment reports

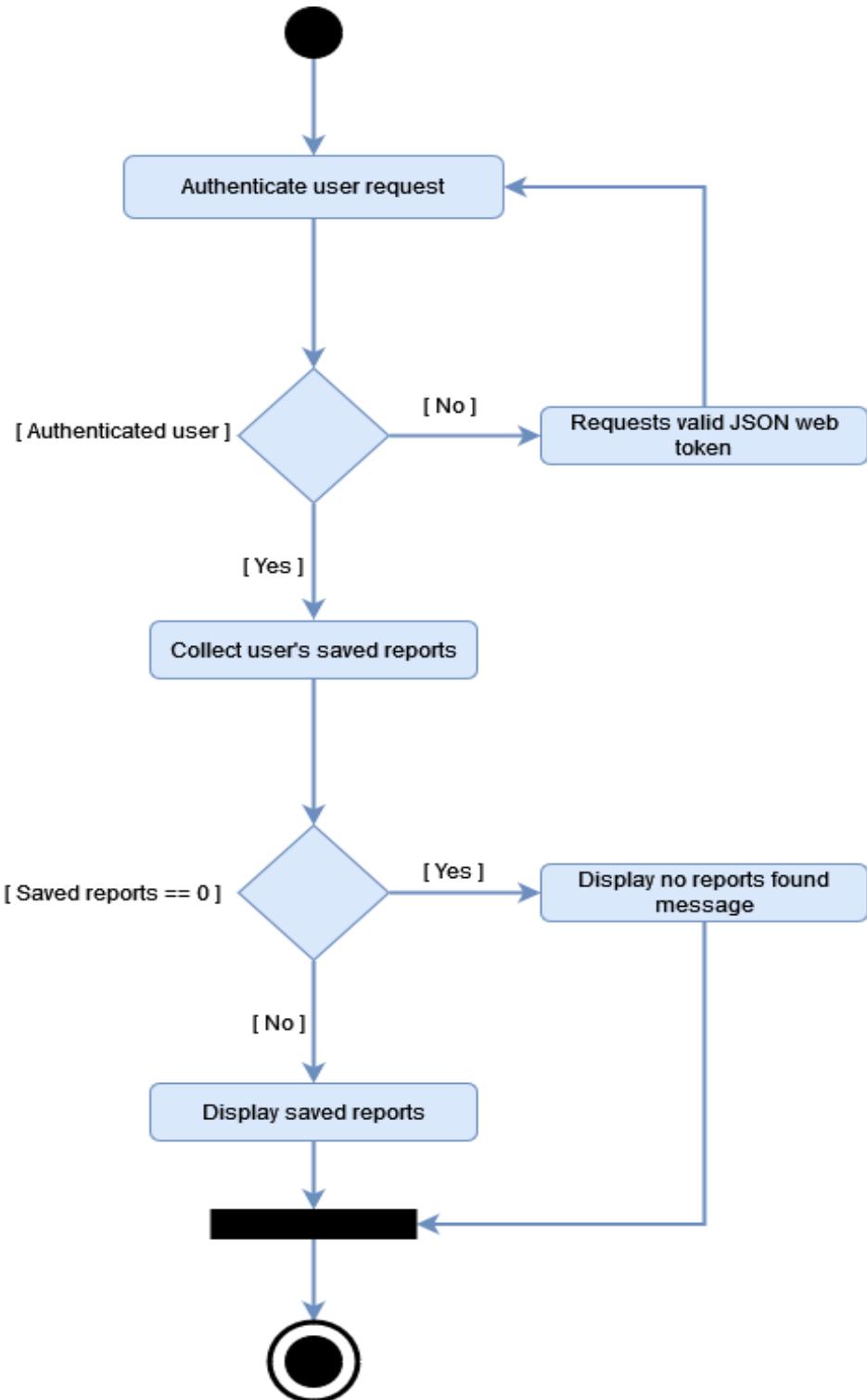


Figure 36 Activity diagram to view saved reports

### 3.6.10 Data flow diagrams

#### 3.6.10.1 Data flow diagram for user login and registration

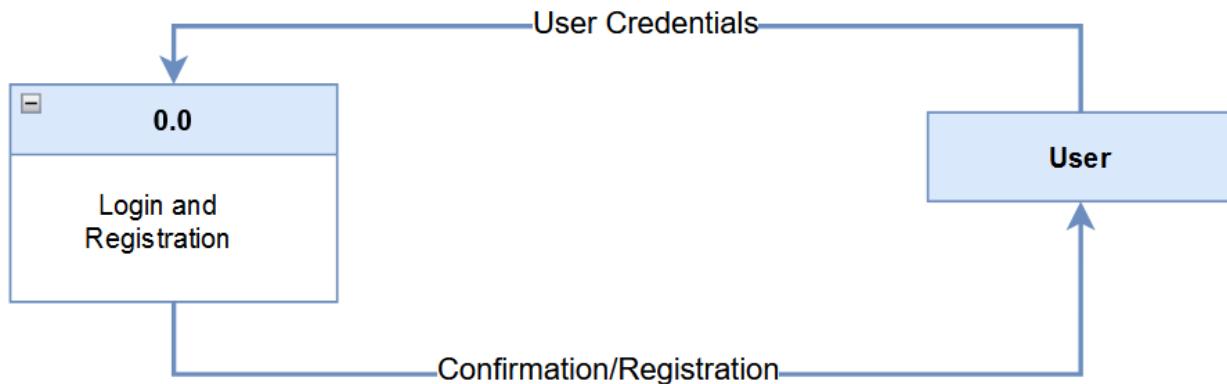


Figure 37 Level 0 DFD for login/registration

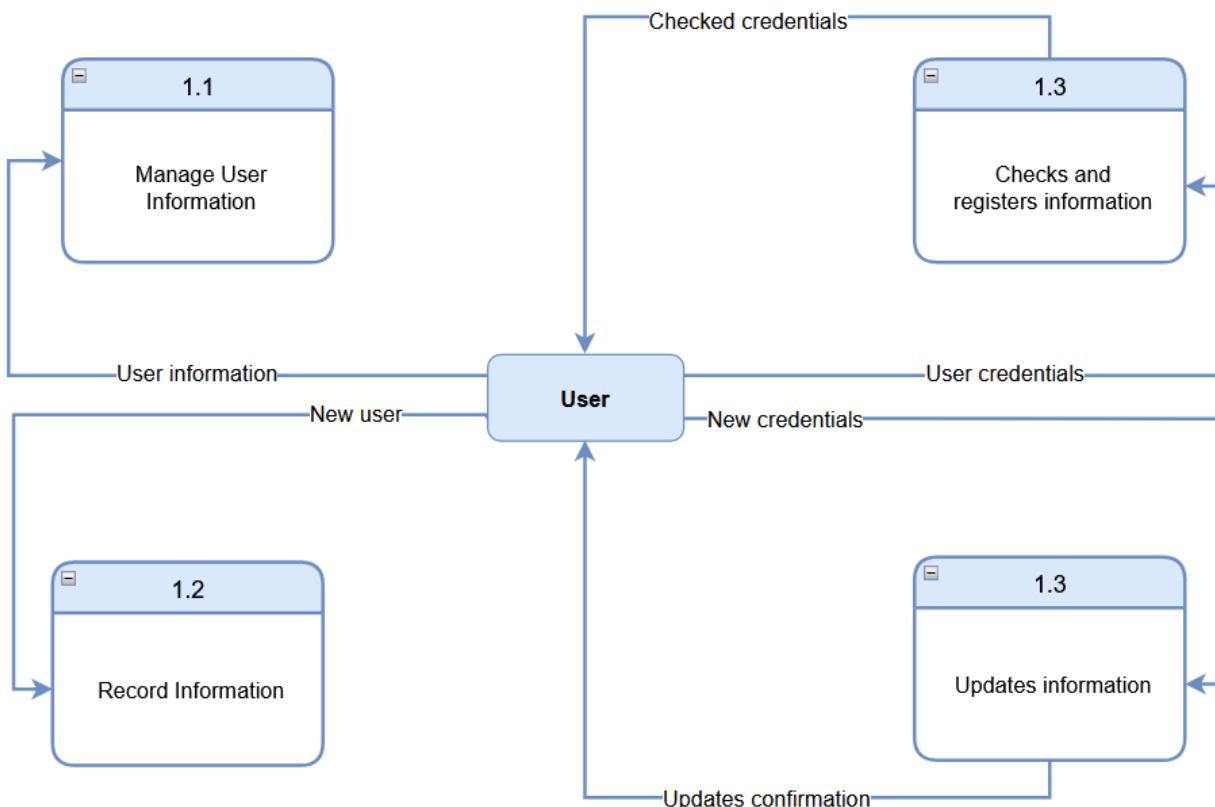


Figure 38 Level 1 DFD for login/registration

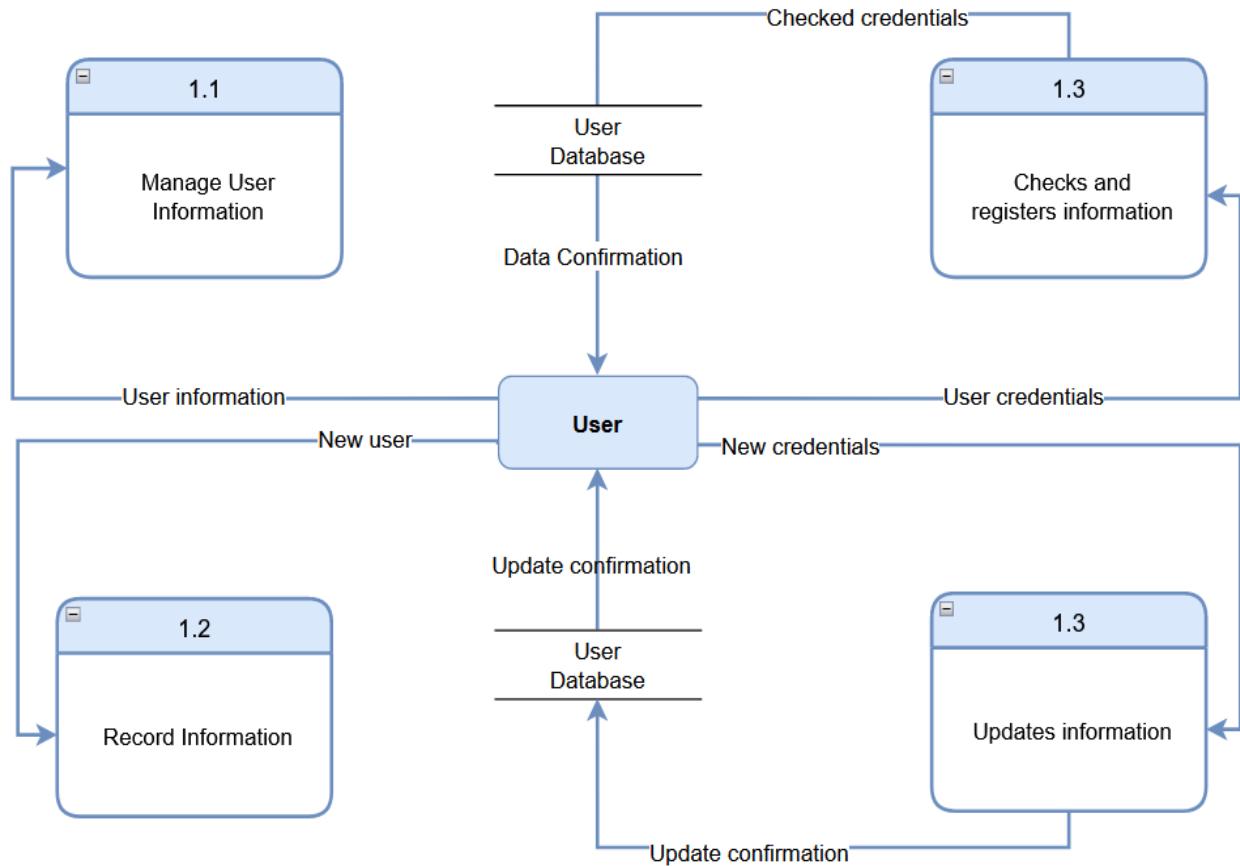


Figure 39 Level 2 DFD for login/registration

### 3.7 Implementation following Scrum

#### 3.7.1 Sprint 1: 1<sup>st</sup> and 2<sup>nd</sup> Increment

##### Highlights of sprint 1

- i. The initial user interface or tiny application using NLTK library is developed for demo showcase showing small glance of how the system will look like.

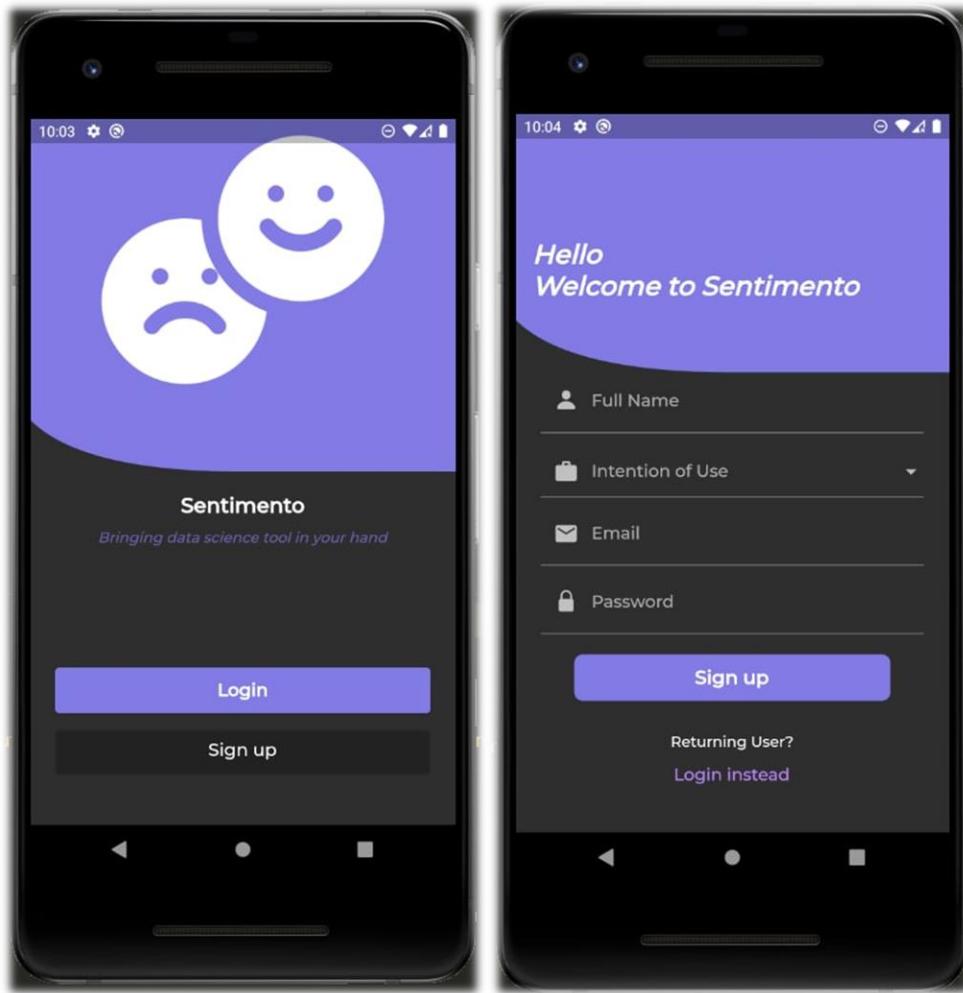


Figure 40 Initial user authentication interface

- ii. Faced memory allocation error while count vectorization of sentences and decided to remove 'not' as stop words from sentences to fix issue but this created another big issue of change of meaning of a sentence.

The screenshot shows a code editor with two files: `learnNB.py` and `functions.py`. The `learnNB.py` file contains Python code for data preprocessing and model training. A red arrow points to the line where a sentence is added to the test texts: `'You are not a good person'] # X_test`. Below the code editor is a terminal window showing the execution of the script. The terminal output indicates that the NLTK stopwords package is being downloaded and updated. It shows the command `"d:/Islington College/Year3/Semester 5/fyp_class/developmentwork/devweek7/learnNB.py"` and the resulting output of the script execution.

```

learnNB.py U ● functions.py U
learnNB.py > ...
4   # This file uses NLTK's corpus module to import stop words
5   # import nltk
6   # nltk.download('stopwords')
7   from functions import getCleanData
8
9   training_data = pd.read_csv('train.csv').values
10  training_texts = [] # X_train
11  training_labels = [] # y_train
12  clean_test_texts = []
13  test_texts = [
14      'I clapped because it's finished, not because I like it.',
15      'Life is good, you should get one',
16      → 'You are not a good person'] # X_test
17      # while filtering stopwords]
18
19  for each in training_data:
20      # first cleaning(preprocessing) the each text and appending
21      training_texts.append(getCleanData(data=each[0]))
22      training_labels.append(each[1])
23
24  for each in test_texts:
25      clean_test_texts.append(getCleanData(data=each))
26
27  print(clean_test_texts)
28

```

```

Python x
(c) Microsoft Corporation. All rights reserved.

D:\Islington College\Year3\Semester 5\fyp_class\developmentwork\devweek7>"c:/Program Files/Python38/python.exe" "d:/Islington College/Year3/Semester 5/fyp_classes/developmentwork/devweek7/learnNB.py"
[nltk_data] Downloading package stopwords to
[nltk_data]   C:/Users/MSI/AppData/Roaming/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['i clap finish i like', 'life good get one', 'you good person']

D:\Islington College\Year3\Semester 5\fyp_class\developmentwork\devweek7>"c:/Program Files/Python38/python.exe" "d:/Islington College/Year3/Semester 5/fyp_classes/developmentwork/devweek7/learnNB.py"
[nltk_data] Downloading package stopwords to
[nltk_data]   C:/Users/MSI/AppData/Roaming/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['i clap finish i like', 'life good get one', 'you good person']

D:\Islington College\Year3\Semester 5\fyp_class\developmentwork\devweek7>[]

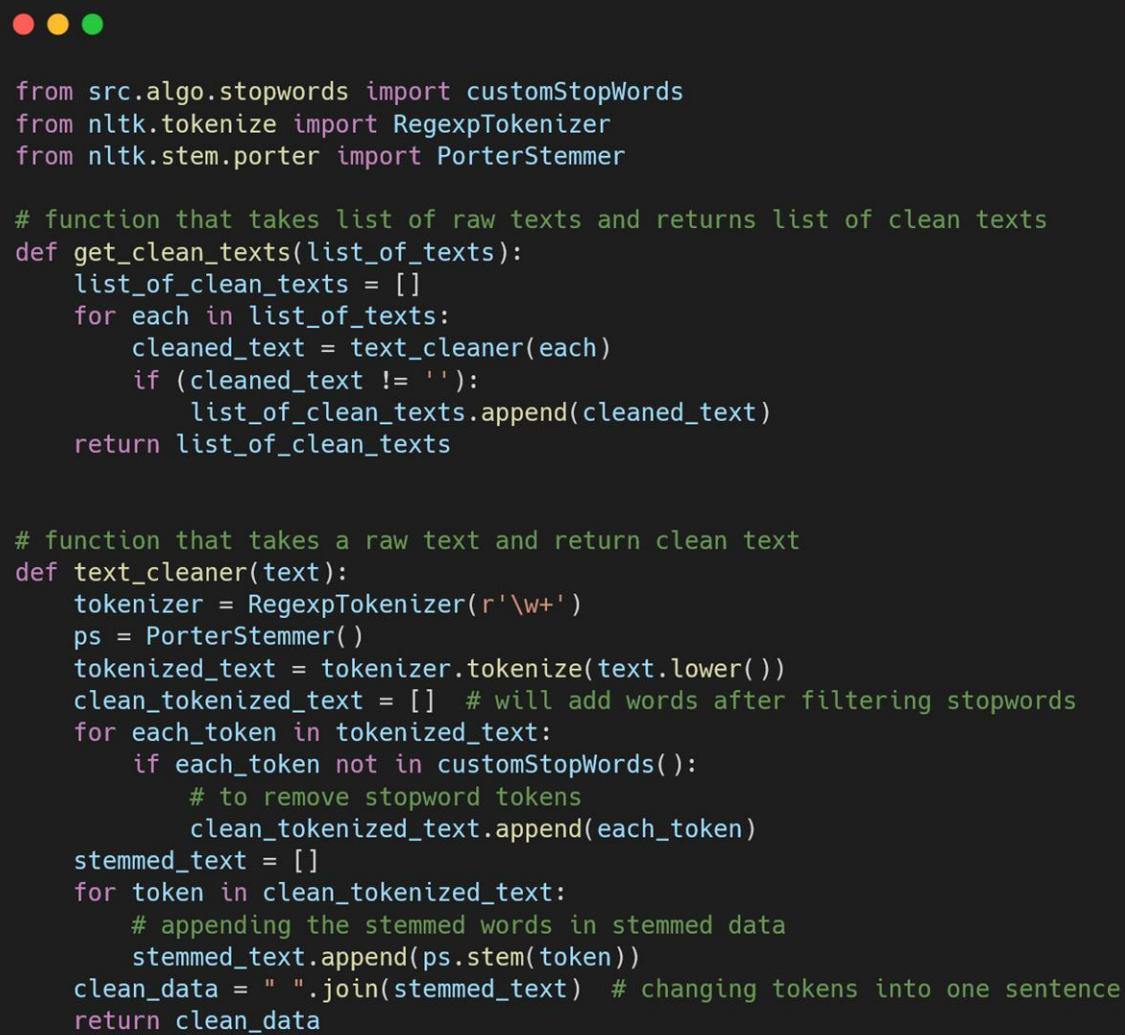
```

Figure 41 Change in meaning of a sentence while removing stop words

### 3.7.2 Sprint 2: 3<sup>rd</sup>, 4<sup>th</sup>, 5<sup>th</sup> Increment

#### Highlights of sprint 2:

- i. Data cleaning or pre-processing



```

from src.algo.stopwords import customStopWords
from nltk.tokenize import RegexpTokenizer
from nltk.stem.porter import PorterStemmer

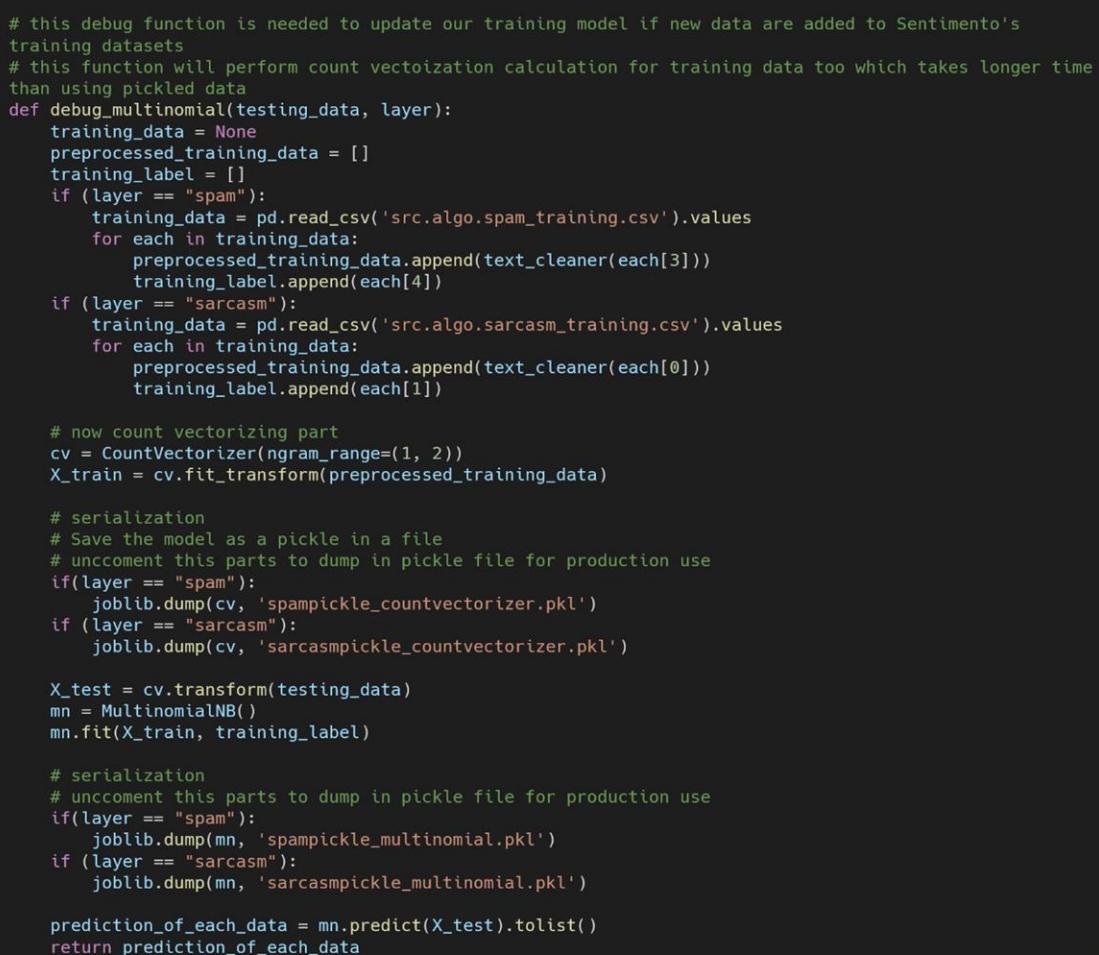
# function that takes list of raw texts and returns list of clean texts
def get_clean_texts(list_of_texts):
    list_of_clean_texts = []
    for each in list_of_texts:
        cleaned_text = text_cleaner(each)
        if (cleaned_text != ''):
            list_of_clean_texts.append(cleaned_text)
    return list_of_clean_texts

# function that takes a raw text and return clean text
def text_cleaner(text):
    tokenizer = RegexpTokenizer(r'\w+')
    ps = PorterStemmer()
    tokenized_text = tokenizer.tokenize(text.lower())
    clean_tokenized_text = [] # will add words after filtering stopwords
    for each_token in tokenized_text:
        if each_token not in customStopWords():
            # to remove stopword tokens
            clean_tokenized_text.append(each_token)
    stemmed_text = []
    for token in clean_tokenized_text:
        # appending the stemmed words in stemmed data
        stemmed_text.append(ps.stem(token))
    clean_data = " ".join(stemmed_text) # changing tokens into one sentence
    return clean_data

```

Figure 42 Function to clean sentence

ii. Multinomial Naive Bayes module for both debug and production level



```

# this debug function is needed to update our training model if new data are added to Sentimento's
# training datasets
# this function will perform count vectoization calculation for training data too which takes longer time
# than using pickled data
def debug_multinomial(testing_data, layer):
    training_data = None
    preprocessed_training_data = []
    training_label = []
    if (layer == "spam"):
        training_data = pd.read_csv('src.algo.spam_training.csv').values
        for each in training_data:
            preprocessed_training_data.append(text_cleaner(each[3]))
            training_label.append(each[4])
    if (layer == "sarcasm"):
        training_data = pd.read_csv('src.algo.sarcasm_training.csv').values
        for each in training_data:
            preprocessed_training_data.append(text_cleaner(each[0]))
            training_label.append(each[1])

    # now count vectorizing part
    cv = CountVectorizer(ngram_range=(1, 2))
    X_train = cv.fit_transform(preprocessed_training_data)

    # serialization
    # Save the model as a pickle in a file
    # unccomment this parts to dump in pickle file for production use
    if(layer == "spam"):
        joblib.dump(cv, 'spampickle_countvectorizer.pkl')
    if (layer == "sarcasm"):
        joblib.dump(cv, 'sarcasmpickle_countvectorizer.pkl')

    X_test = cv.transform(testing_data)
    mn = MultinomialNB()
    mn.fit(X_train, training_label)

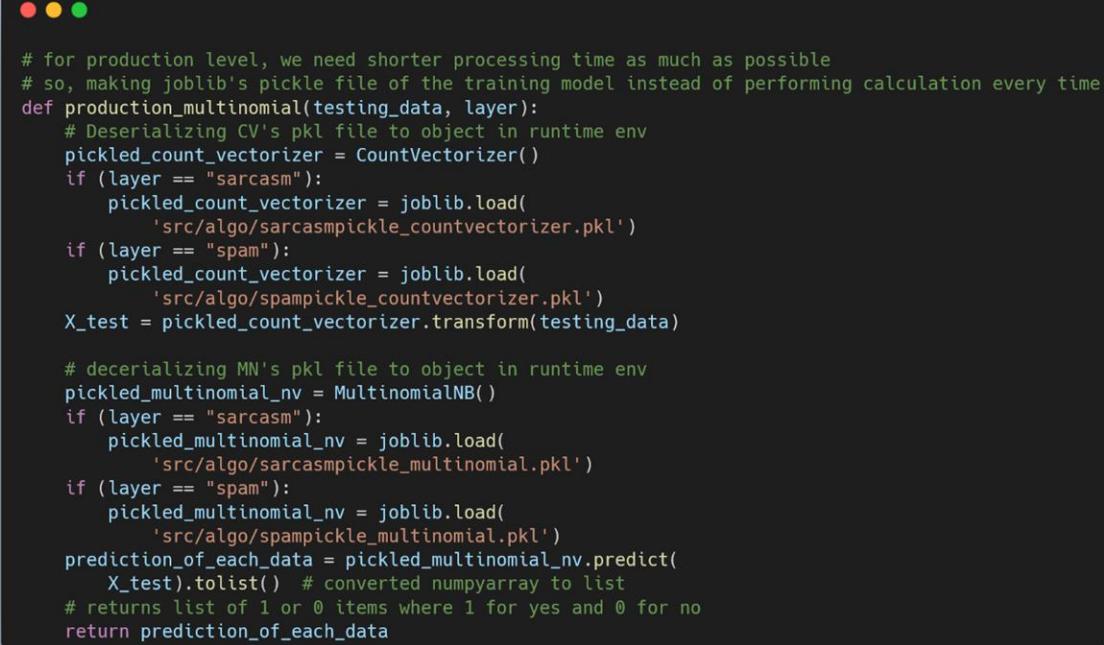
    # serialization
    # unccomment this parts to dump in pickle file for production use
    if(layer == "spam"):
        joblib.dump(mn, 'spampickle_multinomial.pkl')
    if (layer == "sarcasm"):
        joblib.dump(mn, 'sarcasmpickle_multinomial.pkl')

    prediction_of_each_data = mn.predict(X_test).tolist()
    return prediction_of_each_data

```

Figure 43 Function to make updates in serialized training model

In the above code, ‘fit\_transform’ method is used for training data whereas ‘transform’ method is used for testing data. This is done for removing exclusion biasness from the system towards particular features. Exclusion biasness occurs if features of training data are not evaluated just because it does not exist in testing data. ‘fit\_transform’ is used for training data to learn mean and variance of the features from training data. The mean and variance of the testing data needs to be new for our model so we use ‘transform’ for it (Khanna, 2020).



```

# for production level, we need shorter processing time as much as possible
# so, making joblib's pickle file of the training model instead of performing calculation every time
def production_multinomial(testing_data, layer):
    # Deserializing CV's pkl file to object in runtime env
    pickled_count_vectorizer = CountVectorizer()
    if (layer == "sarcasm"):
        pickled_count_vectorizer = joblib.load(
            'src/algo/sarcasm_pickle_countvectorizer.pkl')
    if (layer == "spam"):
        pickled_count_vectorizer = joblib.load(
            'src/algo/spam_pickle_countvectorizer.pkl')
    X_test = pickled_count_vectorizer.transform(testing_data)

    # decerialize MN's pkl file to object in runtime env
    pickled_multinomial_nv = MultinomialNB()
    if (layer == "sarcasm"):
        pickled_multinomial_nv = joblib.load(
            'src/algo/sarcasm_pickle_multinomial.pkl')
    if (layer == "spam"):
        pickled_multinomial_nv = joblib.load(
            'src/algo/spam_pickle_multinomial.pkl')
    prediction_of_each_data = pickled_multinomial_nv.predict(
        X_test).tolist() # converted numpyarray to list
    # returns list of 1 or 0 items where 1 for yes and 0 for no
    return prediction_of_each_data

```

*Figure 44 Function to predict probability of each sentence to fall in 0 or 1 category*

The scikit-learn library has been used for the Multinomial Naïve Bayes calculated. The detailed theory is explained in the previous solution section of the report. There are two functions which does the same work but one is for debug level which uses textual training data for each run while the production ready function uses already existing trained model.

In the production level, using the textual data for training data too can cost higher memory usage. And it is not efficient to do for production. To shorten the runtime duration, once calculated object of the run time of CountVectorizer class is dumped in external file. The once calculated object of Multinomial class is also dumped in another external file. Saving a machine learning model is also known as serialization (GeeksforGeeks, 2021).

```

class Sentimento:

    # instance attribute, constructor
    def __init__(self, testing_data):
        self.testing_data = testing_data
        self.preprocessed_testing_data = get_clean_texts(testing_data)
        self.spam_detected_list = production_multinomial(
            self.preprocessed_testing_data, layer="spam")
        self.sarcasm_detected_list = production_multinomial(
            self.preprocessed_testing_data, layer="sarcasm")

    # to get total data taken to analyse
    def data_count(self):
        return len(self.preprocessed_testing_data)

    # to get total sarcasm/spam detected data's
    def layer_count(self):
        spam_count = 0
        sarcasm_count = 0
        layer_count = []
        for each in self.spam_detected_list:
            if each == 1:
                spam_count += 1
        for each in self.sarcasm_detected_list:
            if each == 1:
                sarcasm_count += 1
        layer_count.append(spam_count)
        layer_count.append(sarcasm_count)
        return layer_count # index 0 for spam count and index 1 for sarcasm count

    # to get dict containing overall polarity, positive/negative count from provided data
    def overall_polarity(self):
        # considering -1 to -0.2 as negative, -0.2 to 0.2 as neutral and 0.2 to 1 as positive
        positive_count = 0
        negative_count = 0
        neutral_count = 0
        overall_polarity = 0.00000
        sid = SentimentIntensityAnalyzer()
        polarity_result = {}
        compound_polarity = 0
        sum_of_all_polarity = 0
        for each in self.preprocessed_testing_data:
            compound_polarity = sid.polarity_scores(each)["compound"]
            sum_of_all_polarity += compound_polarity
            if (compound_polarity <= -0.2):
                negative_count += 1
            elif (compound_polarity >= 0.2):
                positive_count += 1
            else:
                neutral_count += 1
        overall_polarity = (sum_of_all_polarity / self.data_count())
        polarity_result["positive_count"] = positive_count
        polarity_result["negative_count"] = negative_count
        polarity_result["neutral_count"] = neutral_count
        polarity_result["overall_polarity"] = overall_polarity
        return polarity_result

```

Figure 45 Single class to access all algorithmic feature of system

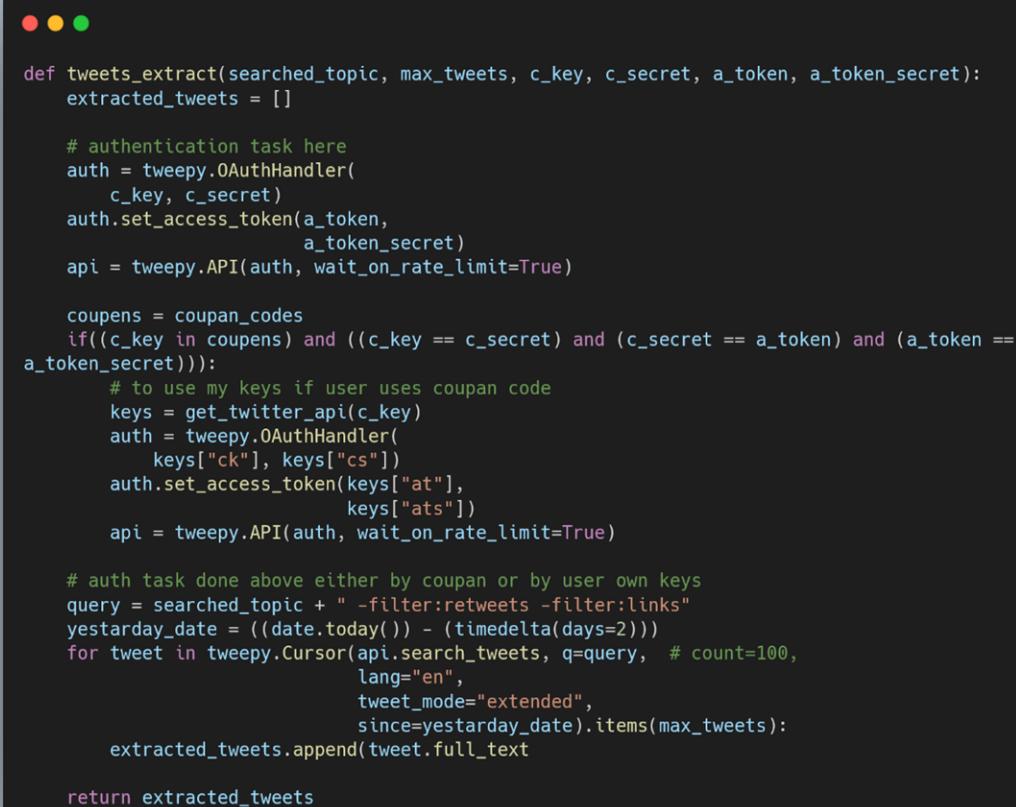
This class is used to make objects of result of sentiment analysis by the backend developed with flask and returns JSON response using the object created from this class.

### 3.7.3 Sprint 3: 6<sup>th</sup> and 7<sup>th</sup> Increment

#### Highlights of the sprint 3:

- i. YouTube and Twitter data integration

For extracting both the third party's data, their official API guidelines or documentation is studied to make these two functions.



```

def tweets_extract(searched_topic, max_tweets, c_key, c_secret, a_token, a_token_secret):
    extracted_tweets = []

    # authentication task here
    auth = tweepy.OAuthHandler(
        c_key, c_secret)
    auth.set_access_token(a_token,
                          a_token_secret)
    api = tweepy.API(auth, wait_on_rate_limit=True)

    coupens = coupan_codes
    if((c_key in coupens) and ((c_key == c_secret) and (c_secret == a_token) and (a_token == a_token_secret))):
        # to use my keys if user uses coupan code
        keys = get_twitter_api(c_key)
        auth = tweepy.OAuthHandler(
            keys["ck"], keys["cs"])
        auth.set_access_token(keys["at"],
                              keys["ats"])
        api = tweepy.API(auth, wait_on_rate_limit=True)

    # auth task done above either by coupan or by user own keys
    query = searched_topic + " -filter:retweets -filter:links"
    yesterday_date = ((date.today() - (timedelta(days=2)))
    for tweet in tweepy.Cursor(api.search_tweets, q=query, # count=100,
                               lang="en",
                               tweet_mode="extended",
                               since=yesterday_date).items(max_tweets):
        extracted_tweets.append(tweet.full_text)

    return extracted_tweets

```

Figure 46 Function to extract tweets of chosen topic

```

def youtube_comments_extract(video_id, user_given_key, max_comments):
    api_service_name = "youtube"
    api_version = "v3"
    # to get required api key from another file
    developer_key = user_given_key
    if (user_given_key in coupan_codes):
        developer_key = get_youtube_api(user_given_key)
        # this will use my api keys if user provides coupan code

    extracted_comments = [] # will append all collected comments here
    data_to_return = [] # will store comments, title, thumbnail

    youtube = googleapiclient.discovery.build(
        api_service_name, api_version, developerKey=developer_key)

    request = youtube.commentThreads().list(
        part="snippet",
        maxResults=max_comments,
        order="relevance",
        videoId=video_id,
    )
    thumbnail_title_request = youtube.videos().list(part="snippet", id=video_id)

    thumbnail_title_response = thumbnail_title_request.execute()
    response = request.execute()

    filtered_response = response["items"]
    filtered_thumbnail_response = thumbnail_title_response["items"]

    for each in filtered_response:
        extracted_comments.append(
            each["snippet"]["topLevelComment"]["snippet"]["textOriginal"])

    thumbnail = filtered_thumbnail_response[0]["snippet"]["thumbnails"]["medium"]["url"]
    video_title = filtered_thumbnail_response[0]["snippet"]["title"]

    while ((("nextPageToken" in response) & (max_comments > 100))):
        request = youtube.commentThreads().list(
            part="snippet",
            maxResults=100,
            order="relevance",
            videoId=video_id,
            pageToken=response["nextPageToken"],
        )
        response = request.execute()
        filtered_response = response["items"]

        for each in filtered_response:
            extracted_comments.append(
                each["snippet"]["topLevelComment"]["snippet"]["textOriginal"])

        if (len(extracted_comments) >= max_comments):
            break

    data_to_return.append(extracted_comments)
    data_to_return.append(thumbnail)
    data_to_return.append(video_title)
    return data_to_return

```

Figure 47 Function to extract YouTube comments of chosen video

ii. Flask setup with user authentication



```
@login_blueprint.route('/login/', methods=['POST'])
def login():
    request_data = request.get_json()
    try:
        email = request_data['email']
        password = request_data['password']

        # to notify user is trying to login
        notifyLoginTry(email, password)

        # to carry further process only if data in post requests are valid
        # first will check if any unwanted format of data is provided
        if len(password) < 6:
            return success_false(msg="Password should be at least 6chars")
        if len(email) > 80:
            return success_false(msg="Email should be less than 80chars")
        if not validate_email(email):
            return success_false(msg="Please enter a valid email address")

        # now moving to database querying if provided registration data format is correct
        trying_user = User.query.filter_by(email=email).first()

        if trying_user:
            # will check password if that user exists
            password_is_correct = check_password_hash(
                trying_user.password, password)

            if password_is_correct:
                access_token = create_access_token(
                    identity=email, expires_delta=False)
                # will notify admin that user loggedin success
                notifyLoginToAdmin(email, password)
                return {"success": "true", "msg": "Login succesful", "access_token": access_token,
                        "name": trying_user.name, "email": trying_user.email}
            # if password is wrong
            else:
                return success_false(msg="Please provide correct password")
        # if trying user does not exist
        else:
            return success_false(msg="User with this email does not exist in our system")

    except:
        return success_false()
```

Figure 48 Function to handle login request

The screenshot shows a POST request to <http://192.168.0.1/login/>. The Body tab is selected, containing the following JSON payload:

```

1 {
2   "email": "prashantghimire@gmail.com",
3   "password": "password20"
4 }

```

The response status is 200 OK, with a response body containing a JSON object:

```

1 {
2   "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdcIGMTY00TgxNDEyMSwianRpIjoiMjA5M2M1MzYtNDNiOC00ZmR1LWIzYmItNDEwMmI1Yj1jODQ2IiwidHlwZSI6ImFjY2VzcycyIsInN1YiI6InByYXNoYW50Z2hpOE8nbWFpbC5jb20iLCJuYmYiOjE2NDk4MTQzMjF9.eLZ9_t3AkrMfBNHeXC1gLJ7UM1xsJEb5mr1kmTC44hI",
3   "email": "prashantghimire@gmail.com",
4   "msg": "Login succesful",
5   "name": "Prashant Ghimire",
6   "success": "true"
7 }

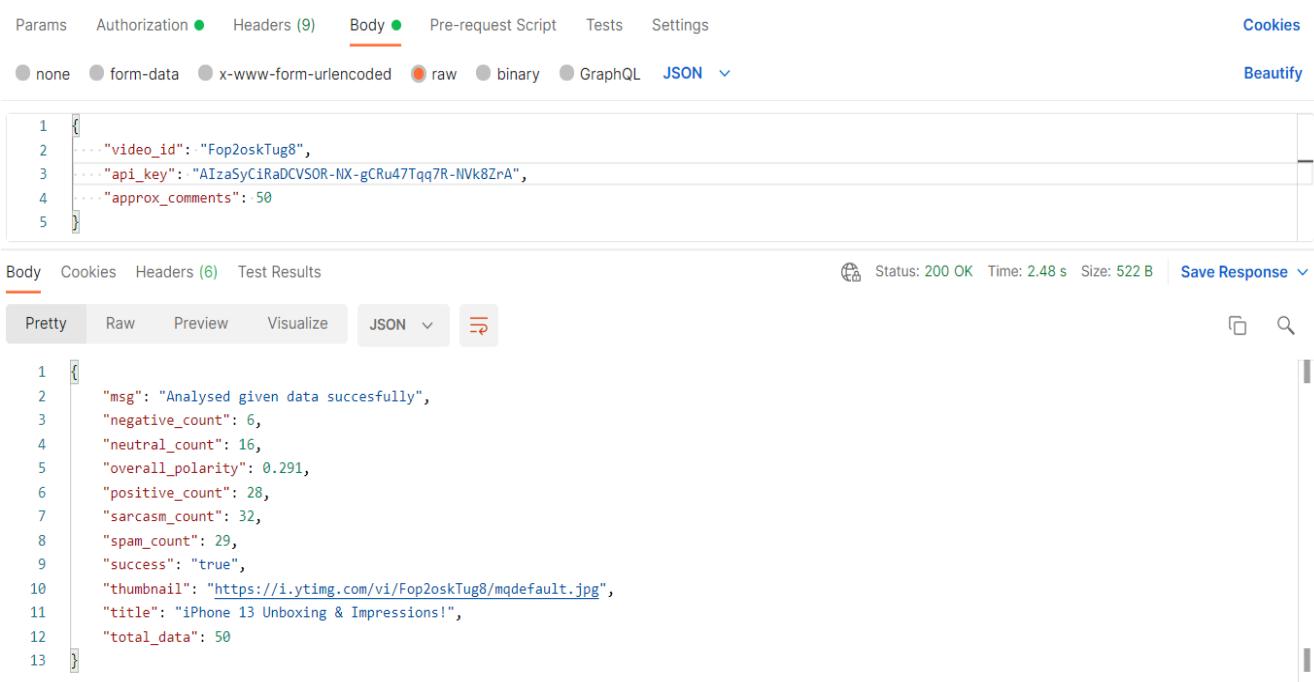
```

Figure 49 User login request task

### 3.7.4 Sprint 4: 8<sup>th</sup>, 9<sup>th</sup> and 10<sup>th</sup> Increment

#### Highlights of the sprint 4:

##### i. Sentiment analysis module integrated with backend



The screenshot shows a Postman API request and its corresponding JSON response. The request body contains the following JSON:

```

1 {
2   "video_id": "Fop2oskTug8",
3   "api_key": "AIzaSyCiRaDCV50R-NX-gCRu47Tqq7R-NVk8ZrA",
4   "approx_comments": 50
5 }

```

The response status is 200 OK, with a response time of 2.48 s and a size of 522 B. The response body is:

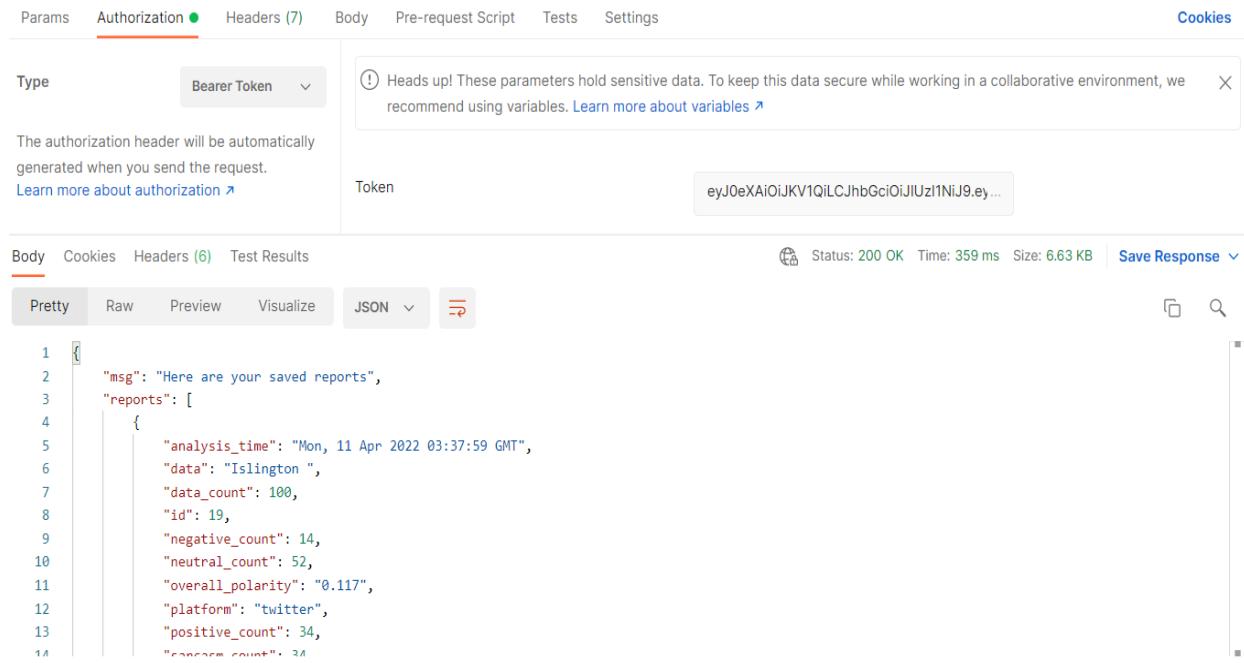
```

1 {
2   "msg": "Analysed given data successfully",
3   "negative_count": 6,
4   "neutral_count": 16,
5   "overall_polarity": 0.291,
6   "positive_count": 28,
7   "sarcasm_count": 32,
8   "spam_count": 29,
9   "success": "true",
10  "thumbnail": "https://i.ytimg.com/vi/Fop2oskTug8/mqdefault.jpg",
11  "title": "iPhone 13 Unboxing & Impressions!",
12  "total_data": 50
13 }

```

Figure 50 API handling sentiment analysis request on chosen YouTube video

## ii. Report and Vacancy saving/viewing feature



The screenshot shows a Postman API request. In the 'Authorization' tab, a 'Bearer Token' is selected, and a token is displayed as a long string of characters. The 'Body' tab shows a JSON response:

```

1  {
2   "msg": "Here are your saved reports",
3   "reports": [
4     {
5       "analysis_time": "Mon, 11 Apr 2022 03:37:59 GMT",
6       "data": "Islington",
7       "data_count": 100,
8       "id": 19,
9       "negative_count": 14,
10      "neutral_count": 52,
11      "overall_polarity": "0.117",
12      "platform": "twitter",
13      "positive_count": 34,
14      "concern_count": 24
    }
  ]
}

```

Figure 51 Viewing user's saved sentiment reports

## iii. Sign in, Signup user interfaces for front end application

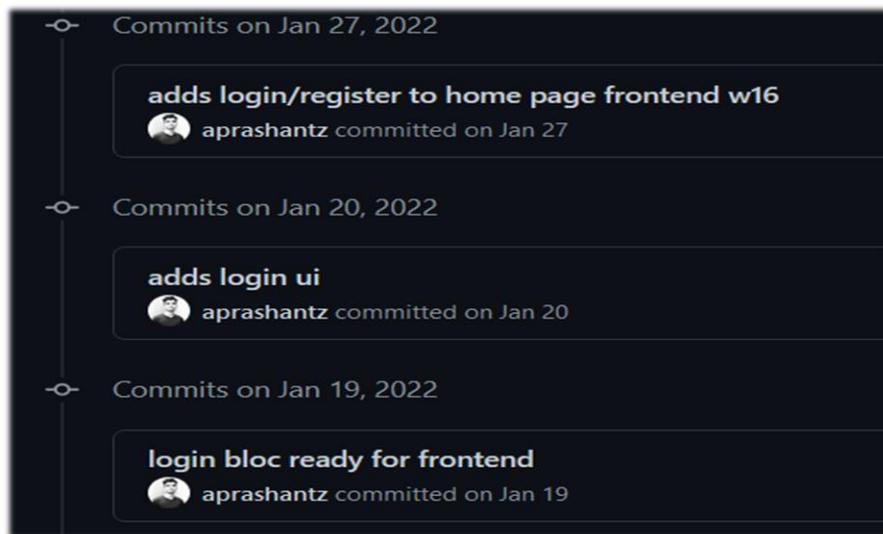


Figure 52 Git commit showing front end user authentication pages task done

### 3.7.5 Sprint 5: 11<sup>th</sup>, 12<sup>th</sup> and 13<sup>th</sup> increment

#### Highlights of sprint 5:

BLOC state management is utilized and all the essential remaining front-end tasks like profile viewing, vacancy posting, managing coupons are completed during this long sprint.

#### State Management:

Business Logic Component (BLOC) is used for state management of the built Flutter application. Building production level or qualitative applications, managing current state becomes critical. State management helps many developers to collaborate and work in a single code base seamlessly following pattern (Bloc Library, 2021).

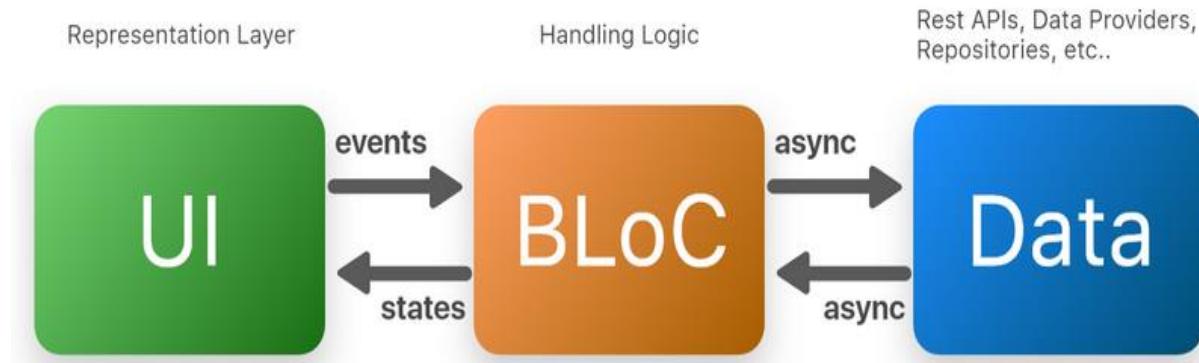


Figure 53 Bloc state management architecture (Aqeel, 2020)

For this application, multiple blocs are created for several use cases like authentication, sentiment analysis, viewing vacancies. Wherever there is presence of multiple state like data loading and loaded state, Bloc has been consumed in the application.

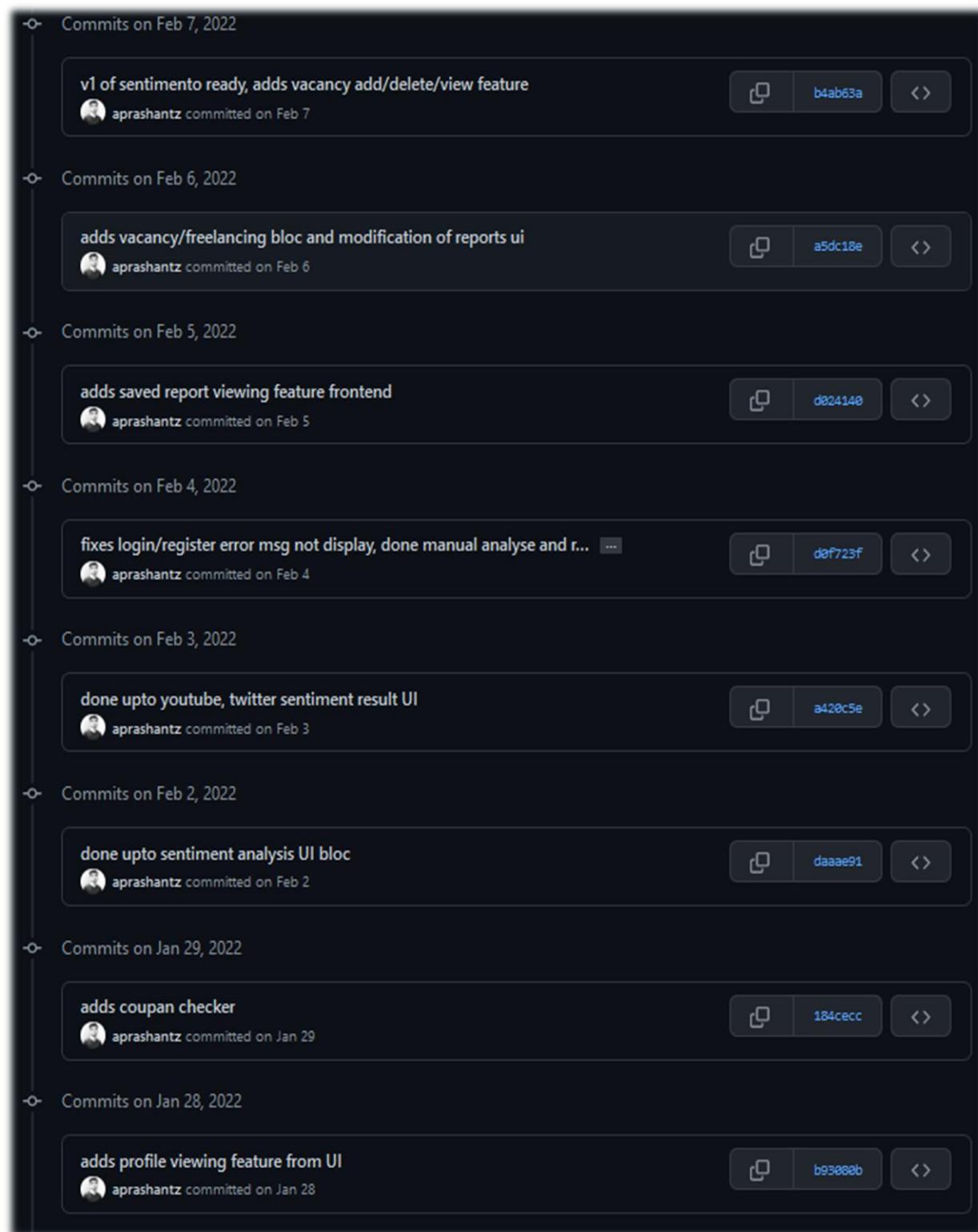


Figure 54 Git commit showing tasks of fifth sprint

### 3.7.6 Sprint 6: 14<sup>th</sup> and 15<sup>th</sup> increment

#### Highlights of sprint 6:

Normally while following Scrum in corporate or office environment, one sprint lasts around one to four weeks. For this sprint (six) of the project, this sprint lasted quite longer than a month as there were no past experiences of publishing a fully featured application to Google Play Store. The policy issues and formalities to publish this application took longer time making this sprint longer than usual Scrum's sprint.

The screenshot shows the Heroku dashboard for the 'sentimentofyp' app. At the top, it displays the URL <https://dashboard.heroku.com/apps/sentimentofyp>. The dashboard includes sections for 'Installed add-ons' (Heroku Postgres Hobby Dev), 'Dyno formation' (free dynos), and 'Collaborator activity' (prashantghi8@gmail.com). The 'Latest activity' section lists several deployment logs from Feb 25 and Feb 10, 2018, all showing success. The 'Overview' tab is selected, and other tabs include Resources, Deploy, Metrics, Activity, Access, and Settings.

Figure 55 API hosted in Heroku platform

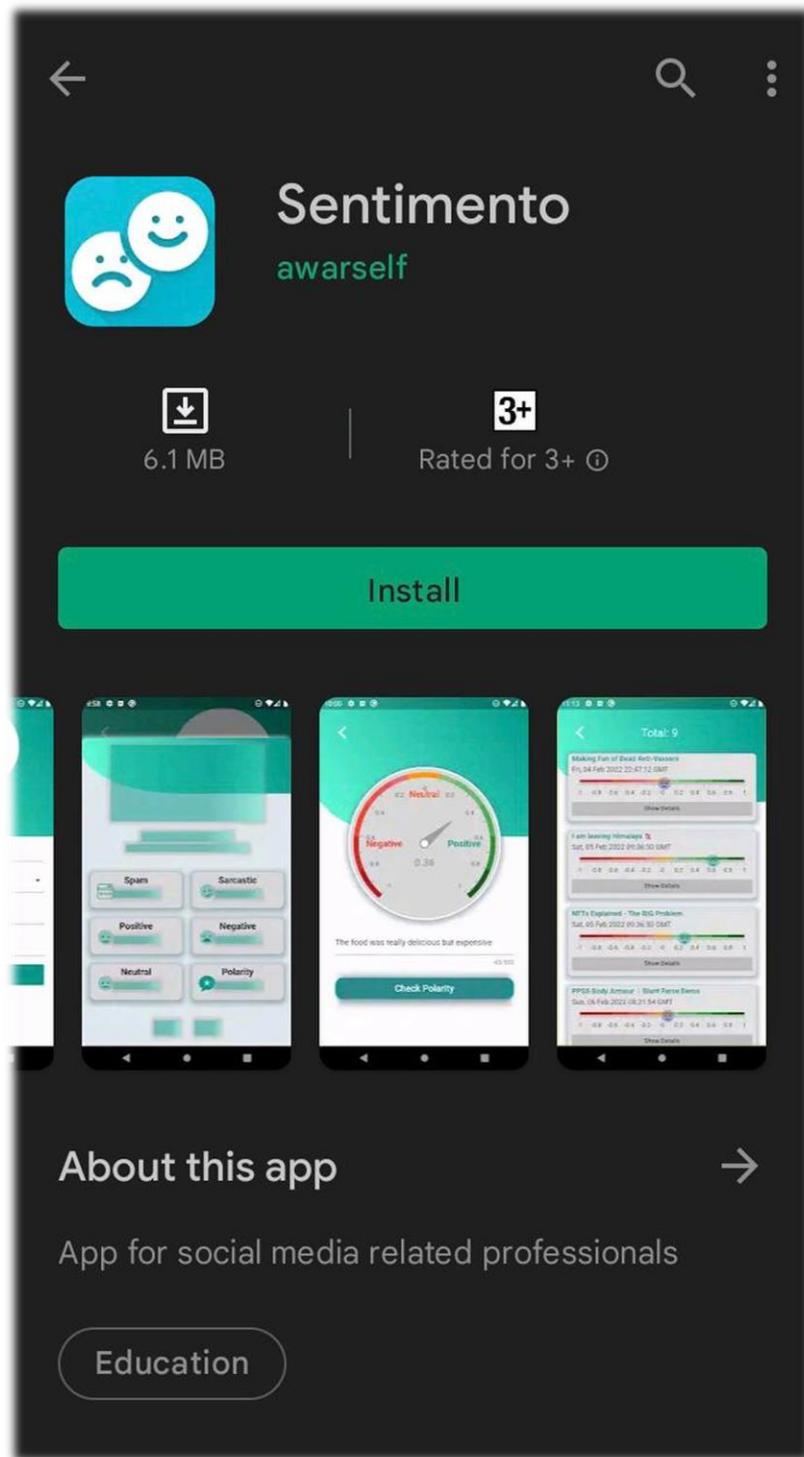


Figure 56 Application published to Google Play Store

## Chapter 4: Testing and Analysis

### 4.1 Test Plan

During the multiple sprints of development of this project ‘Sentimento’, the features built or integrated in the platform are tested properly with brief documentation of the individual test. The testing plans are listed below as following:

#### 4.1.1 Unit testing plan

It is an individual component testing method where fractional logical parts of the system are tested. Unit testing validates that each unit of the software codes performs as expected. It can be performed for a method, user interface functionalities and even for any loops or conditions that plays logical part in the system (Khorikov, 2020).

For the unit testing of this project, the individual interface components or functionalities of the built Flutter application is to be tested with precision. Apart from the application where user interacts, the built Flask API end points is also to be tested. The discrete end point APIs for sentiment analysis, vacancy without user authorization token and all built fragmentary backend logical parts are tested using Postman. Also, the access authentication for third parties is also tested in unit testing of this project.

The unit testing plan for this project are listed in the table below as follows:

#### Flutter application test

Test Id	Objective
1	To check if user can register with already used credentials.
2	To check if user can login without being registered.
3	To check if user can be registered with valid credentials.
4	To check if user can login with valid credentials.
5	To check if user can login with invalid password.
6	To check if user can view their profile.
7	To verify user cannot access sentiment analysis feature without providing tokens or coupons.
8	To test if user can access sentiment analysis feature after providing tokens or coupons.
9	To test user can perform polarity check on manual sentences.
10	To test user cannot perform sentiment analysis with invalid coupon or token.
11	To test user can view their saved sentiment reports.
12	To verify user will be shown error message if no any sentiment reports are saved.
13	To test if dashboard is responsive to screen size on multiple devices.
14	To show user can post a vacancy in the platform.
15	To show user cannot post more than five vacancies.
16	To show user redirects to phone call when presses contact button.
17	To verify user can delete their previously posted vacancy.
18	To test user can logout from the platform.
19	To show error message will be displayed if user tries to access saved reports without internet connection.
20	To show user's name and email in the interface when user is navigated from login screen to dashboard.

Table 5 Unit testing for Flutter application

Flask API test

Test Id	Objective
1	To read a user's profile without supplying a bearer token.
2	To show open vacancies can be viewed without user authentication.
3	To show user cannot delete another user's vacancy post.
4	To show user cannot perform analysis on more than 2000 comments.

*Table 6 Unit testing for Flask API*

#### 4.1.2 System testing plan

It is a comprehensive testing of the system itself. The fundamental or core features implemented in the platform are tested during system testing of the project. A complete project can be divided into multiple segments like frontend, backend. These each development works are integrated together to achieve a fully functional application (Hamilton, 2022).

System testing is about validating that all the integrated systems are merged to make a sole application perform its functionalities. Integration testing justifies the unity of team work of the built platform. Those features included in the software requirement specification document are tested in the system testing of this report.

The system testing plan for this project are listed in table below:

Test Id	Objective
1	To validate backend server is live and responsive in cloud.
2	To show backend is properly integrated with frontend.
3	To show user can perform sentiment analysis on tweets.
4	To show user can save sentiment report.

Table 7 System testing for built platform

## 4.2 Unit Testing

UT1: To check if user can register with already used credentials.

Action	Registration information was provided with already used email.
Expected Result	A toast message about invalid registration would be shown.
Actual Result	A toast message was shown stating user is already registered with this email.
Conclusion	The platform is able to prevent multiple use of same email which states the test is successful.

Table 8 Unit test 1

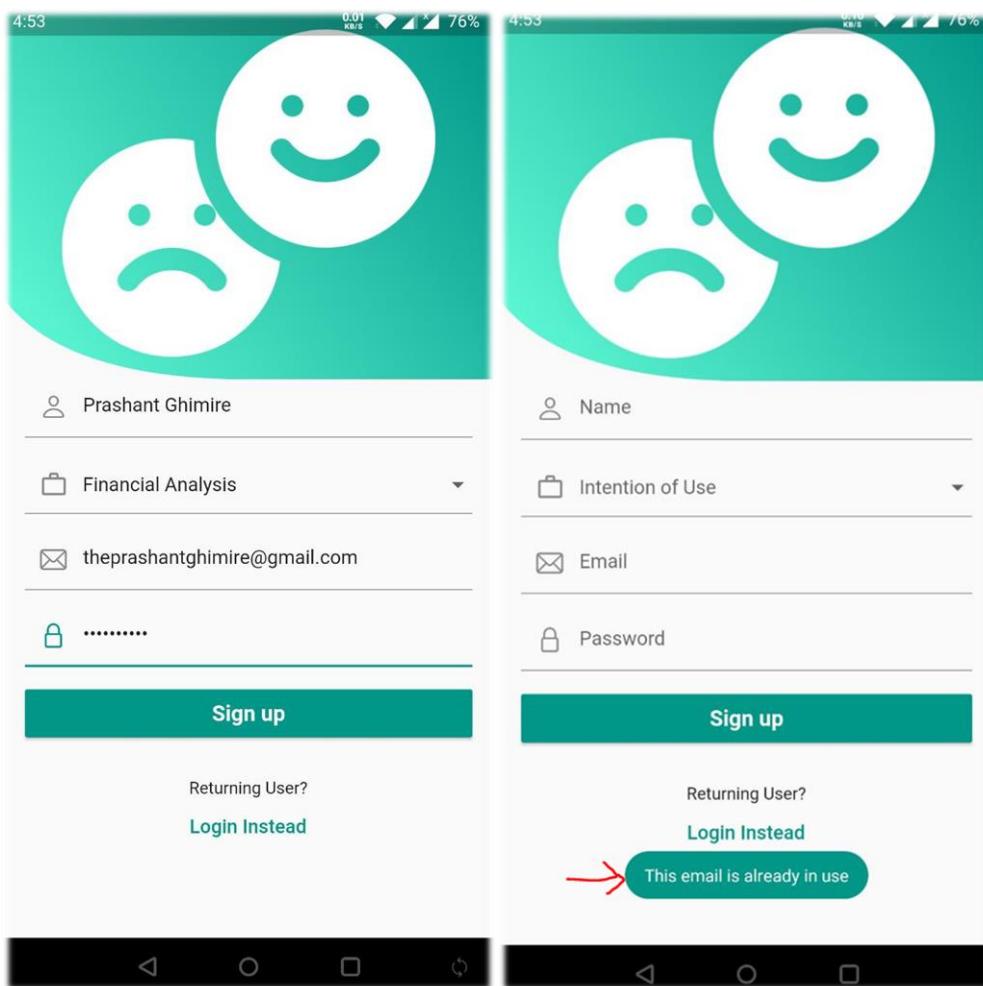


Figure 57 Trying to register with already used email

UT2: To check if user can login without being registered.

Action	Not registered email address was provided while logging in
Expected Result	A toast message about invalid login would be shown.
Actual Result	A toast message was shown stating that user with the provided email does not exist in the system.
Conclusion	Users are not allowed to log in without being registered which states that the test is successful.

Table 9 Unit test 2

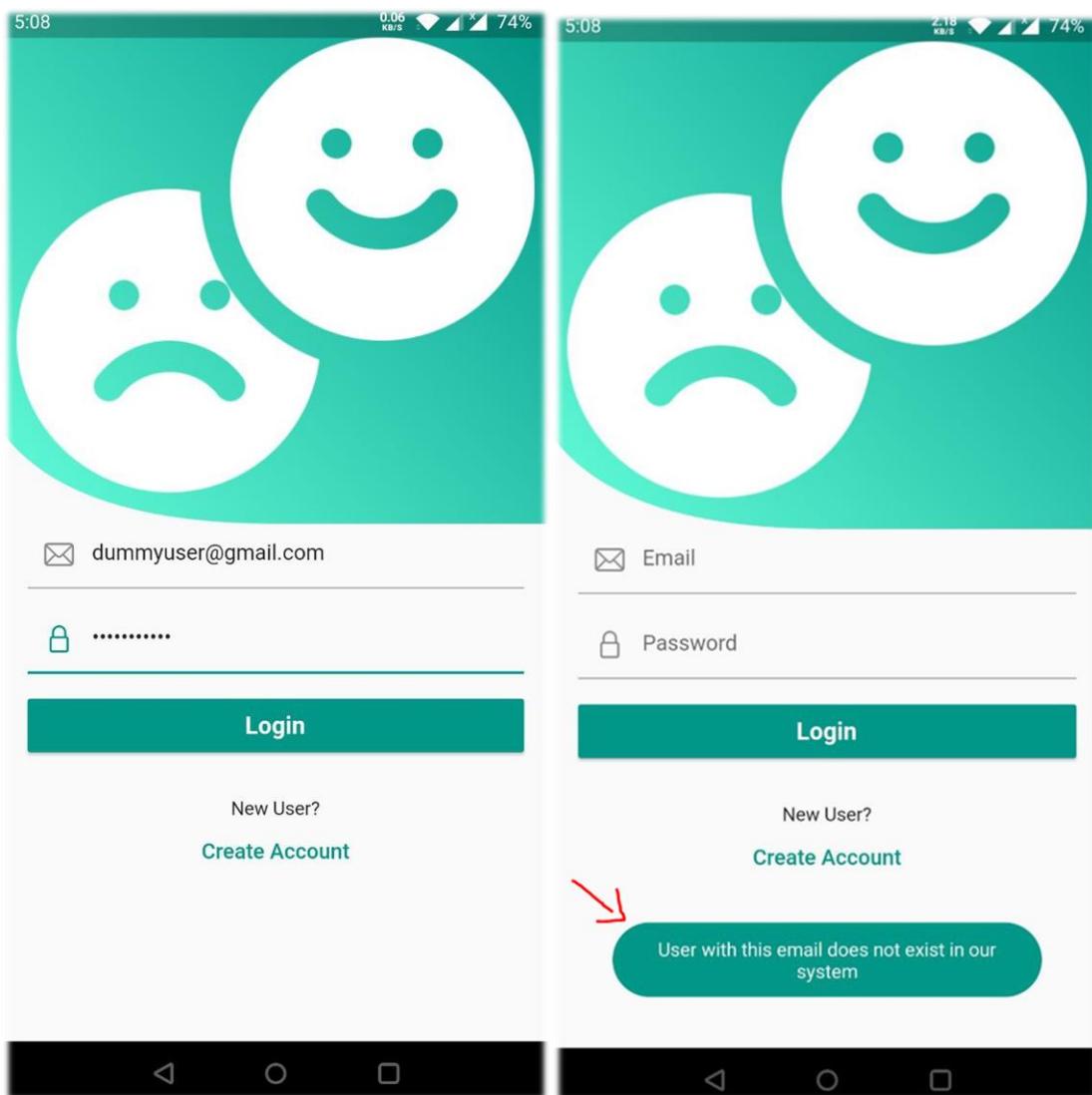


Figure 58 Trying to login without being registered

UT3: To check if user can be registered with valid credentials.

Action	A new user information was provided to register.
Expected Result	User will be navigated to dashboard.
Actual Result	After clicking register button, loading interface was shown and was navigated to dashboard
Conclusion	Users can register themselves to platform stating this test is successful.

Table 10 Unit test 3

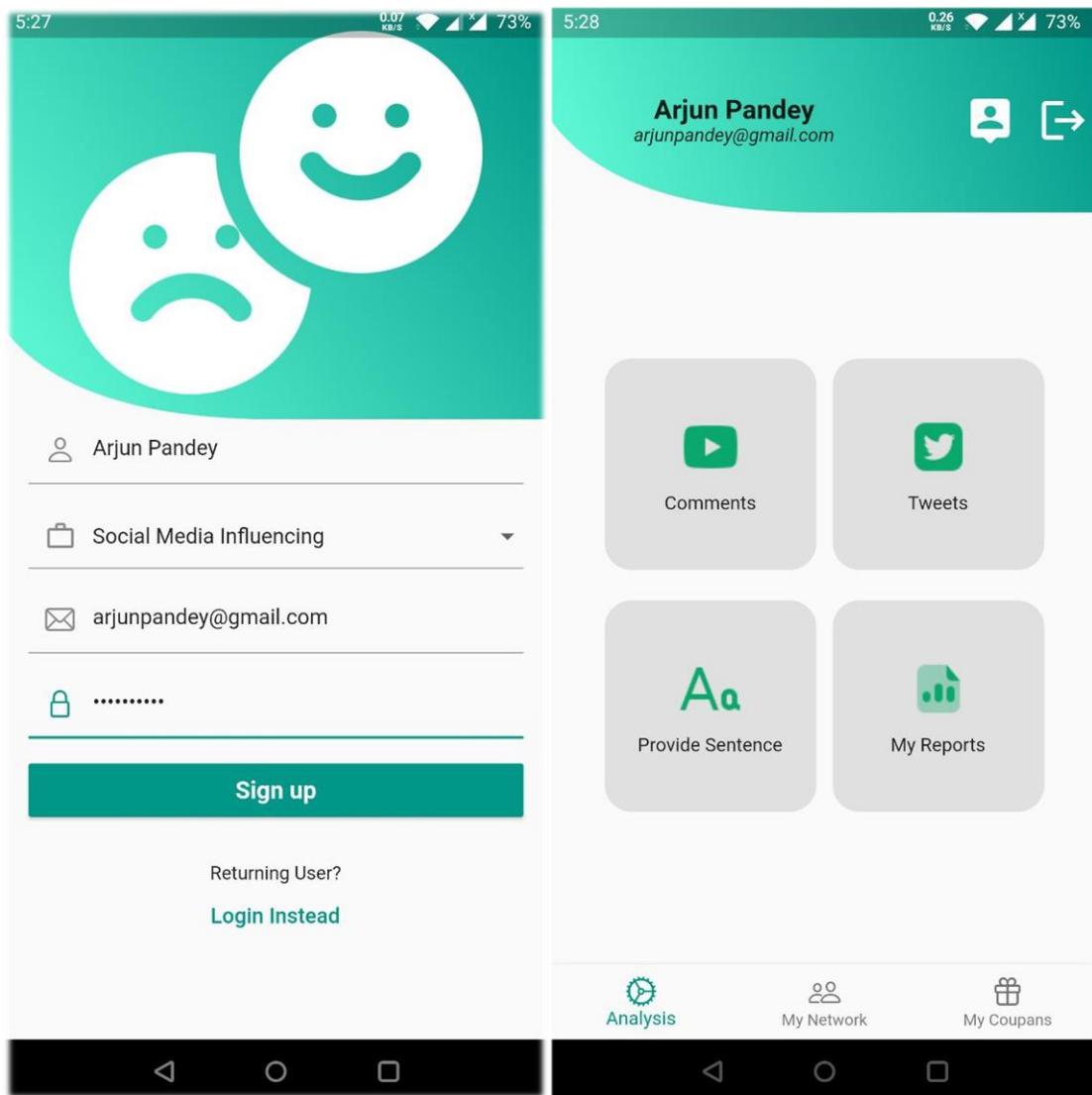


Figure 59 Registering user in the platform

UT4: To check if user can login with valid credentials.

Action	Valid email address and password was provided to login.
Expected Result	User will be navigated to dashboard.
Actual Result	After clicking login button, loading interface was shown and was navigated to dashboard.
Conclusion	User can log in to the platform which makes this test successful.

Table 11 Unit test 4

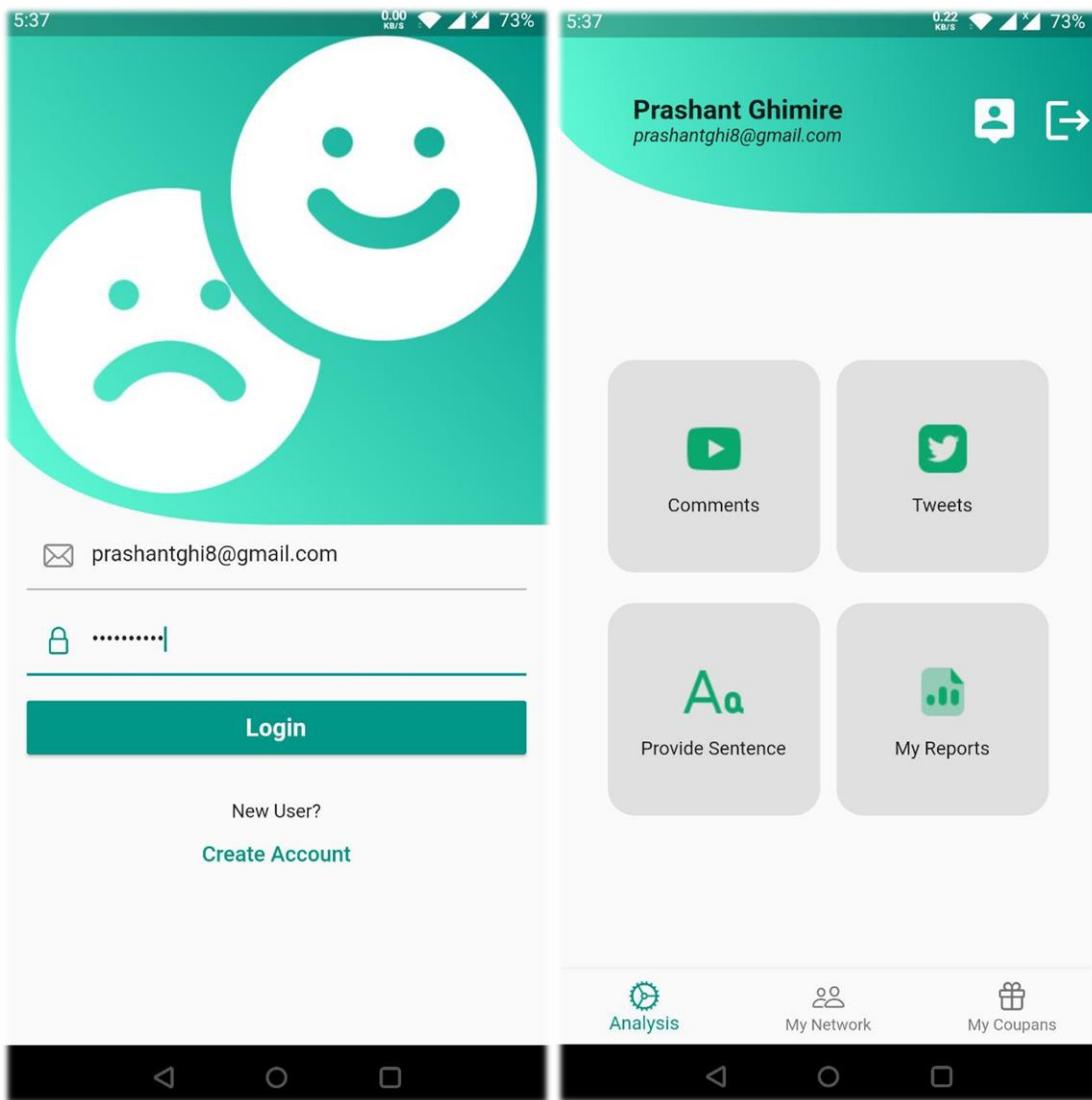


Figure 60 Logging inside the platform

UT5: To show user cannot login with invalid password.

Action	Valid email address and invalid password was provided to login.
Expected Result	User will be shown error message.
Actual Result	After clicking login button, loading interface was shown and toast message about incorrect password was shown.
Conclusion	User cannot log in to the platform with incorrect password which makes this test successful.

Table 12 Unit test 5

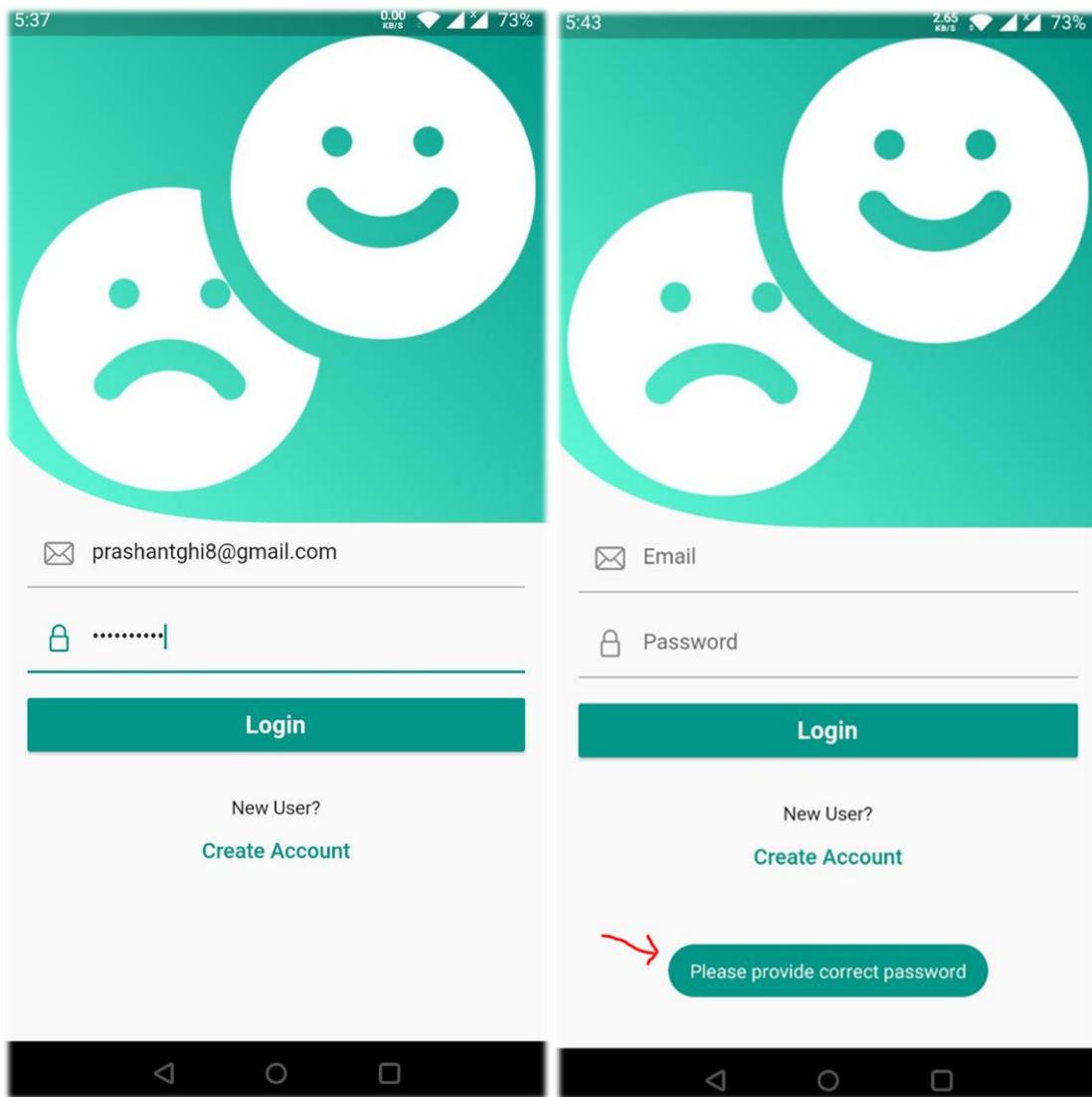


Figure 61 Trying to log in with incorrect password

UT6: To show that user can view their profile.

Action	Profile viewing icon was clicked.
Expected Result	User will be shown their profile.
Actual Result	After clicking profile button, loading interface was shown and profile information was shown
Conclusion	User can view their profile which states the test is successful.

Table 13 Unit test 6

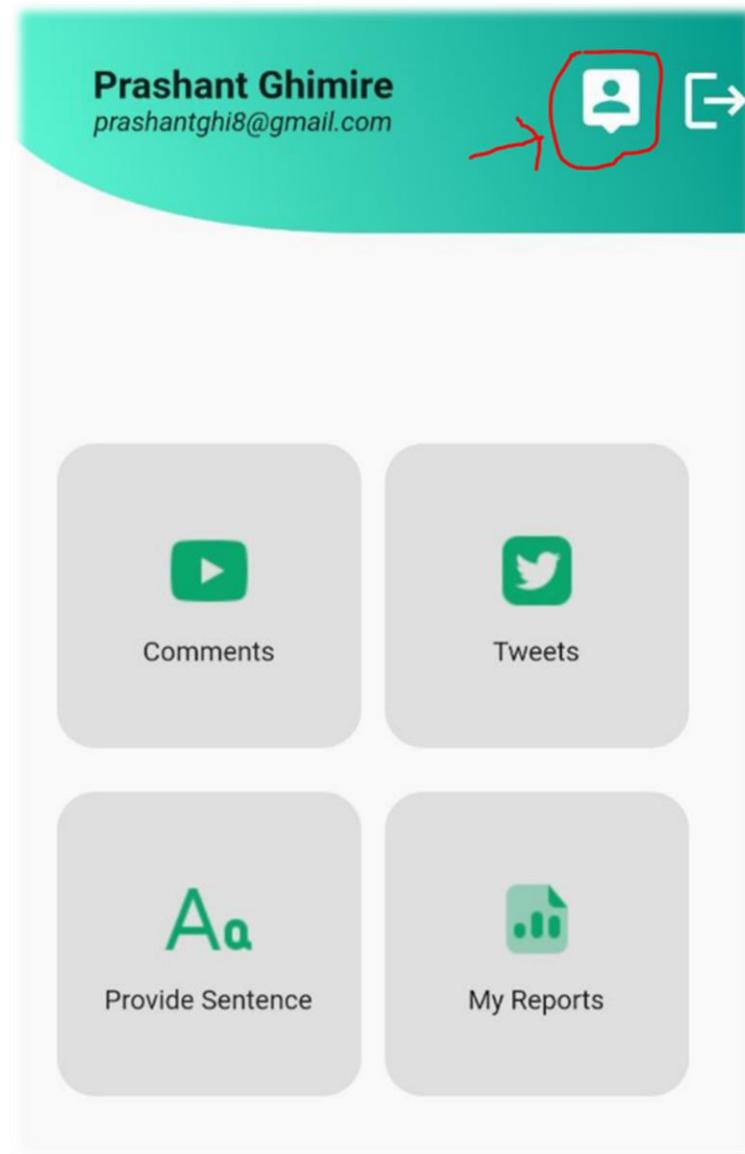


Figure 62 Showing profile icon to enter profile page

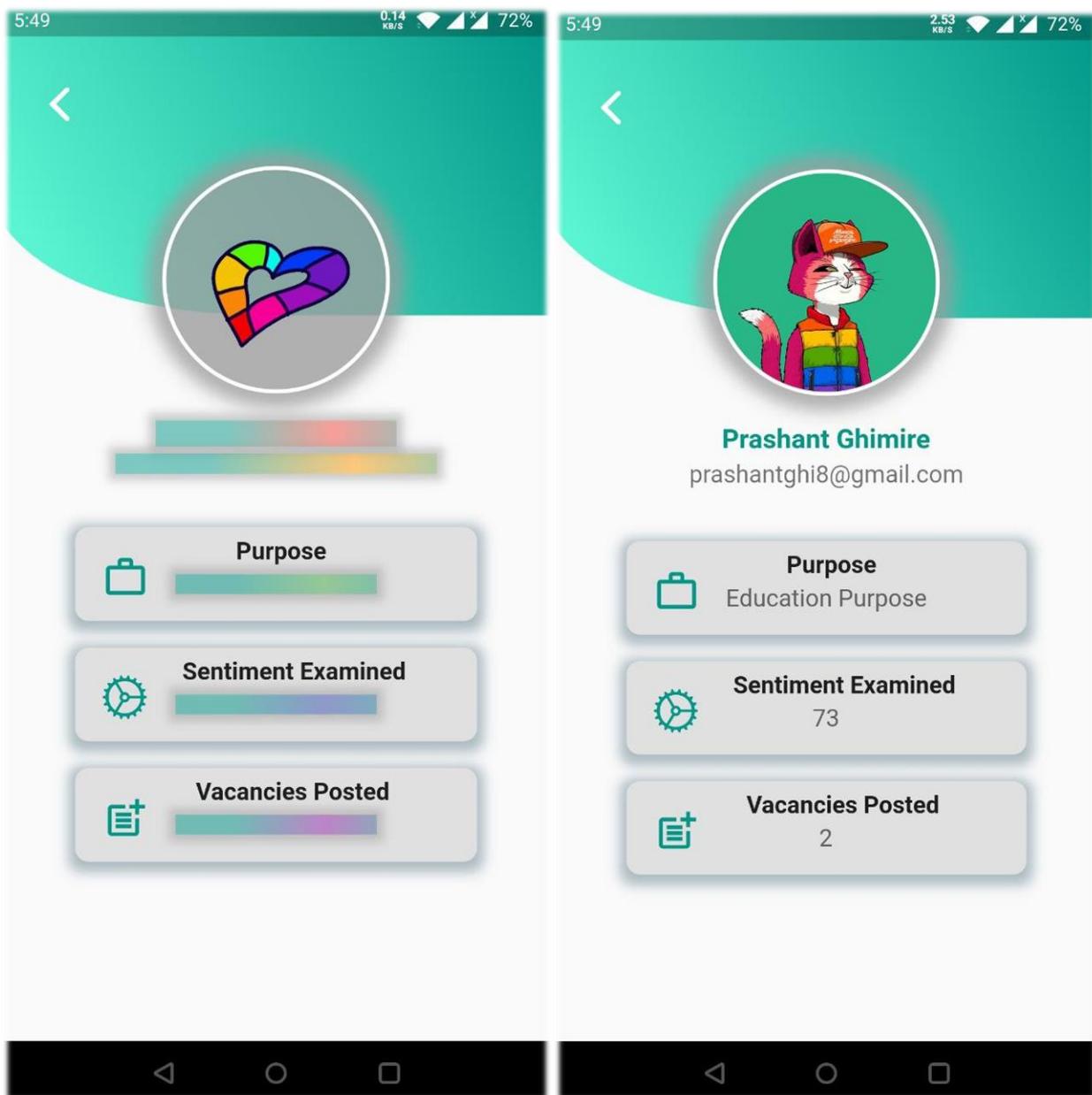


Figure 63 User navigated to profile screen after loading completes

UT7: To verify user cannot access sentiment analysis feature without providing tokens or coupons.

Action	YouTube analysis was tried to access without providing token or coupon.
Expected Result	A toast message is expected to be shown.
Actual Result	A toast message was shown stating tokens to enter first.
Conclusion	User cannot perform sentiment analysis without entering tokens which states the test is successful.

Table 14 Unit test 7

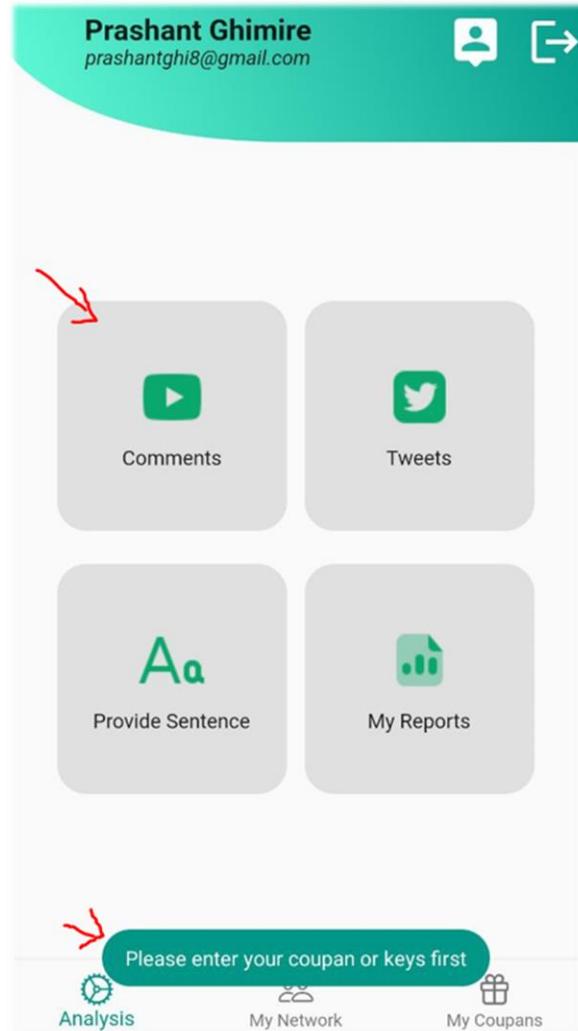


Figure 64 Trying to perform sentiment analysis without providing tokens

UT8: To show user can access sentiment analysis feature after providing tokens or coupons.

Action	Coupons were entered and YouTube comments analysis was selected, video URL and 500 numbers of comments were asked to analyse.
Expected Result	A successful sentiment analysis report would be shown.
Actual Result	Report interface with thumbnail and analysis data was shown.
Conclusion	User can perform sentiment analysis after entering tokens which states the test is successful.

Table 15 Unit test 8

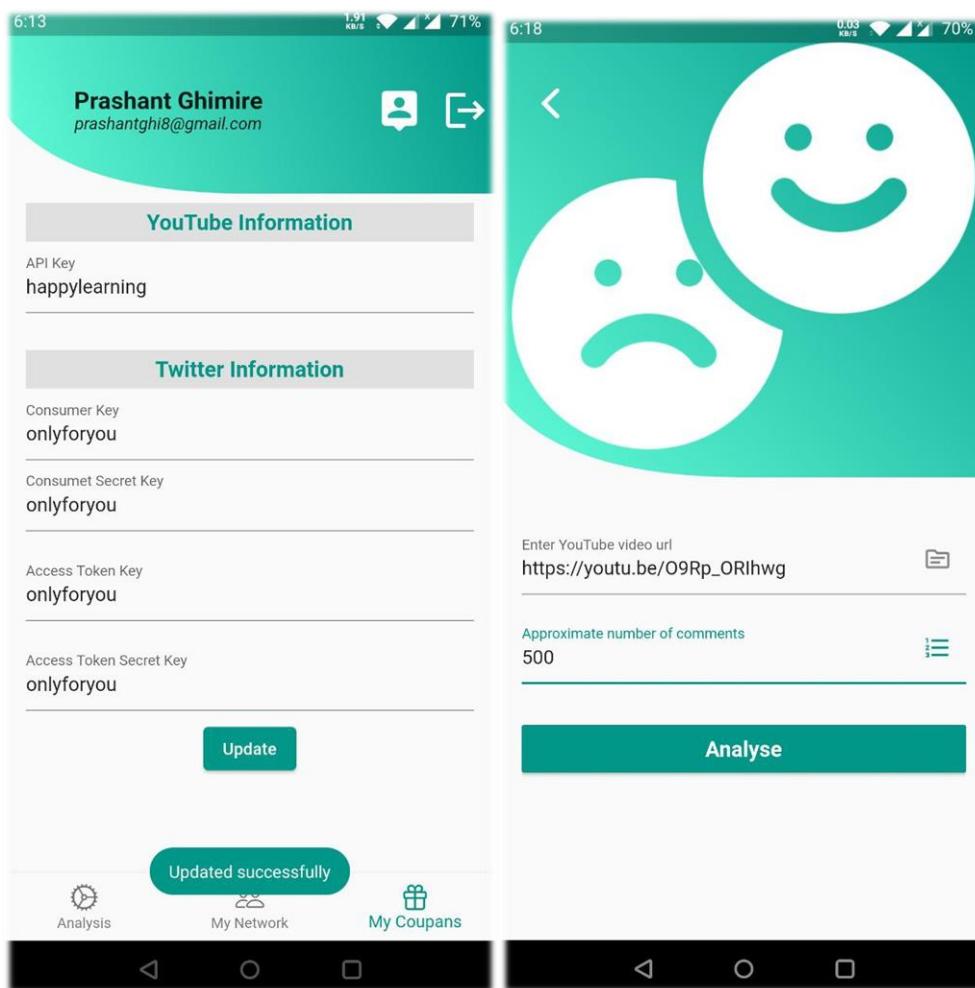


Figure 65 Entering coupons and video information

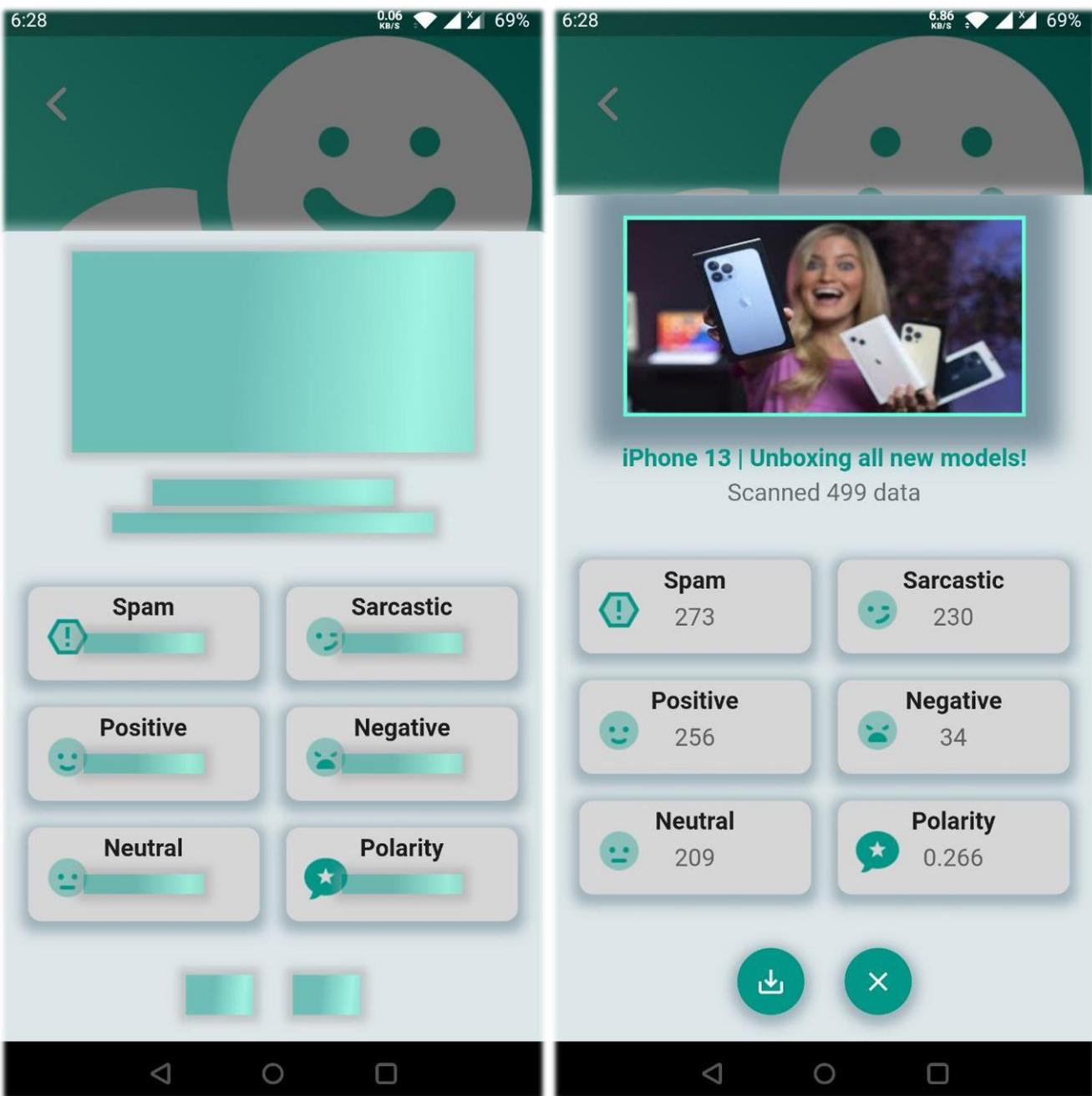


Figure 66 Showing sentiment report after loading completes

UT9: To test user can perform polarity check on manual sentences.

Action	A sentence was provided to check polarity.
Expected Result	The polarity of the sentence will be shown.
Actual Result	The indicator changed its position and polarity was shown.
Conclusion	User can perform sentiment analysis on manual sentences stating this test is successful.

Table 16 Unit test 9

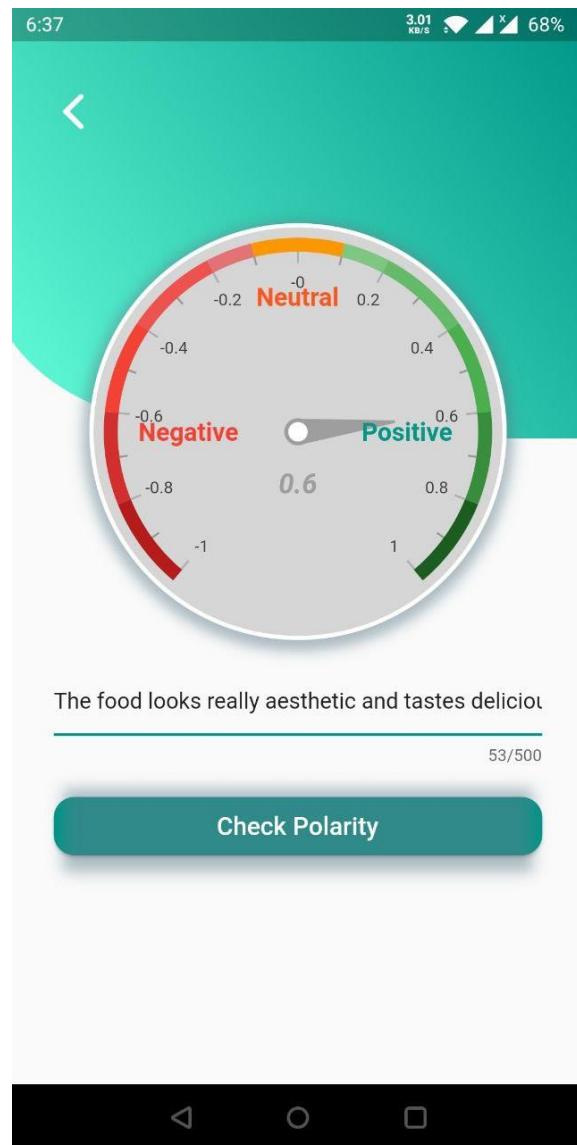


Figure 67 Polarity check of manual sentence

UT10: To show user cannot perform sentiment analysis with invalid coupon.

Action	Invalid coupon was provided to perform YouTube comments analysis.
Expected Result	A toast message would be shown.
Actual Result	A toast message about invalid request was shown.
Conclusion	User cannot perform sentiment analysis using invalid coupons stating this test is successful.

Table 17 Unit test 10



Figure 68 Providing invalid coupon

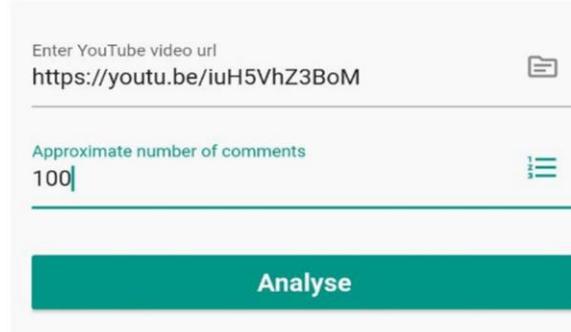


Figure 69 Trying to perform analysis using invalid coupon

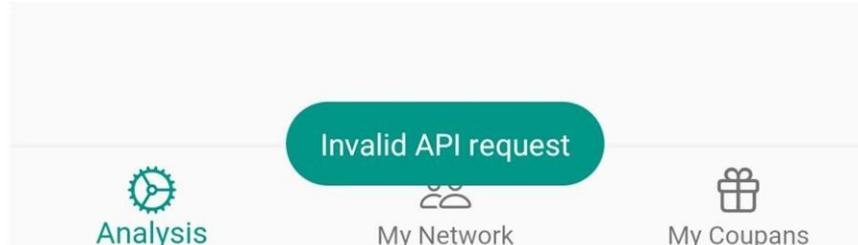


Figure 70 Toast message about invalid request

UT11: To verify that user can view their saved sentiment reports.

Action	'My Reports' button was clicked then 'Show Details' was clicked.
Expected Result	All the saved sentiment reports were expected to appear.
Actual Result	All the saved reports were shown in order.
Conclusion	User can view their saved reports stating this test is successful.

Table 18 Unit test 11

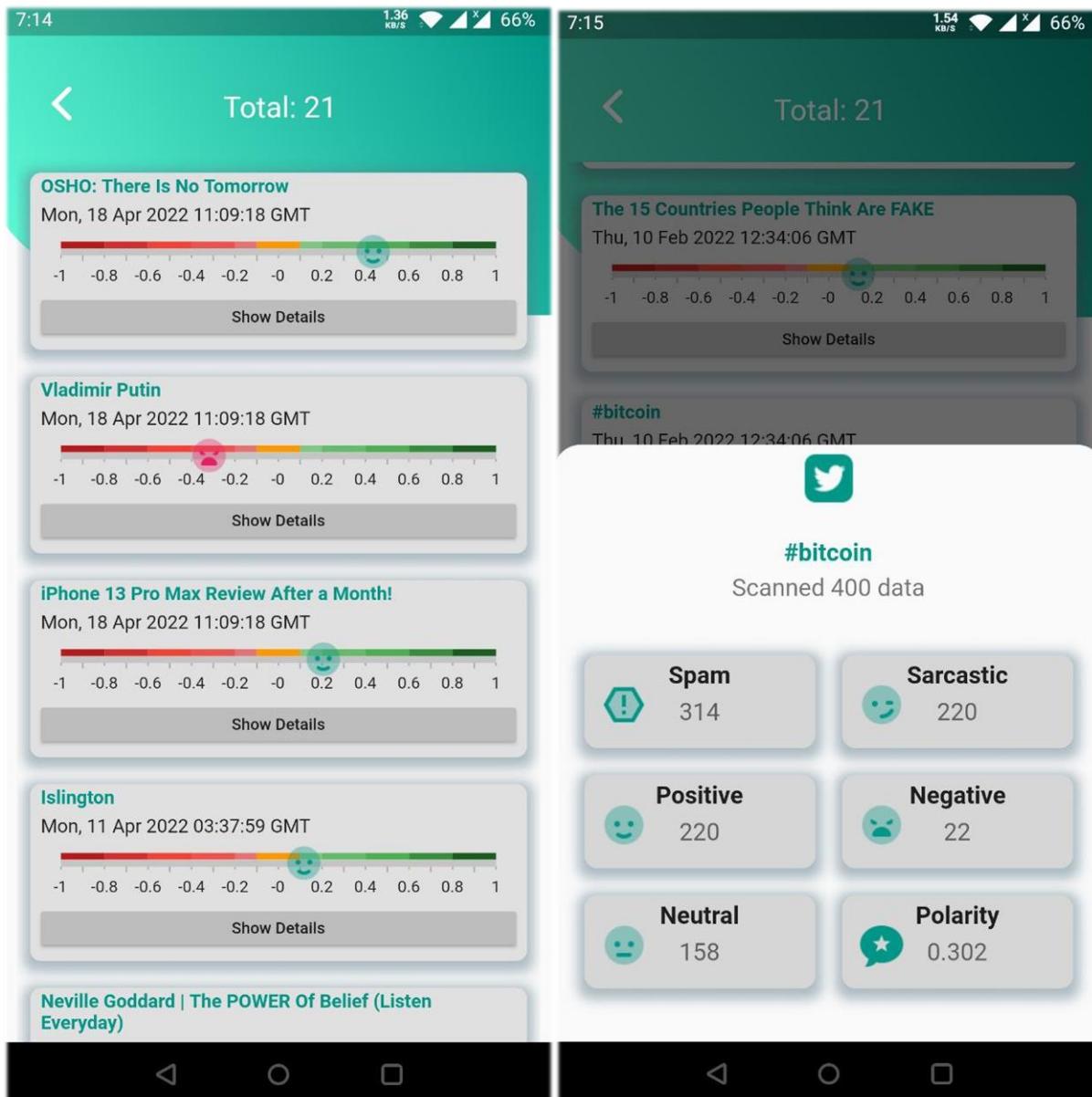


Figure 71 Saved sentiment reports of user

UT12: To verify user will be shown error message if no any sentiment reports are saved.

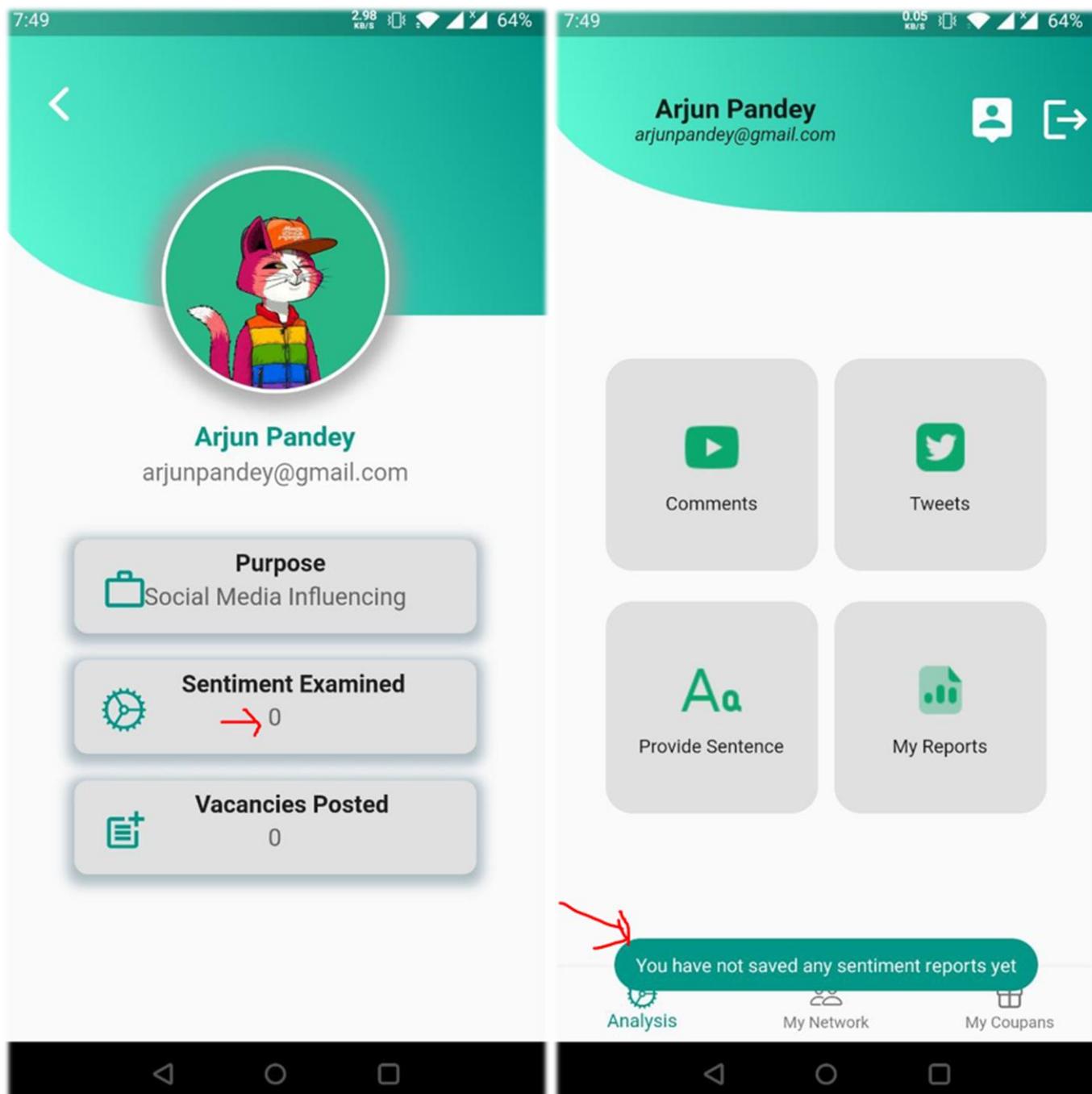


Figure 72 Trying to view reports without performing any analysis

UT13: To test if dashboard is responsive with screen size on multiple devices.

Action	Phone was rotated to landscape from dashboard screen
Expected Result	The four menus were expected to align in same line on landscape.
Actual Result	All the four menus were aligned on same line as expected.
Conclusion	The dashboard menus align according to screen size which states test is successful.

Table 19 Unit test 13

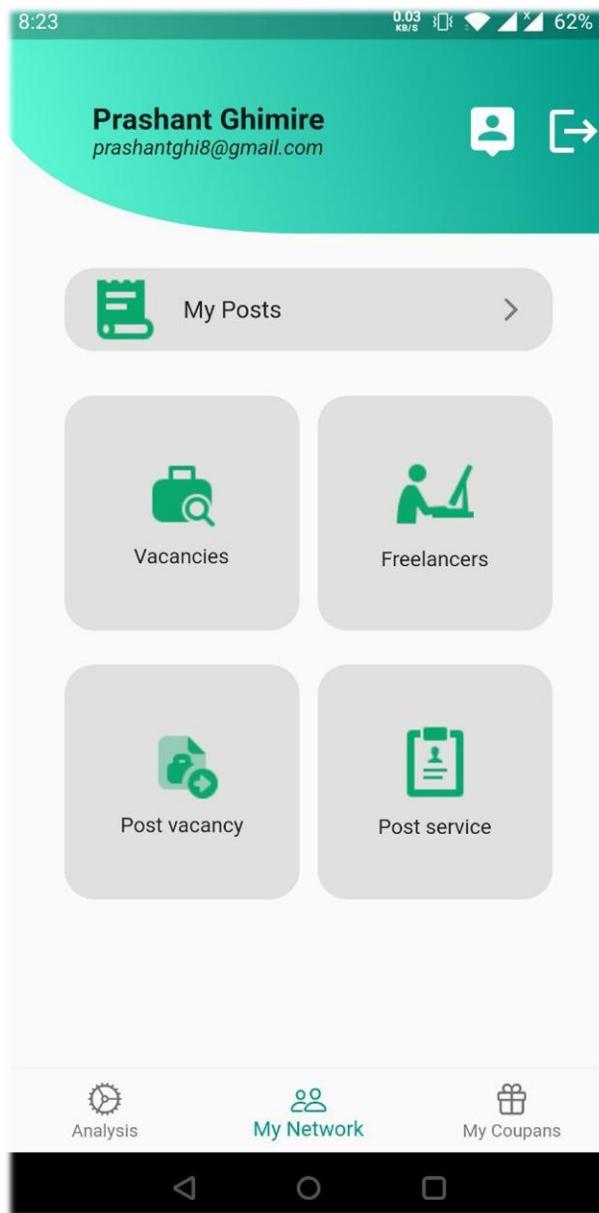


Figure 73 Dashboard in portrait view

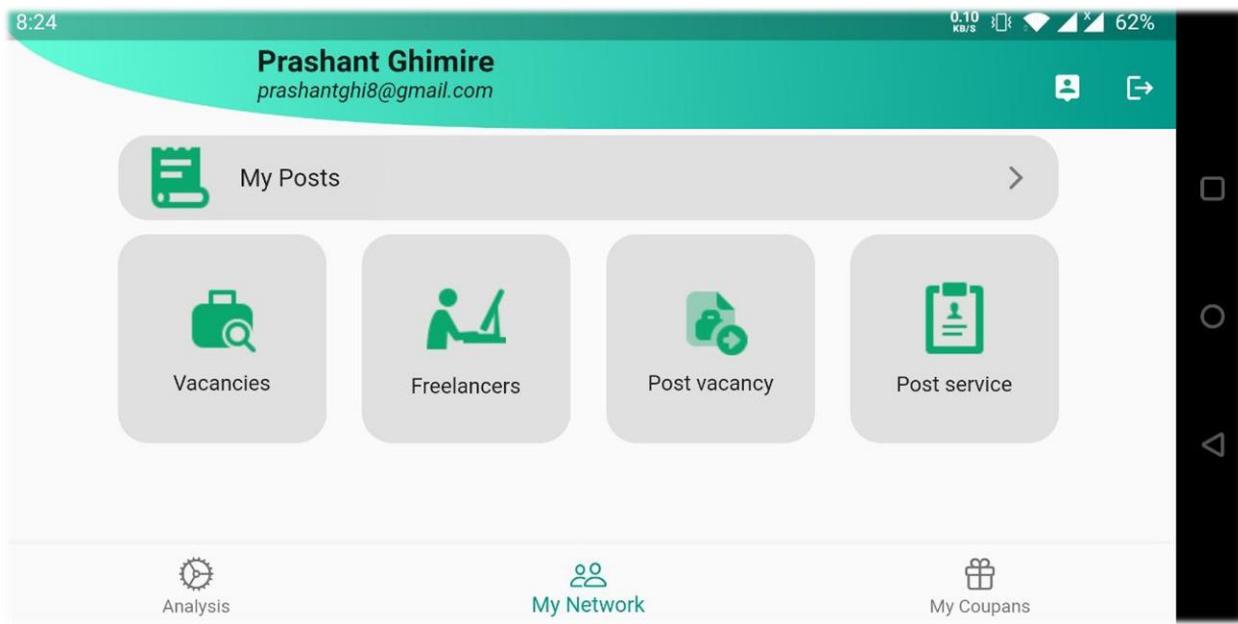


Figure 74 Dashboard in landscape view

UT14: To show user can post a vacancy in the platform.

Action	All the fields were filled and 'Post vacancy' button was clicked.
Expected Result	A toast message would be shown.
Actual Result	A toast message stating vacancy posted on the platform was shown.
Conclusion	The vacancy was added which states test is successful.

Table 20 Unit test 14

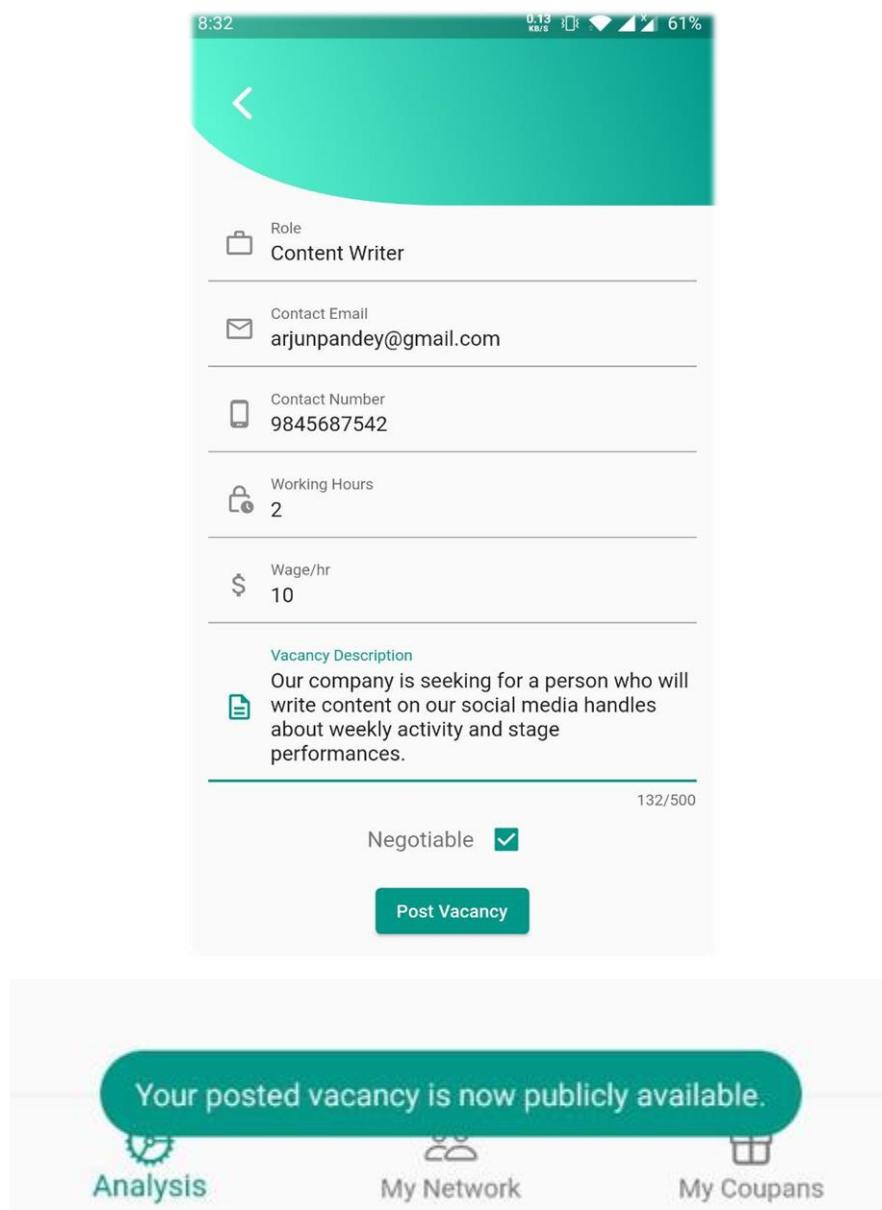


Figure 75 Posting vacancy in the platform

UT15: To show user cannot post more than five vacancies.

Action	A new vacancy is tried to add from the user account which already has five active posts.
Expected Result	A toast message would be shown.
Actual Result	A toast message stating vacancy posted on the platform was shown.
Conclusion	The vacancy was added which states test is successful.

Table 21 Unit test 15

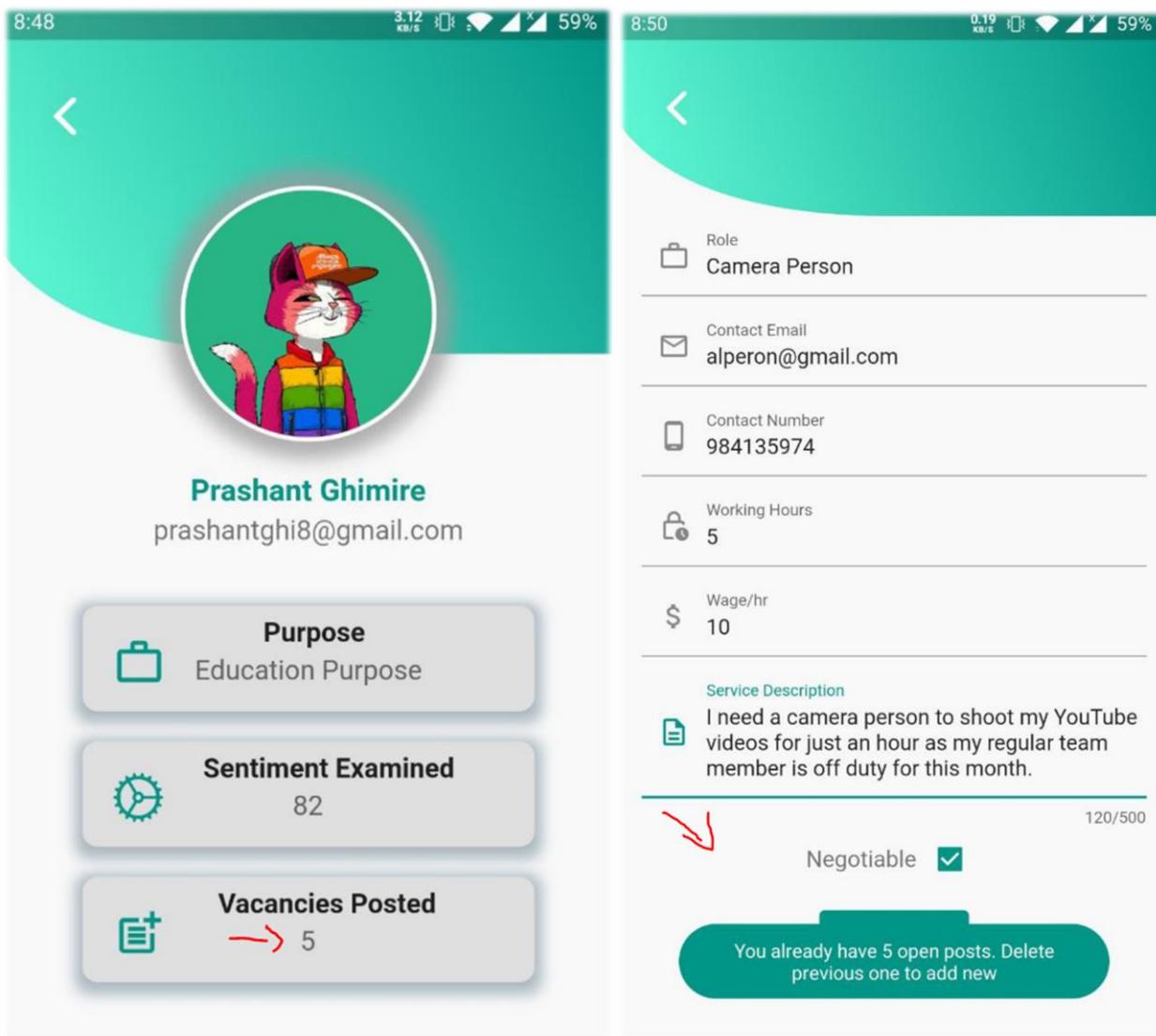


Figure 76 User having five open vacancies trying to add another new vacancy

UT16: To show user navigates to phone call when presses contact button.

Action	'Contact' button was pressed from detailed view of a freelancer.
Expected Result	User will be taken to phone call user interface.
Actual Result	Cellular calling window was opened after pressing 'Contact' button.
Conclusion	The feature worked as expected which states test is successful.

Table 22 Unit test 16

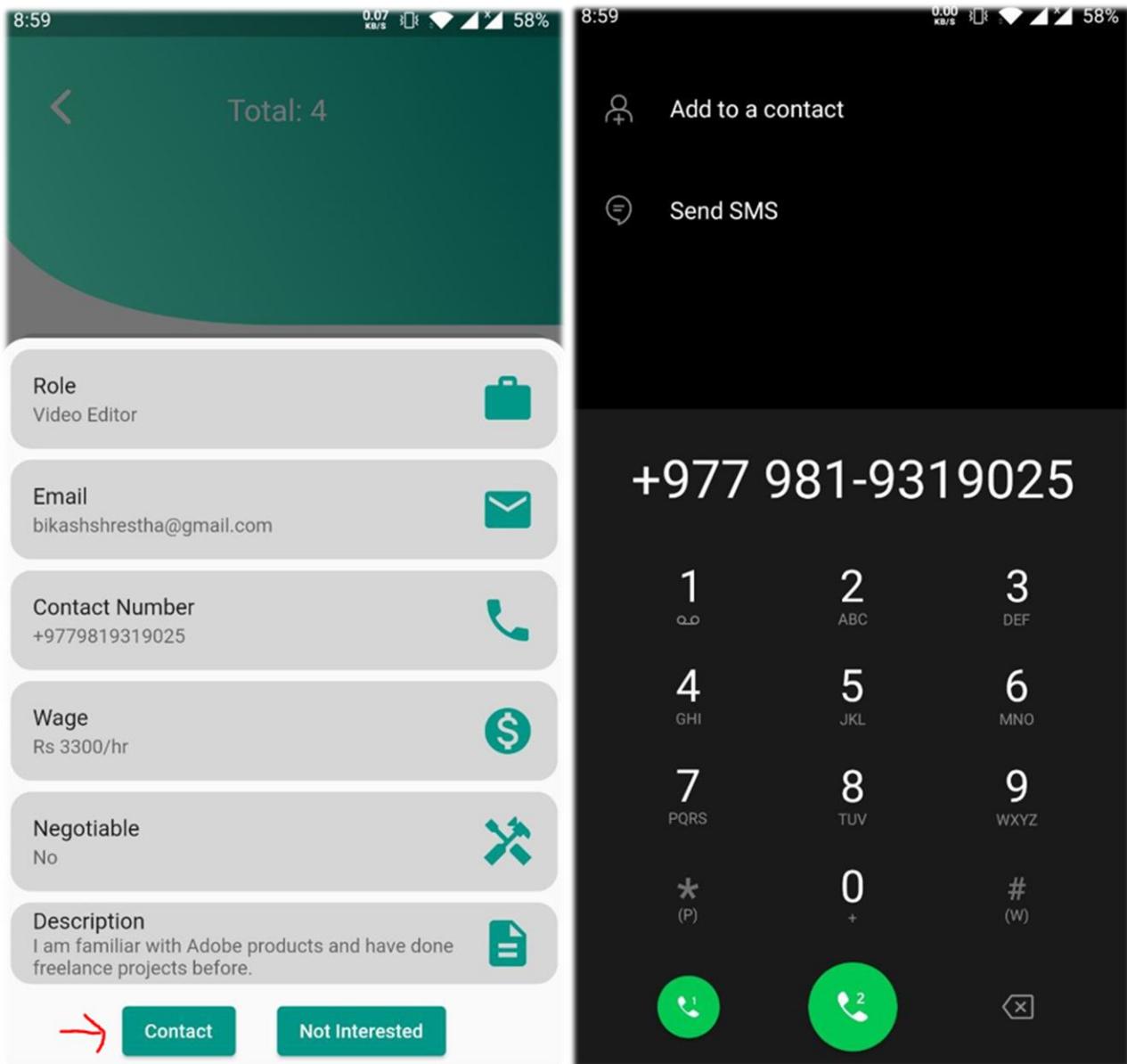


Figure 77 User navigated to phone call window

UT17: To verify user can delete their previously posted vacancy.

Action	'Delete Post' button was pressed from detailed view of a freelancer.
Expected Result	User will be taken to dashboard after deleting with a toast message.
Actual Result	Dashboard was opened with a message stating post deleted.
Conclusion	The feature worked as expected which states test is successful.

Table 23 Unit test 17

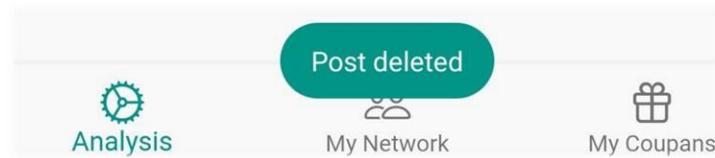
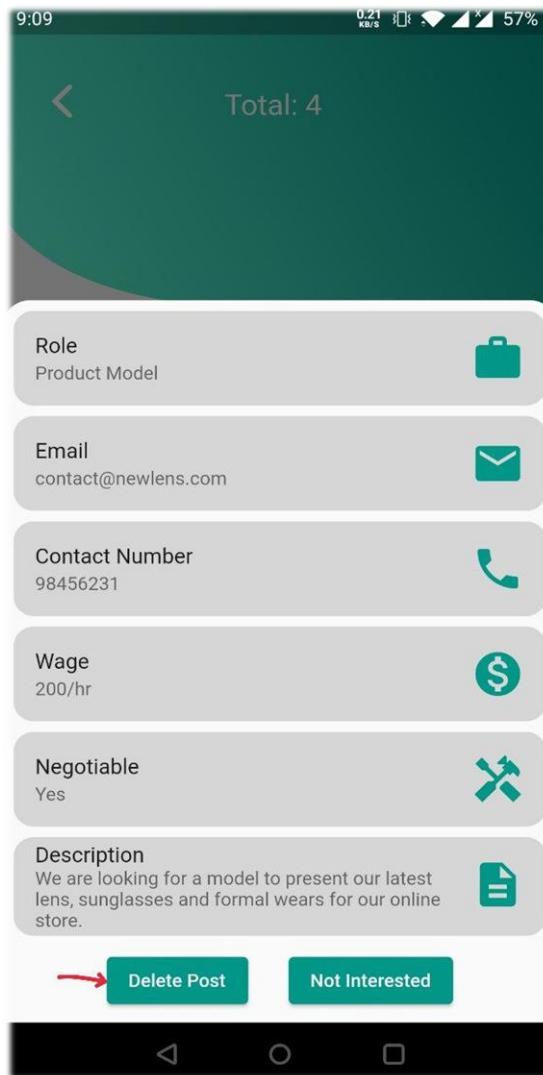


Figure 78 Deleting previously added post

UT18: To test user can logout from the platform.

Action	'Logout' icon was pressed from dashboard.
Expected Result	User will be taken to Login screen.
Actual Result	Login screen was opened after pressing logout icon.
Conclusion	The feature worked as expected which states test is successful.

Table 24 Unit test 18

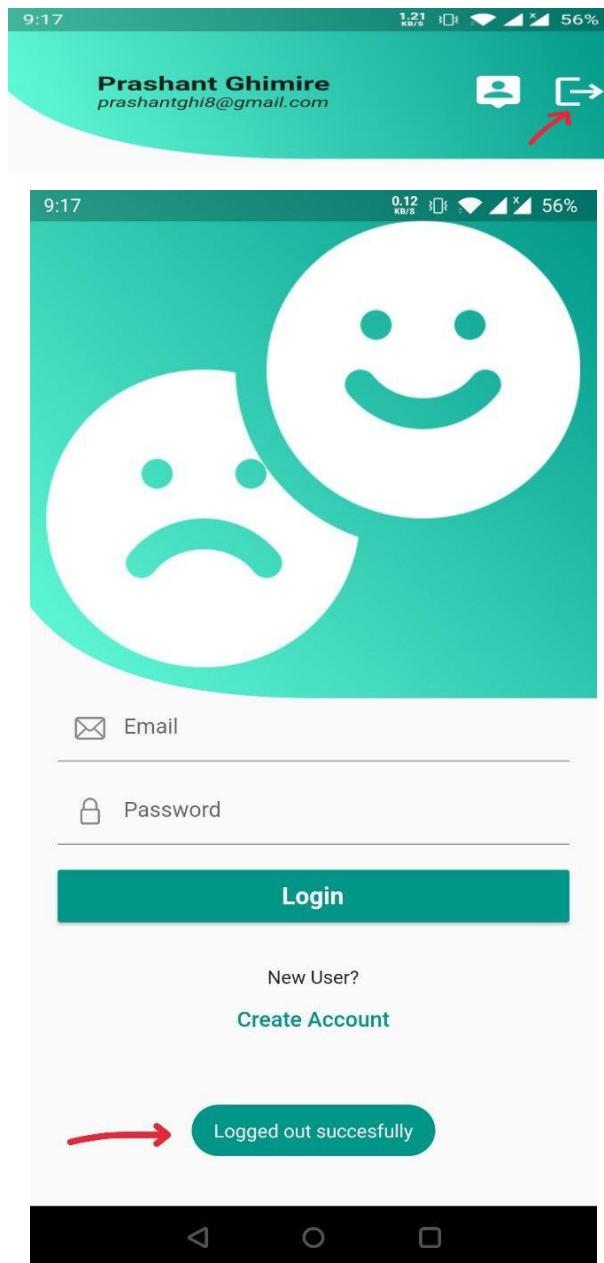


Figure 79 Logging out from the platform

UT19: To verify that error message will be displayed if user tries to access saved reports without internet connection.

Action	Both the Wi-Fi and cellular data was turned off and 'My Reports' was clicked.
Expected Result	A toast message would be shown.
Actual Result	A toast message about no internet was shown.
Conclusion	The feature worked as expected which states test is successful.

Table 25 Unit test 19

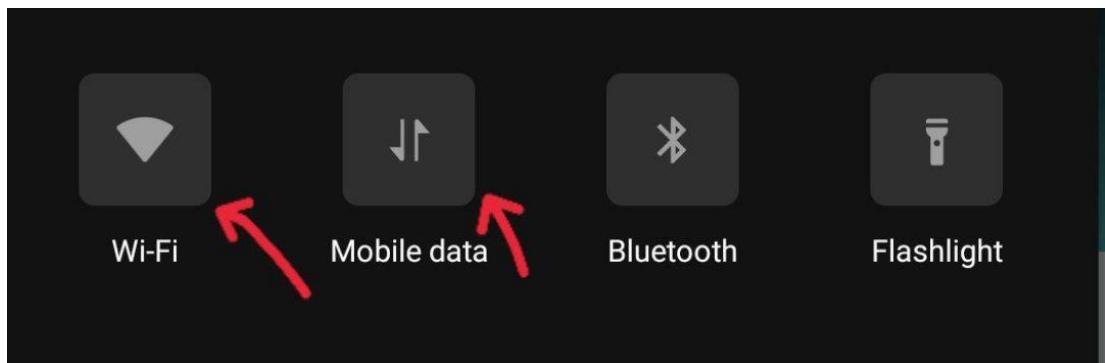


Figure 80 Turning off Wi-Fi and Mobile data

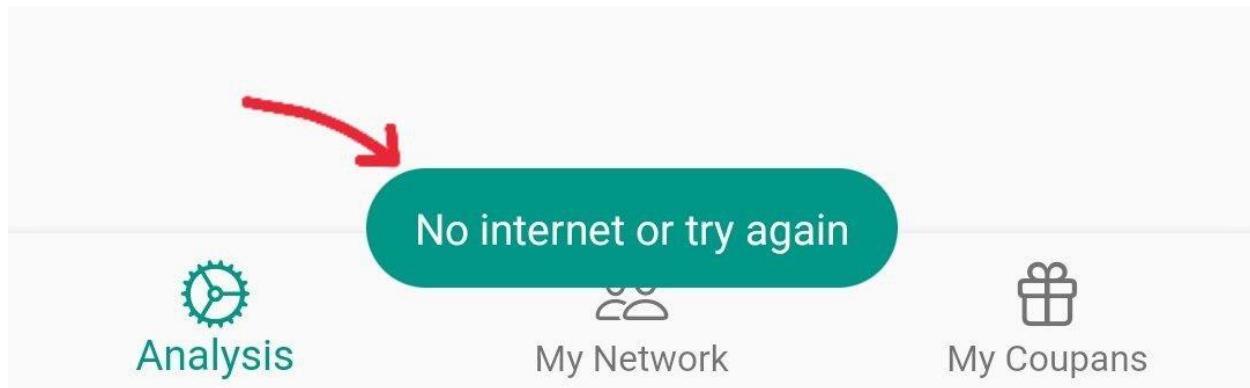


Figure 81 Trying to view reports without internet connection

UT20: To show user's name and email in the interface when user is navigated from login screen to dashboard.

Action	Valid login credential was provided and navigated to dashboard.
Expected Result	User's name and email was expected to be shown on top of dashboard.
Actual Result	Name and email were not shown while opening application for first time.
Conclusion	User information was shown from second and further opening of application but not in first time stating the test failed and there is space to improve.

Table 26 Unit test 20



Figure 82 Dashboard after logging in

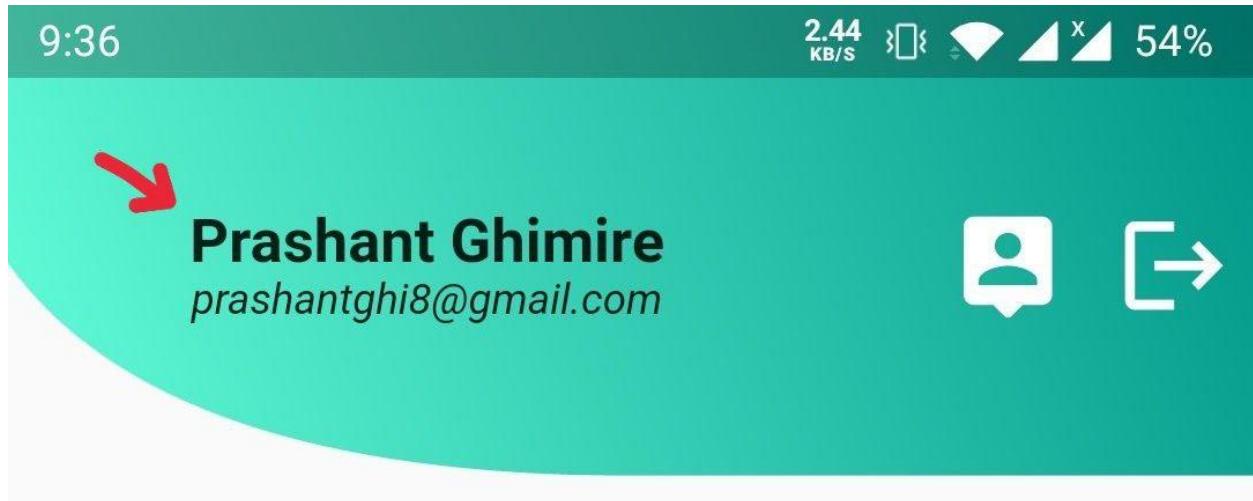


Figure 83 Dashboard after opening for second time

API unit testing:

UT21: Trying to view user's profile without supplying a bearer token.

Action	HTTP GET request was done on profile viewing backend URL
Expected Result	An error message would be provided.
Actual Result	Error message about missing authorization header was shown.
Conclusion	The test is successful.

Table 27 Unit test 21

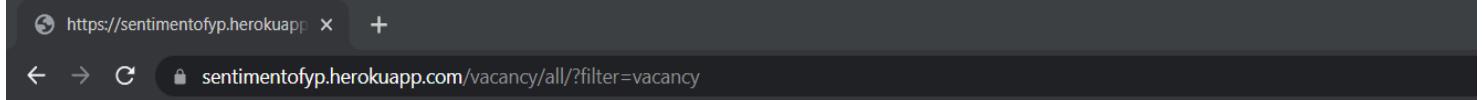
The screenshot shows the Postman application interface. At the top, there is a header bar with the method **GET** and the URL <https://sentimentofyp.herokuapp.com/profile/>. Below the header, there are tabs for **Params**, **Authorization**, **Headers (6)**, **Body**, and **Pre-request Script**. The **Authorization** tab is currently selected, showing a dropdown menu set to **No Auth**. In the main body area, there are tabs for **Body**, **Cookies**, **Headers (6)**, and **Test Results**. The **Body** tab is selected, displaying a JSON response. The response is a single-line object with three numbered lines: Line 1 shows an opening brace {}, Line 2 shows the key "msg": "Missing Authorization Header", and Line 3 shows a closing brace }. The JSON response is displayed in a syntax-highlighted format.

Figure 84 Trying to access profile without providing authentication header

UT22: To show open vacancies can be viewed without user authentication.

Action	Vacancies viewing URL was opened on browser.
Expected Result	JSON data of vacancies would be shown.
Actual Result	Vacancies information was shown.
Conclusion	Vacancies data was shown without providing authentication header which makes this test successful.

Table 28 Unit test 22



```
{
  "msg": "Vacancies loaded successfully",
  "success": "true",
  "vacancies": [
    {
      "contact_email": "theprashantghimire@gmail.com",
      "contact_number": "+9779818483125",
      "description": "I can work as cross platform application developer for your company. Familiar with BLOC State Management",
      "id": 1,
      "is_negotiable": true,
      "is_vacancy": true,
      "title": "Flutter Developer",
      "wage_hr": "Rs 4200",
      "work_hr": 3
    },
    {
      "contact_email": "achyutchapagain05@gmail.com",
      "contact_number": "9824085023",
      "description": "C Programming Teacher to teach to Bachelor Level Students. Time: Morning Shift. Location: Birtamode ! Number of Classes: 2.",
      "id": 9,
      "is_negotiable": true,
      "is_vacancy": true,
      "title": "C programming Teacher",
      "wage_hr": "30",
      "work_hr": 2
    },
    {
      "contact_email": "jobs@findajob.com",
      "contact_number": "+977985342861345",
      "description": "Looking for a flutter developer to develop travel guid android application. Interested people can apply ",
      "id": 11,
      "is_negotiable": true,
      "is_vacancy": true,
      "title": "Flutter Junior Developer",
      "wage_hr": "$7",
      "work_hr": 6
    },
    {
      "contact_email": "samirakharel@yahoo.com",
      "contact_number": "9812345678",
      "description": "I need a graphic designer to convert all the rough sketches into digital graphics for my upcoming book.",
      "id": 12,
      "is_negotiable": false,
      "is_vacancy": true,
      "title": "Graphic Designer",
      "wage_hr": "30/hr",
      "work_hr": 4
    }
  ]
}
```

Figure 85 Viewing vacancies information from web browser

UT23: To show user cannot delete another user's vacancy post.

Action	Vacancy id of other user's post was provided to delete.
Expected Result	Response about invalid API request would be provided.
Actual Result	Message stating unauthorized request was shown.
Conclusion	The vacancies are protected from unauthorized removal stating the test successful.

Table 29 Unit test 23

DELETE  https://sentimentofyp.herokuapp.com/vacancy/?id=13

Params ● Authorization ● Headers (7) Body Pre-request Script Tests Settings

Type Bearer Token

! Heads up! These parameters hold sensitive data. To keep this data secure while working in a collection, we recommend using variables. [Learn more about variables](#)

The authorization header will be automatically generated when you send the request.  
[Learn more about authorization](#)

Token eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ...

Body Cookies Headers (6) Test Results Status: 200 OK Time: 834 ms

Pretty Raw Preview Visualize JSON

```

1  {
2    "msg": "You are not authorized",
3    "success": "false"
4  }

```

Figure 86 Trying to delete another user's vacancy

UT24: To show user cannot perform analysis on more than 2000 comments.

Action	Video id, API key and 5000 number of comments were given as input for post request.
Expected Result	Sentiment analysis would be prevented and error message would be provided in response.
Actual Result	Message guiding how many comments can be taken was shown in API response.
Conclusion	The backend is throttled from bulk comments analysis stating the test successful.

Table 30 Unit test 24

The screenshot shows a POST request to <https://sentimentofyp.herokuapp.com/red/>. The request body is JSON with the following content:

```

1 {
2   "video_id": "Fop2oskTug8",
3   "api_key": "AIzaSyCiRaDCVSOR-NX-gCRu47Tqq7R-NVk8ZrA",
4   "approx_comments": 5000
5 }

```

The response body is JSON with the following content, where the 'approx\_comments' value is highlighted in red:

```

1 {
2   "msg": "Comments between 20 to 1000 can only be set at the moment",
3   "success": "false"
4 }

```

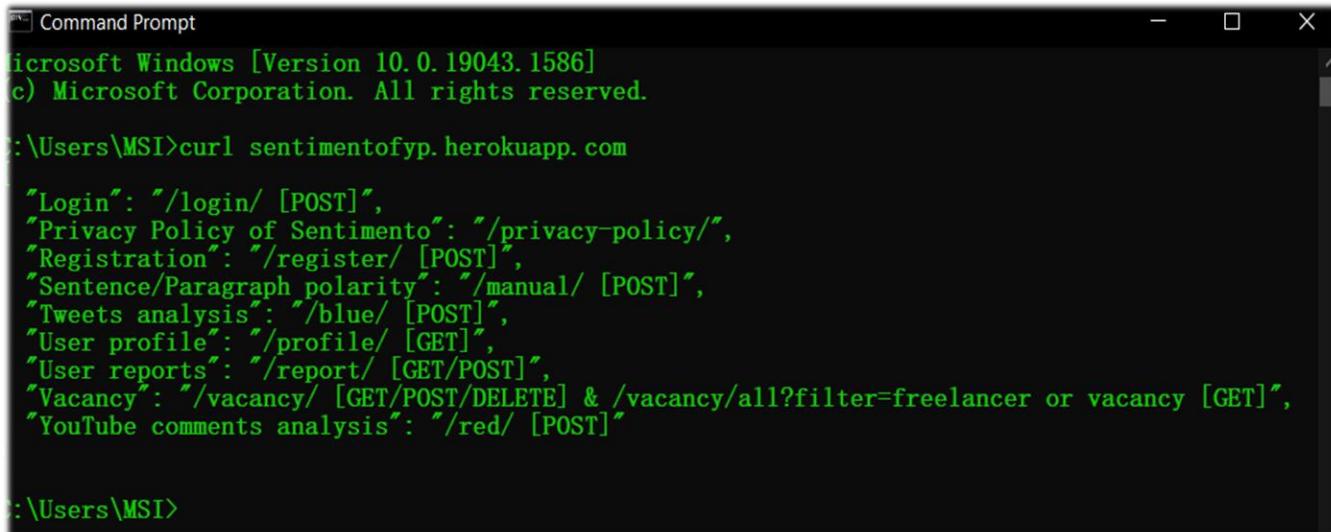
Figure 87 Trying to perform sentiment analysis on more than 1000 data

#### 4.3 System Testing

ST1: To validate backend server is live and responsive in cloud.

Action	Curl command was applied to domain of the hosted API from command prompt.
Expected Result	The built API documentation would be shown.
Actual Result	JSON format information was shown.
Conclusion	The Flask built backend API hosted on Heroku transferred data as response which states server is live and test is successful.

Table 31 System test 1



```

Command Prompt
Microsoft Windows [Version 10.0.19043.1586]
© Microsoft Corporation. All rights reserved.

:C:\Users\MSI>curl sentimentofyp.herokuapp.com

"Login": "/login/ [POST]",
"Privacy Policy of Sentimento": "/privacy-policy/",
"Registration": "/register/ [POST]",
"Sentence/Paragraph polarity": "/manual/ [POST]",
" Tweets analysis": "/blue/ [POST]",
"User profile": "/profile/ [GET]",
"User reports": "/report/ [GET/POST]",
"Vacancy": "/vacancy/ [GET/POST/DELETE] & /vacancy/all?filter=freelancer or vacancy [GET]",
"YouTube comments analysis": "/red/ [POST]"

:C:\Users\MSI>

```

Figure 88 Checking hosted backend API on Heroku

ST2: To show backend is properly integrated with frontend.

Action	Sentiment analysis count is noted, then a manually provided sentence analysis is performed then again, the analysis count is viewed.
Expected Result	Sentiment analysis count would increase by 1.
Actual Result	The analysis count increased from 83 to 84 after performing analysis.
Conclusion	Increase on analysis count indicates that database is updated indicating successful integration of frontend, backend and database.

Table 32 System test 2



Figure 89 Count before performing analysis

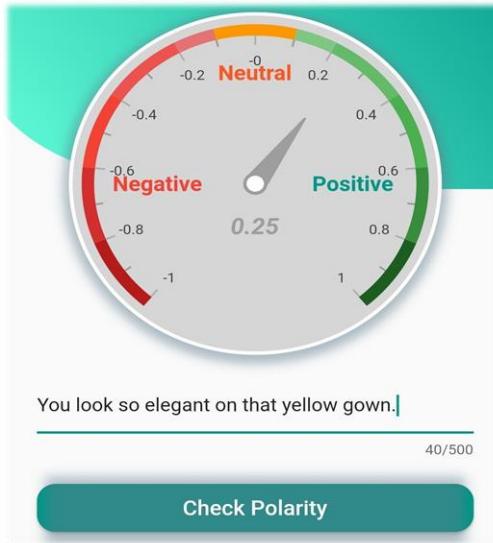


Figure 90 Performing polarity check

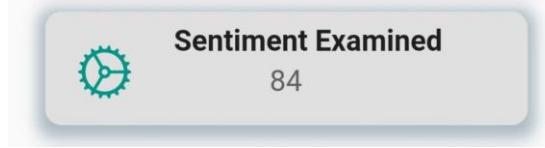


Figure 91 Count after performing analysis

ST3: To show user can perform sentiment analysis on tweets.

Action	Sentiment analysis on 'Nepse' topic was done.
Expected Result	Analysis report would be shown.
Actual Result	The detailed report was shown.
Conclusion	Test is successful.

Table 33 System test 3

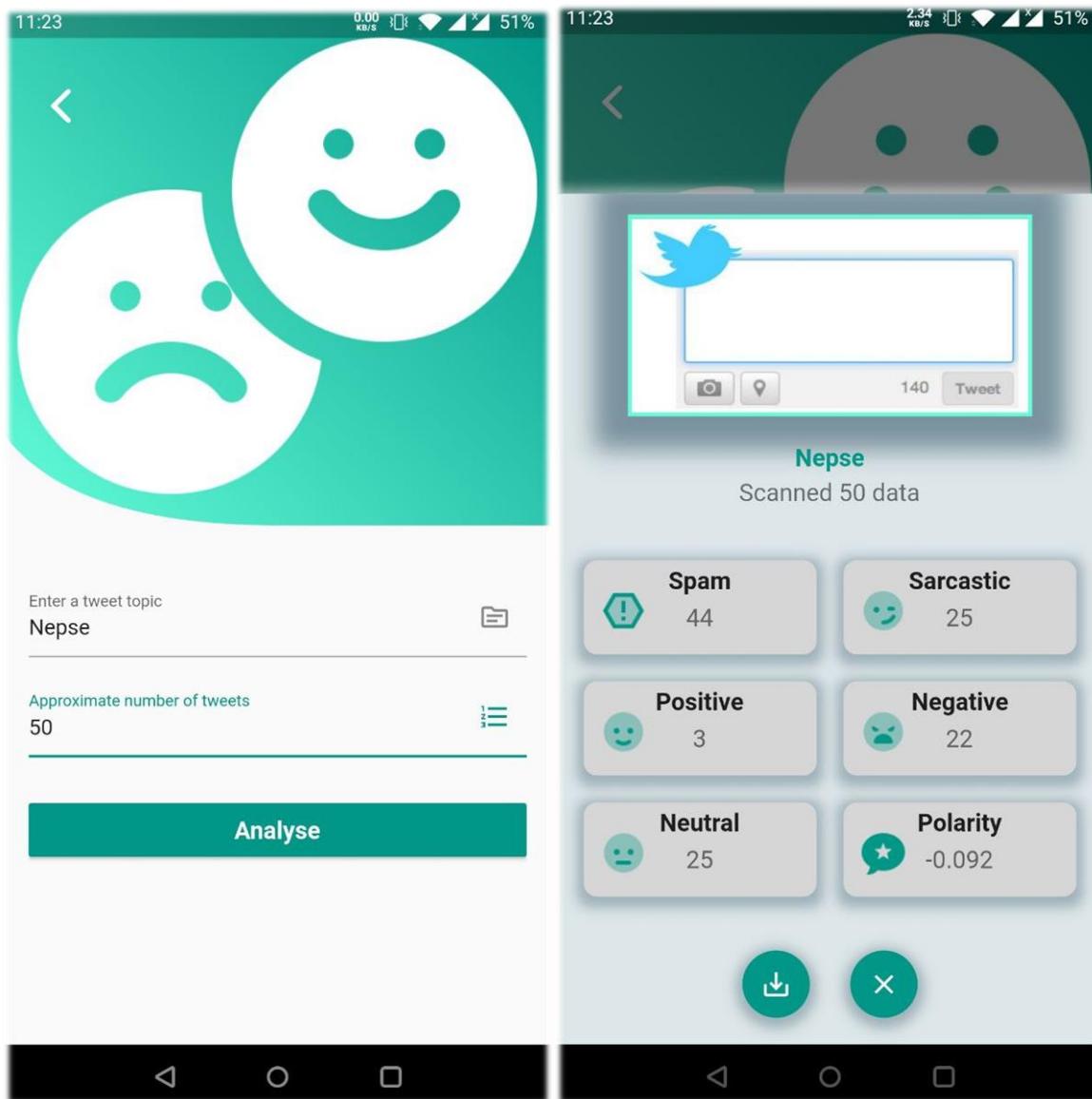


Figure 92 Performing sentiment analysis on tweets

ST4: To show user can save sentiment report.

Action	'Save' icon was clicked after viewing sentiment report.
Expected Result	The recently saved report would be shown on 'My Reports' interface.
Actual Result	Report on lastly saved analysis was shown.
Conclusion	The saved report is saved on database indicating integrity of Report table and user interface. So, this test is successful.

Table 34 System test 4

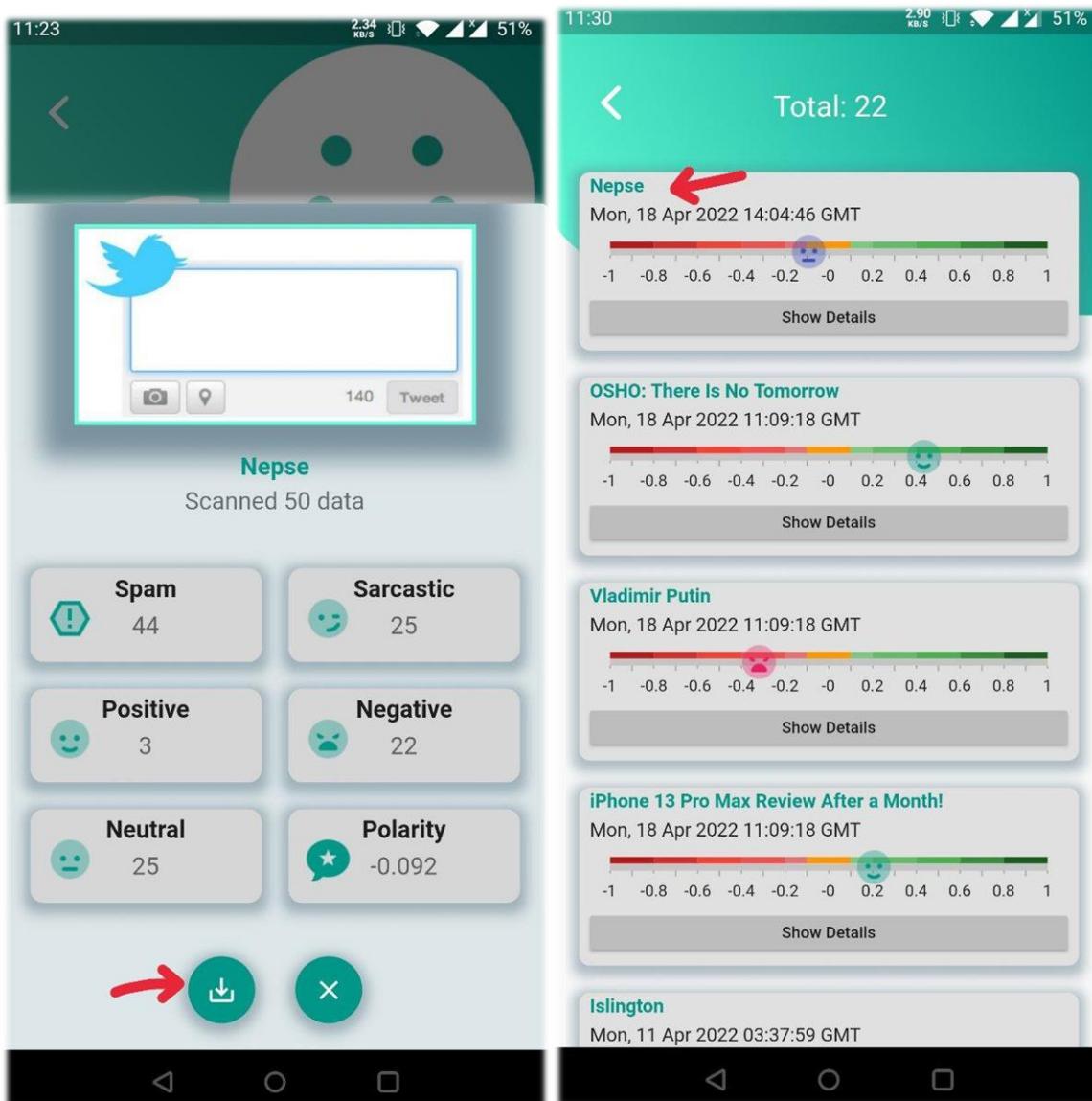


Figure 93 Viewing recently saved report

ST5: To show build Flutter application runs with no compilation errors

Action	Android Emulator was selected from VS Code and 'flutter run' command was executed from the terminal.
Expected Result	The application was expected to be opened in Android emulator.
Actual Result	Application was opened after 8.2 second of compilation without any errors.
Conclusion	This test is successful as application opened as expected.

Table 35 System test 5

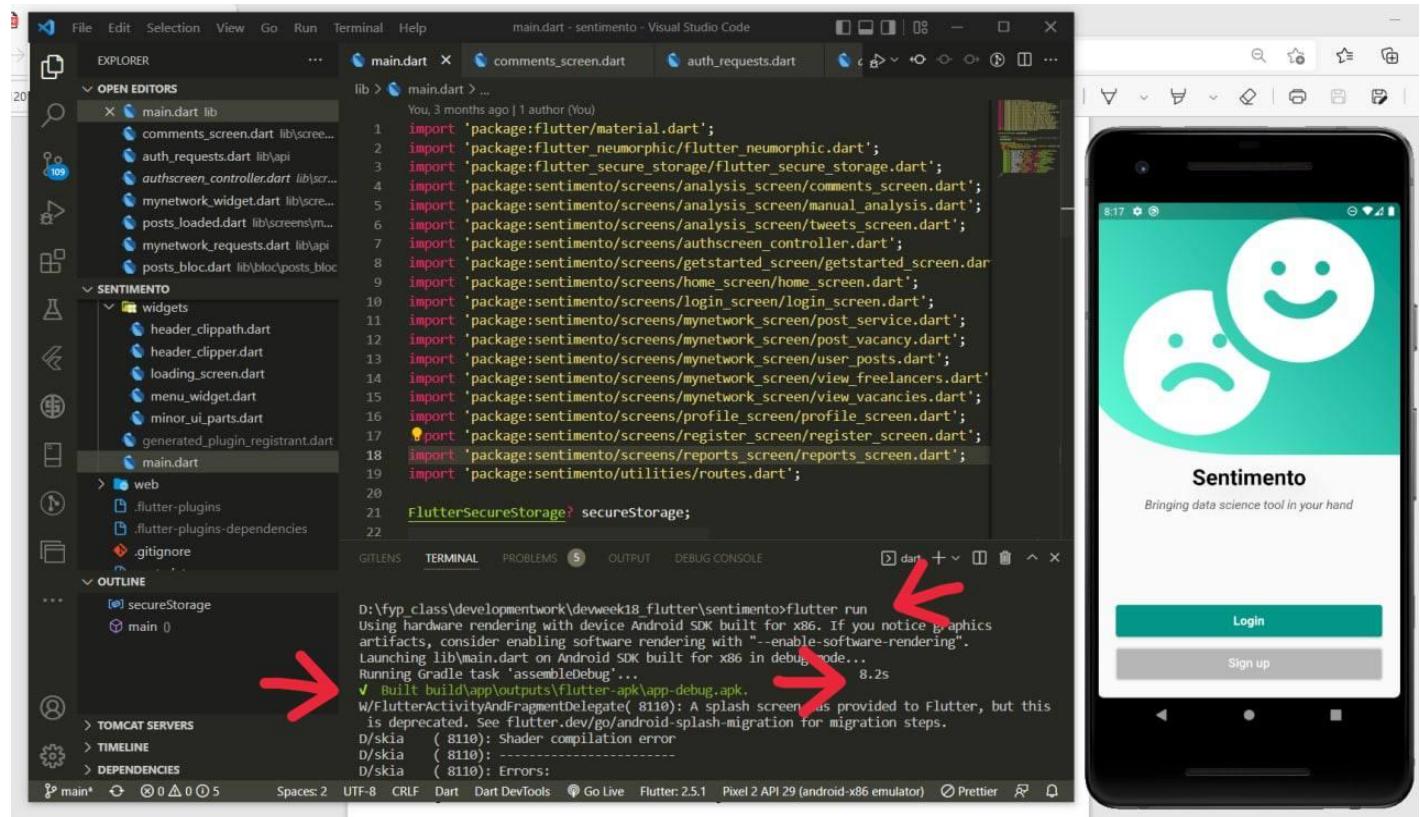


Figure 94 Flutter Application Compilation

#### 4.4 Critical Analysis

The above documented unit and system testing demonstrates that all the requirements are considered and implemented in this platform. By following the SCRUM principle while carrying out the project, the development was completed within allocated timeframe and enough time was available for additional improvements.

One of the testings failed for this project. The error was faced in frontend part of the project. User's name and email was not shown in the top of profile on first login. This error existed as there was no state management applied for this part. Apart from this, all the testings were successful. To summarize the carried-out test, following points are written as follows:

- i. Functionalities mentioned in the requirement analysis worked smoothly as expected.
- ii. Frontend, backend was fully integrated and worked all together.
- iii. All the validation, data throttling and logical parts of the interface worked as expected.
- iv. The state management worked smooth and loading screens were shown in every buffering stage for better user experience.

## Chapter 5: Conclusion

### 5.1 Legal, Social and Ethical Issues

Here's how the platform is built to prevent any professional issues from happening:

Creating a product with the potential of numerous users to be involved adds a sensitive responsibility to protect the rights and information of consumers of the product. Considering this part of the professional ethics and computer law: certain ethics, moral and legal duties are strictly followed in the development of this project which are as follows:

- i. Users are trusting with their personal information while using this app, understanding this, passwords are hashed before storing in database. The email that user provides while registering in the platform is not visible to third person/parties.
- ii. The third party's (YouTube and Twitter) API key user has to enter to perform sentiment analysis are user's sensitive information. Keeping this in mind, those API keys are not stored in system's database. Instead, the keys are encrypted and locally saved in the user's device.
- iii. To prevent bad actors from posting spam vacancies, the system throttles users to post only five open vacancies at a time.
- iv. There might be any incorrect information while posting vacancy by user. In such case user wants to remove such misinformation. Considering this, there is a feature to delete previous posts.

## 5.2 Advantages

### Real life uses of sentiment analysis

Social Media Monitoring, Political Campaigns, Advertising/Marketing, Customer Review Extraction, Market Research, Mental Health Brand/Influencer's Reputation Monitoring are some of the areas where sentiment analysis can be used. All these mentioned fields or parties can be benefited in accessing their strength and weakness by analysing their audience's opinions or from public (Saifee Vohra, 2013).

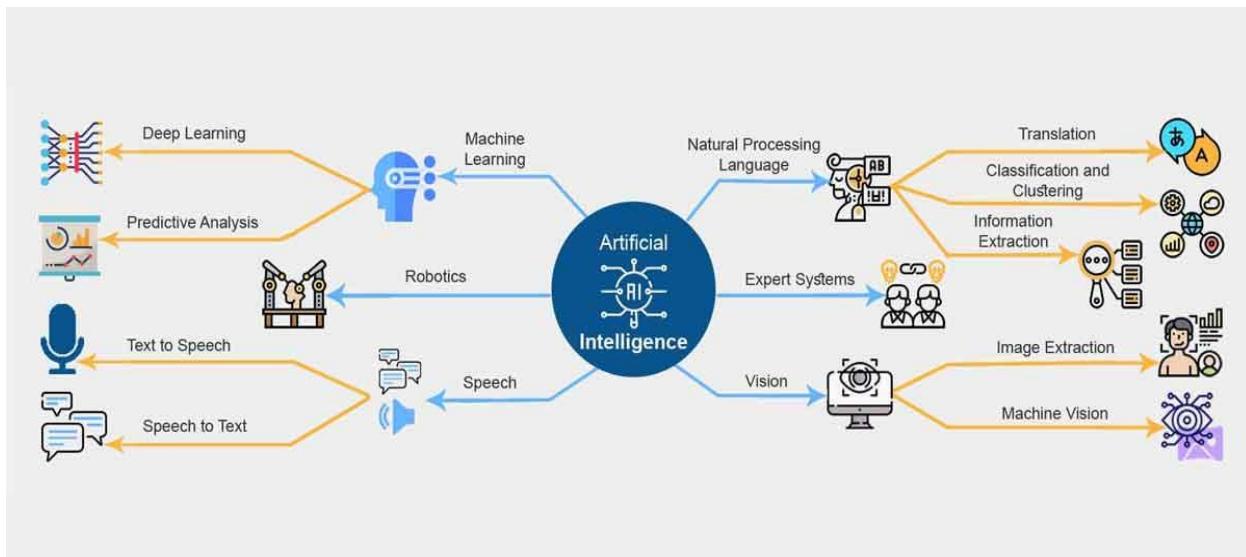


Figure 95 Distinct use cases of sentiment analysis (Samadhan Engineering, 2021)

For the artificial intelligence or machine trained model to better communicate with humans, there is need for machines to understand the human sentiments. Thus, the application of sentiment analysis is not only limited to human uses but also to artificial intelligence. The tasks where humans are better than machines at this stage can be replaced by machines themselves with the better sentiment consumption by machines. There are endless areas where sentiment analysis can be performed as it carries huge potential use cases. And it is one of the emerging, innovative and applicable area for the present and future human society (Somers, 2019).

Real life uses of built application:

- i. Users can view/add posts about freelancing service and job vacancies related to social media field.
- ii. Data analysts can quickly perform sentiment analysis and extract summary from the platform for their any sorts of use case.
- iii. Marketing professionals can observe current public sentiment about any public figures and compare among them which can be useful for decision taking about collaborating with best influencers.
- iv. Social media influencers can observe audience's feedbacks on their content and publish further content accordingly.
- v. Public sentiment regarding any topic can also be useful for news media professionals. Also, other users can also benefit from this feature.

### 5.3 Limitations

Challenges that can arise while performing sentiment analysis are listed below:

(Mohammad, 2016)

Sentiment Analysis is a complex and arduous task in natural language processing. The challenges of machine learning based sentiment analysis are as follows:

- i. Irony and Sarcasm: People uses sarcasm and ironic sentences to express the meaning. In such cases, the real point they want to make can be one thing but the straight sentence can mean something different. Even human can find it difficult to understand sometimes which makes more difficult for trained models to predict the actual meaning.
- ii. Context and Polarity: As trained model works within the provided boundaries, it's hard to understand the actual sentiment according to the context. Example: the opinion can be regarding something but the context can be different. Here the calculated sentiment polarity does not belong to real context resulting less trustworthy analysis.
- iii. Stop words: The stop words are better to remove for reducing the numerical vectorized array of data. This can save memory and better performance but removal of stop words can alter the meaning of a sentence. Example: In the sentence 'I do not like you'. When 'not' stop word is removed while pre-processing, the meaning changes. So, it is challenging to deal with stop words.

Limitations of this built application:

The designed algorithm is not well optimized to perform analysis quickly. There is still adequate place to improve time and space complexity of the algorithm. So, the platform is throttled to only take maximum of one thousand data (comment/tweet) at a time.

## 5.4 Personal experience and Future work

### 5.4.1 Self-reflection or personal experience:

Addition of this project on my curriculum vitae is a significant factor that supported me to receive invitation for a trainee interview in Business Informatics department at Technical University Dortmund located in Germany. This opportunity was achieved as I was looking for international opportunities on Data Science. I got the chance to describe my final year project about the overall integration of the built system. Encouraging feedback and short mentoring was received during the interview.

Using Flutter to build the frontend part of this project and my past internship on this framework improved my Flutter knowledge and hands-on experience. This provided me the confidence and certain level of quality to engage in mobile application to smoothly transit from academic to professional life.

### 5.4.2 Here's how I will be further engaged on this project in future:

Being a data enthusiast from early days of entering technology field, there is always a keen interest of pursuing Data Science for my post graduate studies. This is an undergraduate academic work but I will be engaged in this same field even after the completion of this undergraduate project.

The further aim of mine is to improve space and time complexity of the sentiment analysis part of this project. The accuracy of the analysis at this level is also not well reliable. Further research and theoretical findings from multiple scholars will be absorbed to improvise the quality of this work. And there is a significant possibility to pursue this same topic for my post-graduate thesis work.

## Chapter 6: References

Excluding theoretical research for carrying out this project, there is also use of diverse contents that assisted in executing development work of the project. Attachment of all the references that were accessed during or before the development of this project are as follows:

- i. LinkedIn learning course for building Flask API by Bruce Van Horn



Bruce Van Horn  
Lead Software Developer at Visual Storage Intelligence

+ Follow on LinkedIn Show LinkedIn Profile

Bruce Van Horn is a lead software developer at Visual Storage Intelligence.  
A full-stack software engineer with a proven ability to develop high-performance applications for any platform or medium, Bruce has a history of serving his employers with distinction. For example, in 2008, he began working at One Network Enterprises, a company formed by his former colleagues; he worked as a Java developer on custom projects for the United States Marine Corps and lent his

show more ▾

5 Courses

Sort by: Newest ▾

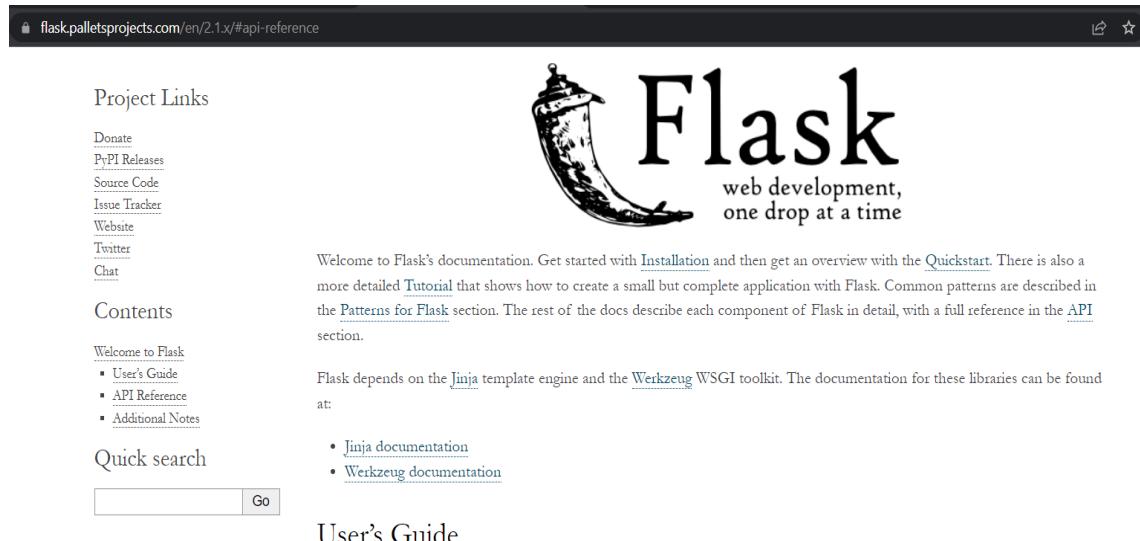


COURSE  
**Building RESTful APIs with Flask**  
By: Bruce Van Horn • Aug 2019  
2h 32m 1h 1m 55s left

... Unsave

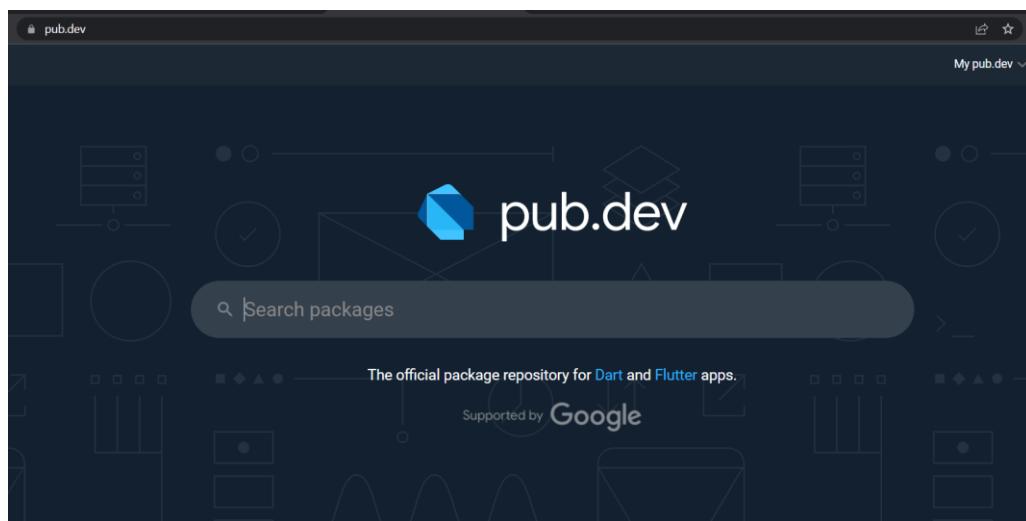
Figure 96 Snapshot of a LinkedIn learning course for Flask

ii. Flask official documentation: While performing CRUD operations and making models for database, Flask's official documentation was surfed for class, methods using guide.



*Figure 97 Snapshot of Flask documentation web page*

iii. Flutter package manager: It is an official package manager where developers can contribute their work and enormous cross-platform application components are available here.



*Figure 98 Snapshot of Flutter package manager*

- iv. Stack overflow community: When there was run-time error building frontend or backend, stack overflow was surfed to view how developers faced and solved similar issues in the past. With such reference, the debugging and solving run-time errors were smooth.
- v. Scikit-learn: For the sentiment analysis (Multinomial Naive Bayes algorithm) part of the project, scikit-learn, a machine learning library has been used.

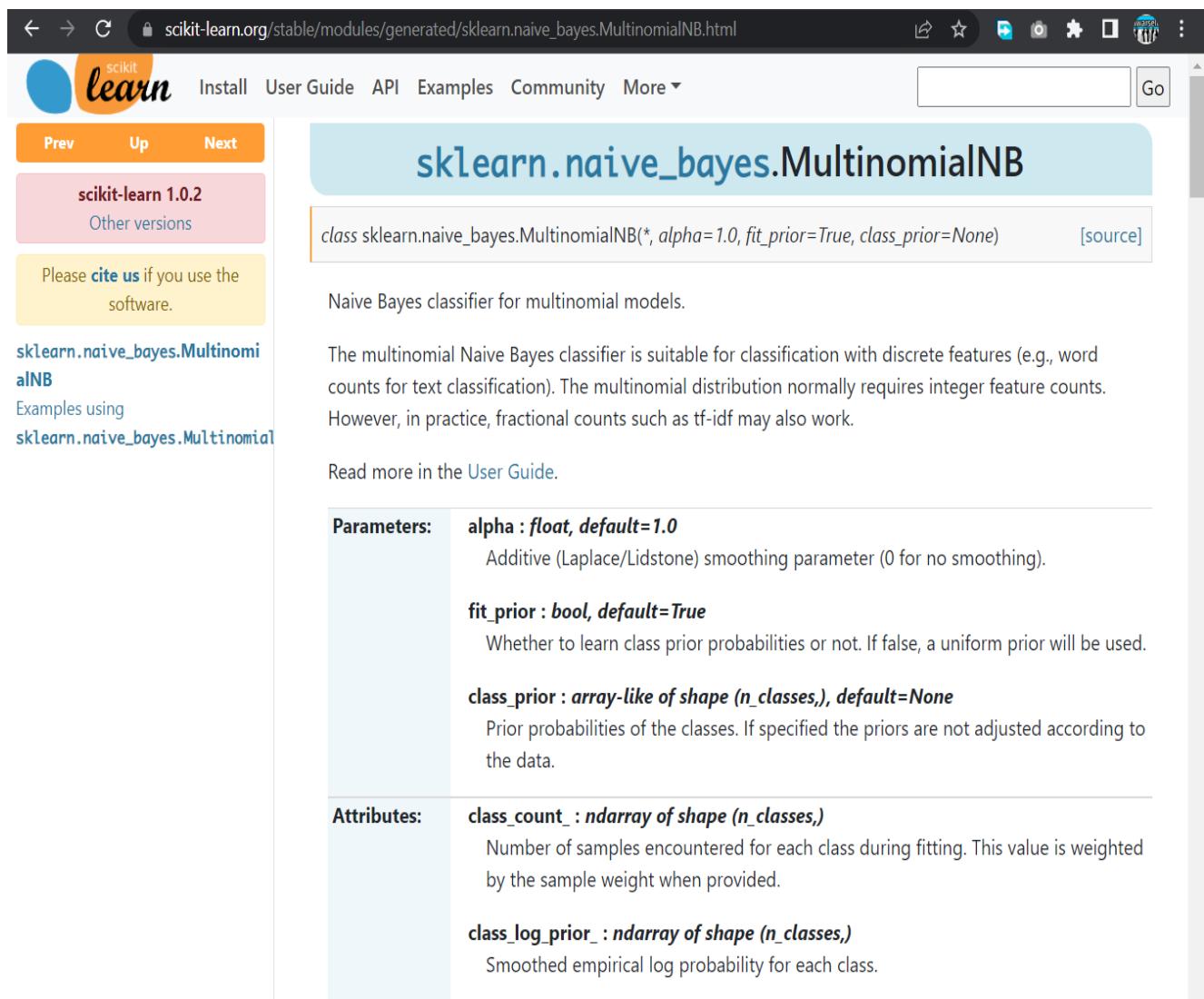


Figure 99 Snapshot of scikit-learn library

## Chapter 7: Bibliography

- Aqeel, A., 2020. *BLoC (Local & Global) State Management with Flutter*. [Online] Available at: <https://devmuaz.medium.com/bloc-local-global-state-management-with-flutter-e8e443b135f8> [Accessed 23 March 2022].
- Binita Verma, R. S. T., 2018. *Sentiment Analysis Using Lexicon and Machine Learning-Based Approaches: A Survey*. Bhopal, Maulana Azad National Intitute of Technology.
- Bloc Library, 2021. *Why Bloc?*. [Online] Available at: <https://bloclibrary.dev/#/whybloc> [Accessed 2 April 2022].
- C9 Apps Desenvolvimento de Software Ltda ME, 2017. *TSentiment*. s.l.:C9 Apps Desenvolvimento de Software Ltda ME.
- Doshi, S., 2021. Why I Prefer Flutter Over React Native for App Development. *Web Dev Zone*, 23 August, pp. 2-4.
- GeeksforGeeks, 2021. *Saving a machine learning Model*. [Online] Available at: <https://www.geeksforgeeks.org/saving-a-machine-learning-model/> [Accessed 10 January 2022].
- GeeksforGeeks, 2021. *Software Engineering | Quality Characteristics of a good SRS*. [Online] Available at: <https://www.geeksforgeeks.org/software-engineering-quality-characteristics-of-a-good-srs/> [Accessed 9 December 2021].
- Glamazdina, Y., 2021. *A Guide to the Agile Software Development Life Cycle (SDLC)*. [Online] Available at: <https://brocoders.com/blog/agile-software-development-life-cycle/> [Accessed 26 March 2022].
- Hamilton, T., 2022. *Integration Testing: What is, Types, Top Down & Bottom Up Example*. [Online] Available at: <https://www.guru99.com/integration-testing.html> [Accessed 28 March 2022].

- Jay Khatri, I. K., 2020. *Sentiment Analysis - Know The Opinion*. s.l.:Jay Khatri.
- Jha, V., 2017. *Naive Bayes - machine learning algorithm for classification problems*. [Online]
- Available at: <https://www.techleer.com/articles/200-naive-bayes-machine-learning-algorithm-for-classification-problems/>
- [Accessed 07 December 2021].
- John Hattie, H. T., 2007. *The Power of Feedback*, Auckland: University of Auckland.
- Khanna, C., 2020. *What and why behind fit\_transform() and transform() in scikit-learn!*. [Online]
- Available at: <https://towardsdatascience.com/what-and-why-behind-fit-transform-vs-transform-in-scikit-learn-78f915cf96fe>
- Khorikov, V., 2020. *Unit Testing: Principles, Practices and Patterns*. 1st ed. Shelter Island: Manning Publications.
- Khristich, S., 2020. *The Scrum Cycle in Agile Software Development*. [Online]
- Available at: <https://tateeda.com/blog/the-scrum-cycle-in-agile-software-development>
- [Accessed 17 Febraury 2022].
- Linders, B., 2013. *Which Questions do you Ask in Retrospectives?*. [Online]
- Available at: <https://www.benlinders.com/2013/which-questions-do-you-ask-in-retrospectives/>
- [Accessed 28 March 2022].
- LinkedIn, 2021. *About LinkedIn*. [Online]
- Available at: <https://about.linkedin.com/>
- [Accessed 8 December 2021].
- Lior Rokach, O. M., 2015. *Decision Trees*, Tel Aviv: Oded Maimon Department of Industrial Engineering.
- Liu, J., 2016. *Data Visualization on Sentiment Analysis*. Chicago: Jiaqi Liu.
- Ludvig Persson, J. L., 2018. *Sarcasm Detection with TensorFlow*, Stockholm: School of Electrical Engineering and Computer Science.
- Mohammad, S. M., 2016. *Challenges in Sentiment Analysis*, Ottawa: National Research Council Canada.

- Mulvenna, J. A., 2015. *Improved lexicon-based sentiment analysis for social media analytics*, Belfast: Queen's University Belfast - Research Portal.
- Perisic, M., 2014. *The Good, The Bad and The Ugly of Agile Software Development*. [Online]
- Available at: <https://www.linkedin.com/pulse/20141106203747-11592828-the-good-the-bad-and-the-ugly-of-agile-software-development/>  
[Accessed 1 April 2022].
- Prokopisko, A., 2019. *Software development life cycle*. [Online]
- Available at: <https://medium.com/@artjoms/software-development-life-cycle-sdlc-6155dbfe3cbc>  
[Accessed 04 April 2022].
- Rajat Kathuria, M. K. G. V. K. B. S. K., 2017. *Future of Work in a Digital Era: The Potential and Challenges for Online Freelancing and Microwork in India*, New Delhi: INDIAN COUNCIL FOR RESEARCH ON INTERNATIONAL ECONOMIC RELATIONS.
- Saifee Vohra, J. T., 2013. Applications and Challenges for Sentiment Analysis : A Survey. *International Journal of Engineering Research & Technology (IJERT)*, 2(2), pp. 3-5.
- Samadhan Engineering, 2021. *Services: Artificial Intelligence Applications*. [Online]
- Available at: <https://www.thesamadhan.com/services/artificial-intelligence-applications>  
[Accessed 13 December 2021].
- Scikit-learn, 2021. *User Guide: Naive Bayes*. [Online]
- Available at: [https://scikit-learn.org/stable/modules/naive\\_bayes.html#multinomial-naive-bayes](https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes)  
[Accessed 14 Decembwr 2021].
- scrum.org, 2020. *What is Scrum?*. [Online]
- Available at: <https://www.scrum.org/resources/what-is-scrum>  
[Accessed 21 November 2021].
- Shimon Ullman, T. P., 2014. *Unsupervised learning*, Massachusetts: Massachusetts Institute of Technology.
- Somers, M., 2019. *Emotion AI, explained*. Massachusetts: MIT Management Sloan School.

- Sommerville, I., 2011. Software Requirement Documents. In: M. H. Marcia Horton, ed. *SOFTWARE ENGINEERING*. Hagerstown: Pearson, pp. 91-98.
- Stanford Edu, 2009. *Properties of Naive Bayes*. [Online]  
Available at: <https://nlp.stanford.edu/IR-book/html/htmledition/properties-of-naive-bayes-1.html>  
[Accessed 18 December 2021].
- Star Agile, 2020. *Overview of Scrum Phases*. [Online]  
Available at: <https://staragile.com/blog/scrum-phases>  
[Accessed 6 April 2022].
- Walaa Medhat, A. H. H. K., 2014. Sentiment Analysis Algorithms and applications. *Ain Shams Engineering Journal*, V(8), pp. 1093-1113.
- Young, D. C., 2013. *Software Development Methodologies*. Huntsville: Alabama Supercomputer Center.

## Chapter 8: Appendix

### Appendix 1: Additional Diagrams/Figures

#### i. Gantt chart

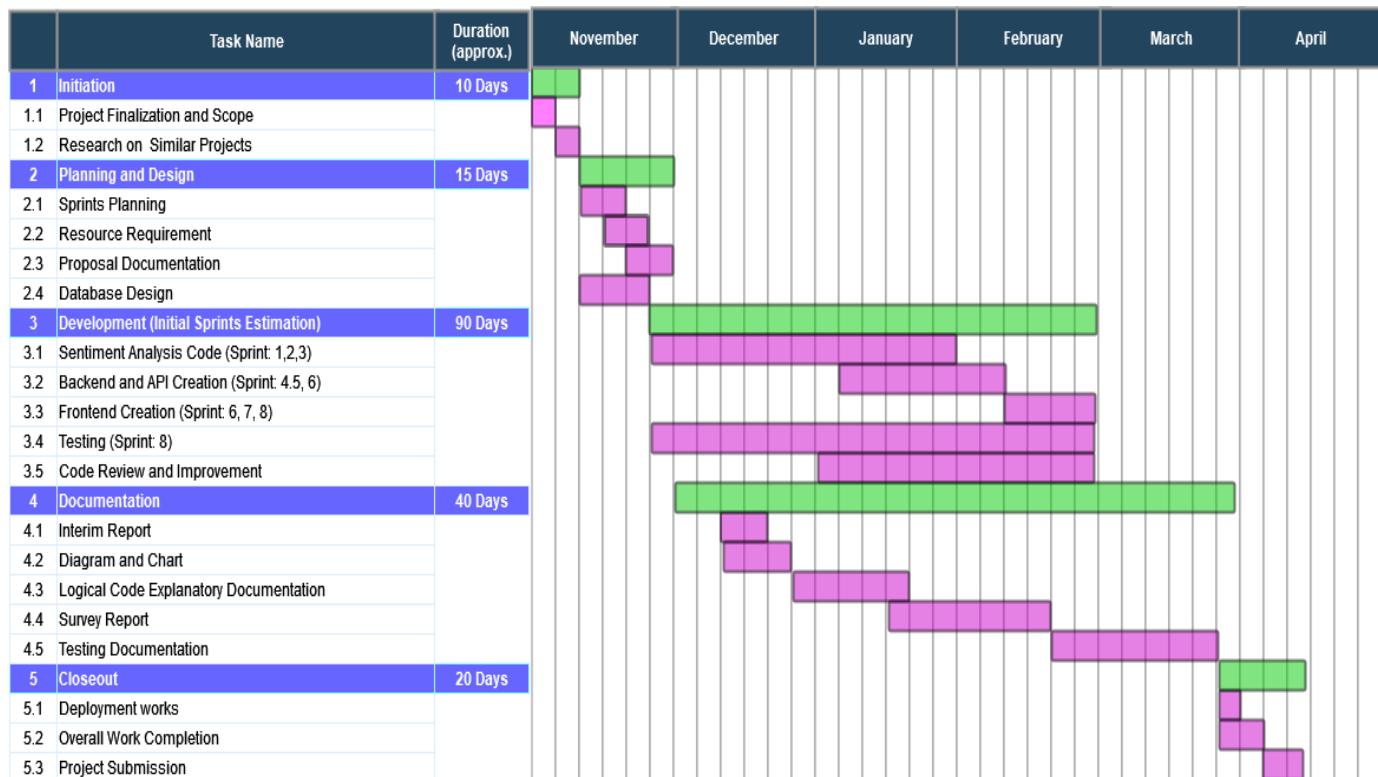


Figure 100 Gantt Chart of the project work division

ii. Work breakdown structure

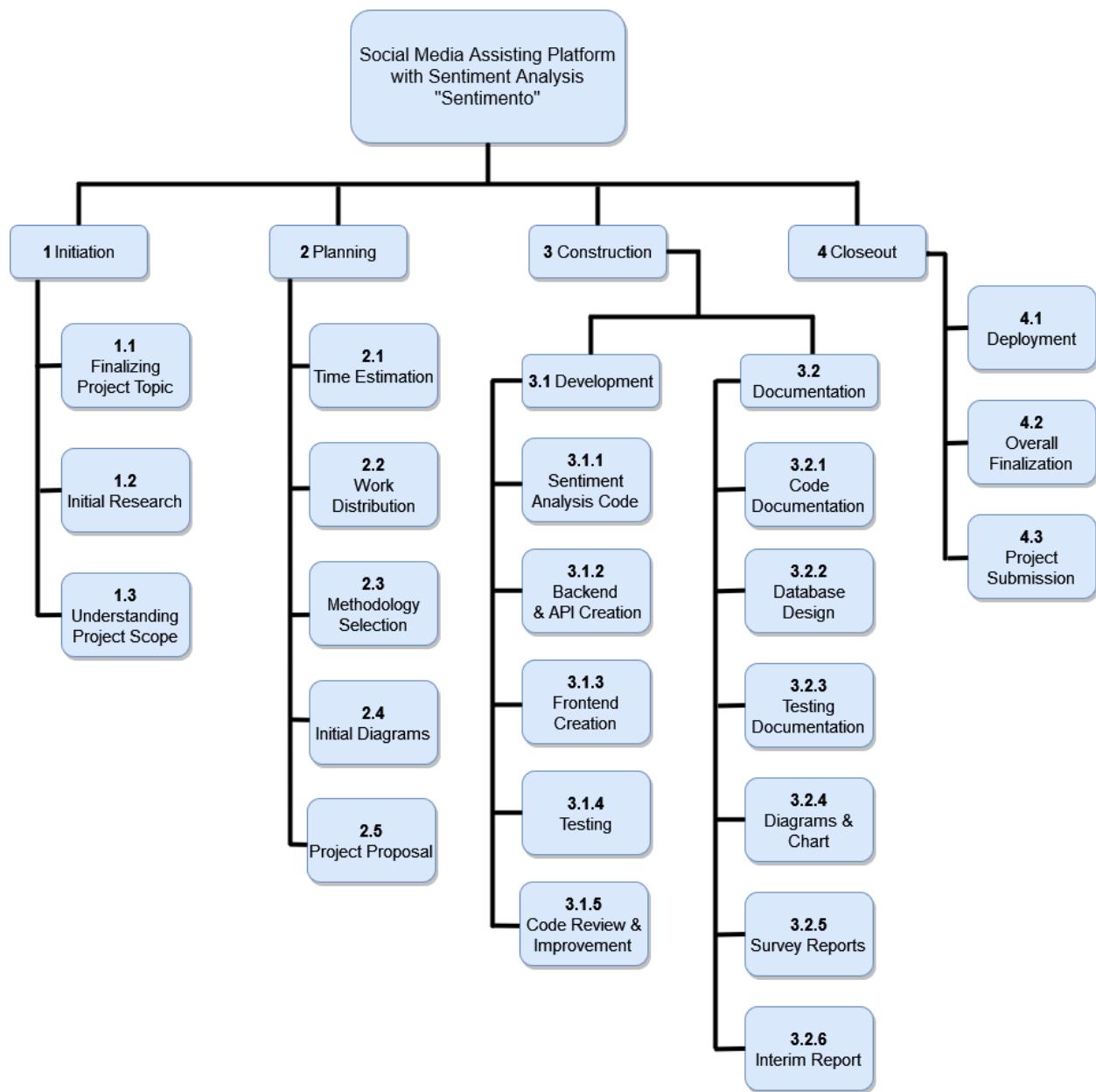


Figure 101 Work breakdown structure of overall project

### iii. Milestone chart

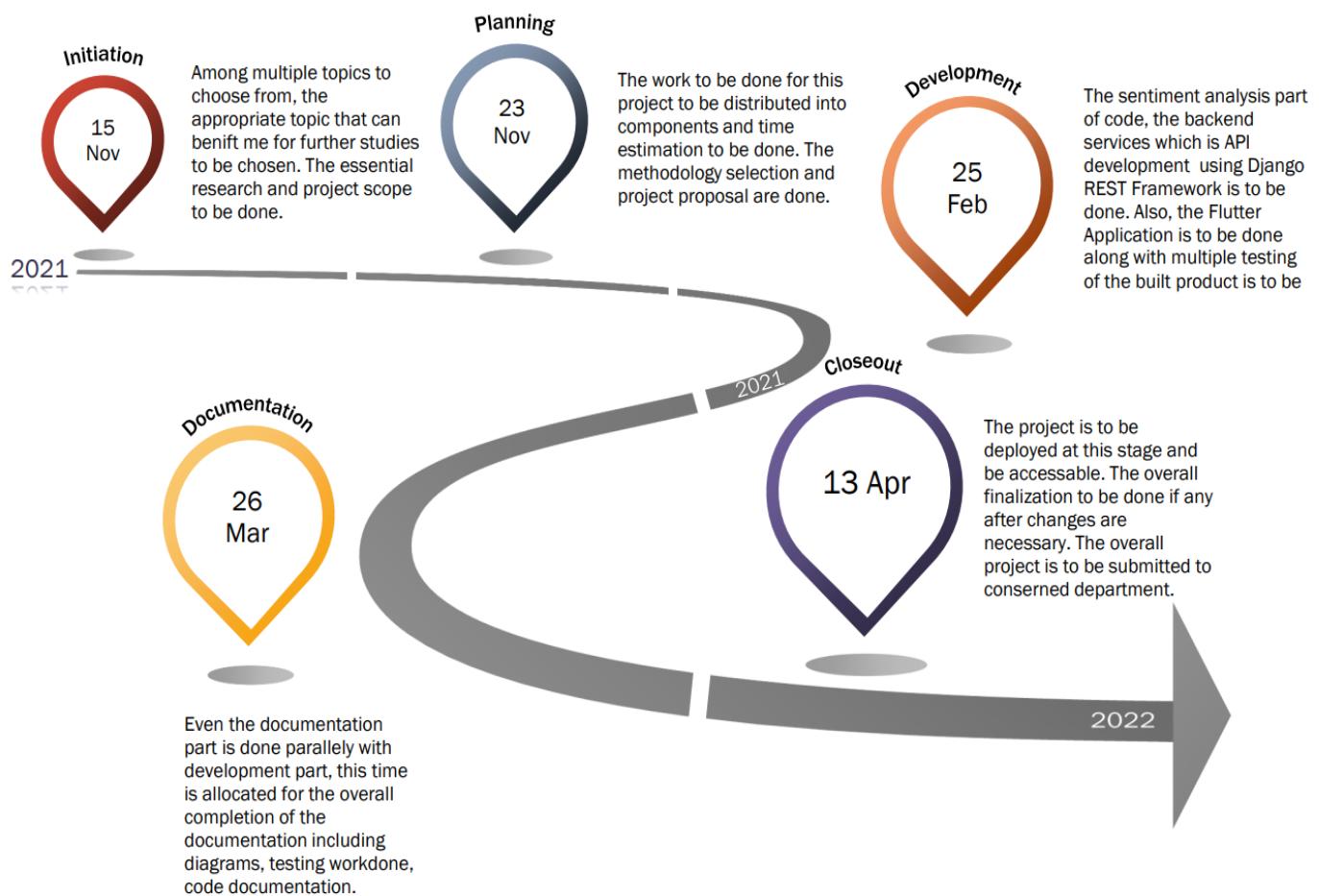


Figure 102 Milestone chart of overall project

iv. Layers in the designed sentiment analysis algorithm

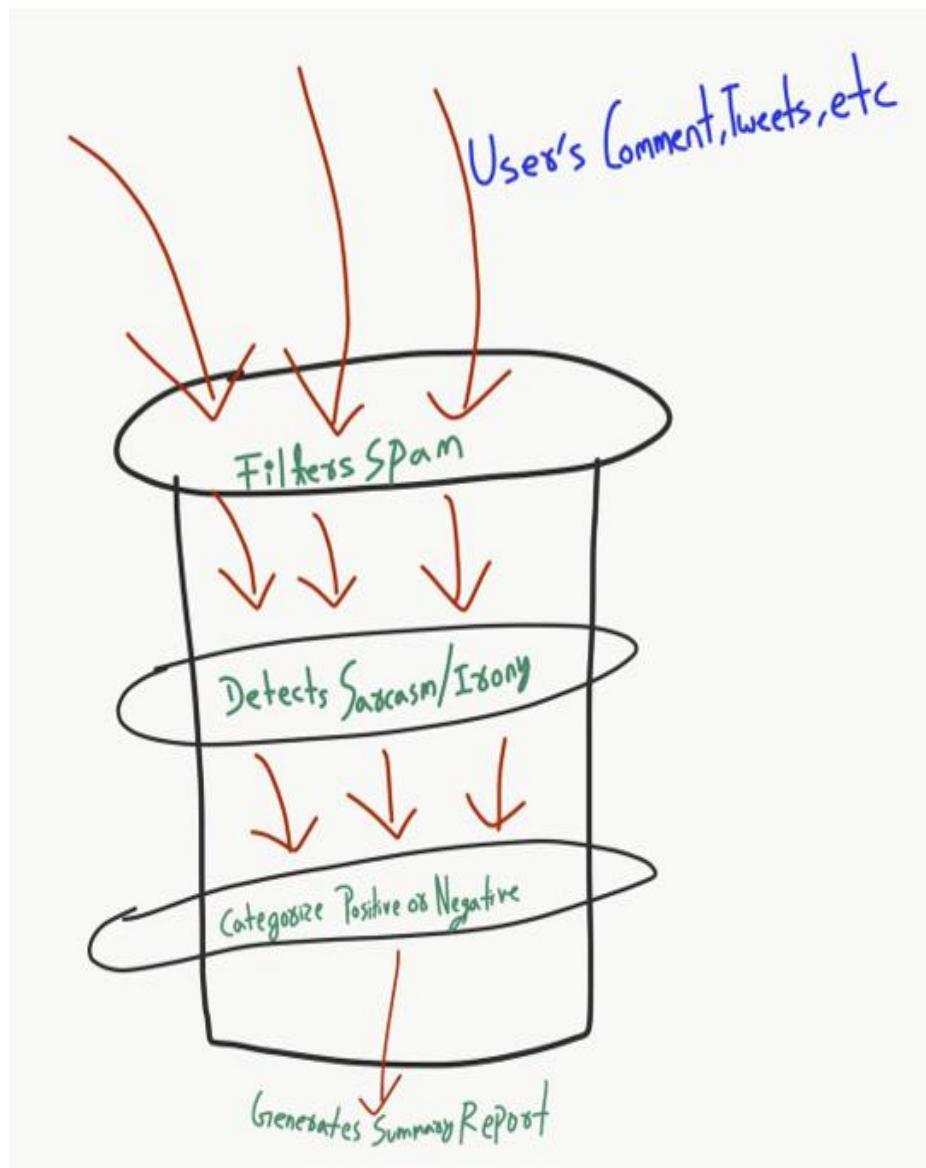


Figure 103 Layers showing how sentiment analysis works

## v. Product Backlog

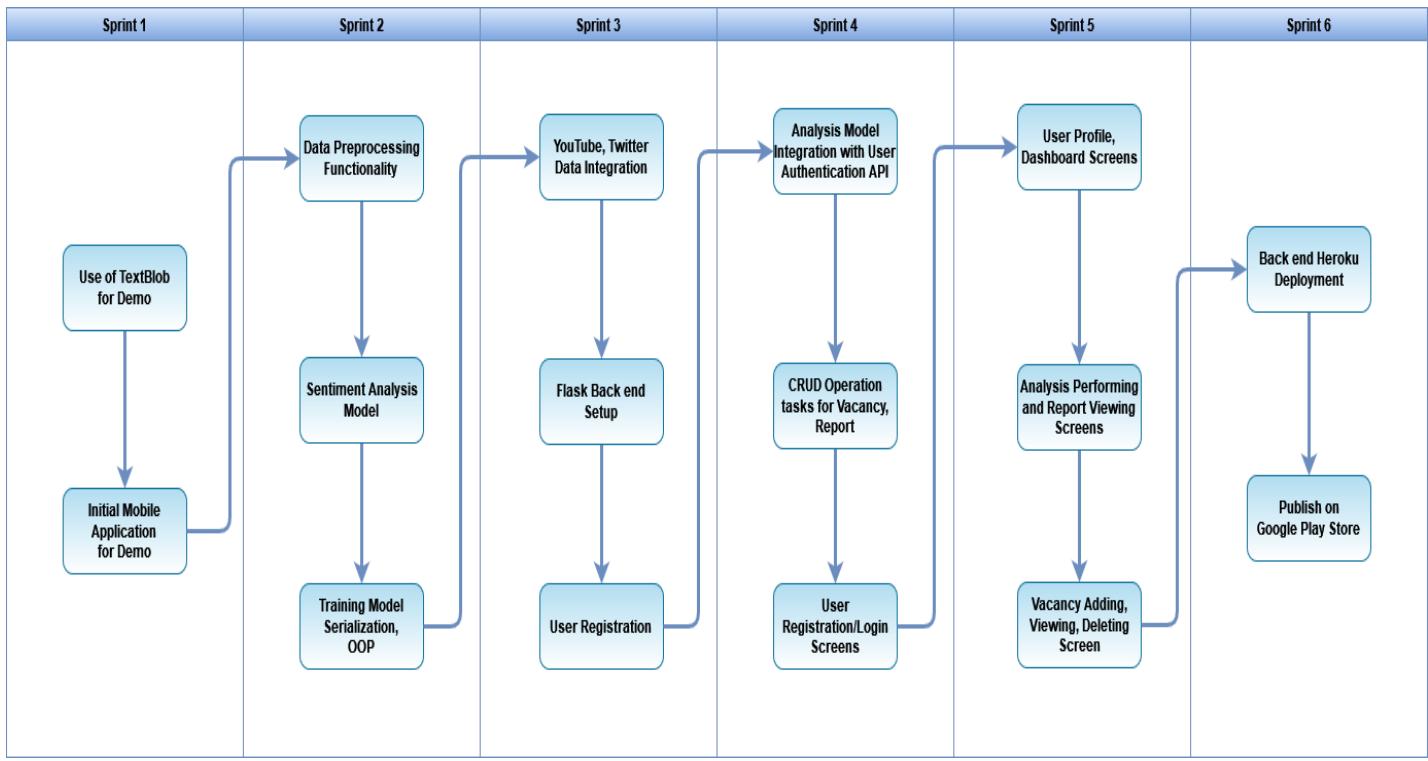


Figure 104 Product backlog design

## vi. Sprint Backlogs

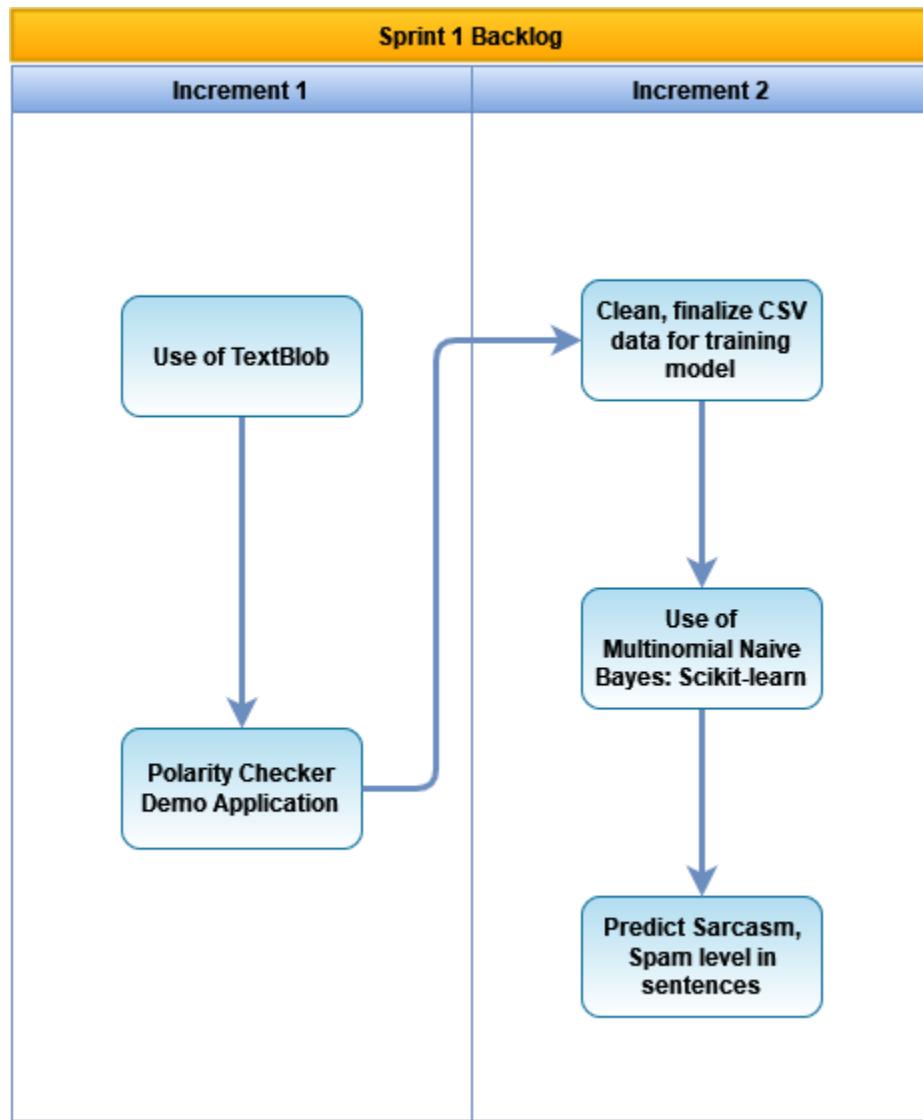


Figure 105 Sprint 1 Backlog

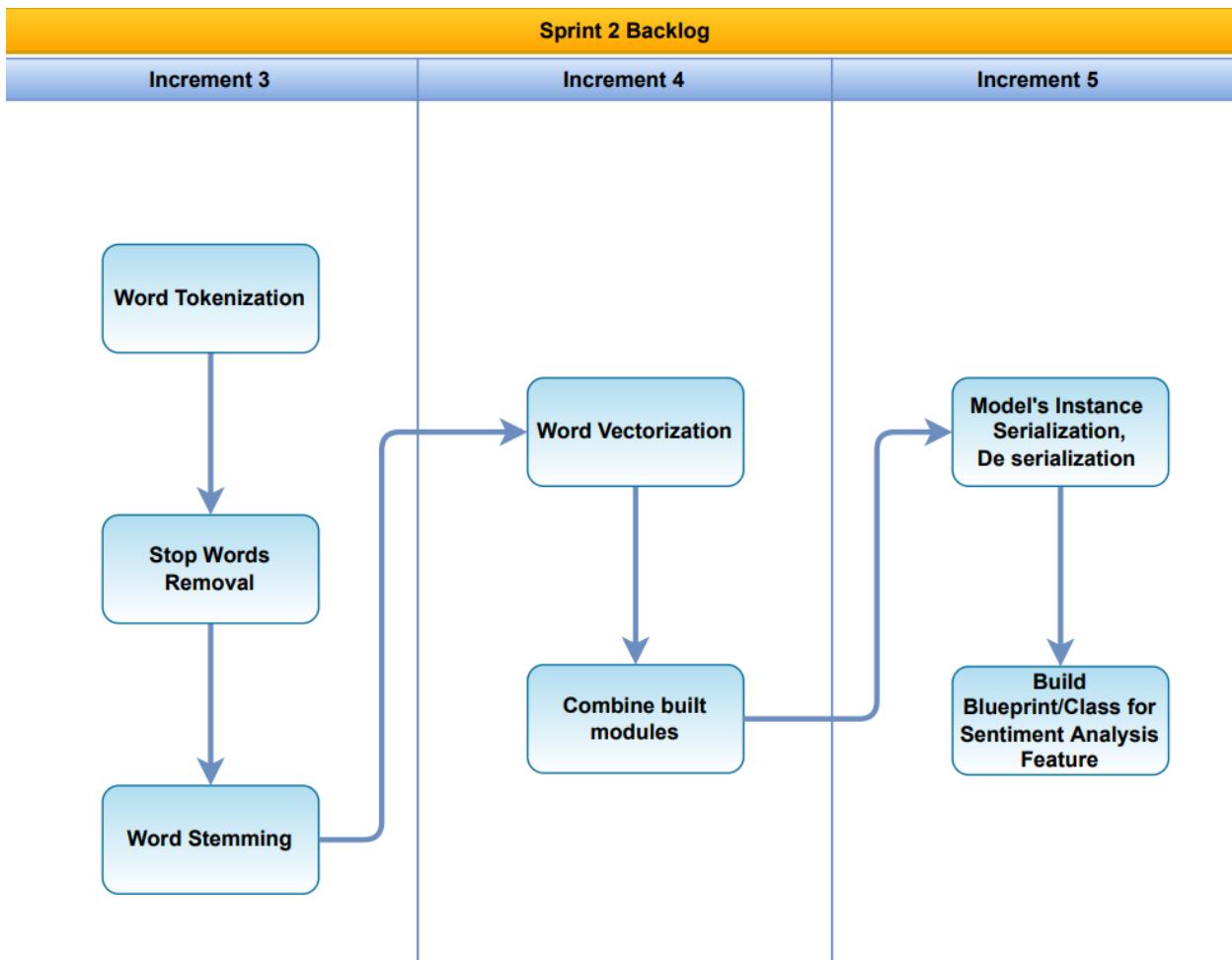


Figure 106 Sprint 2 Backlog

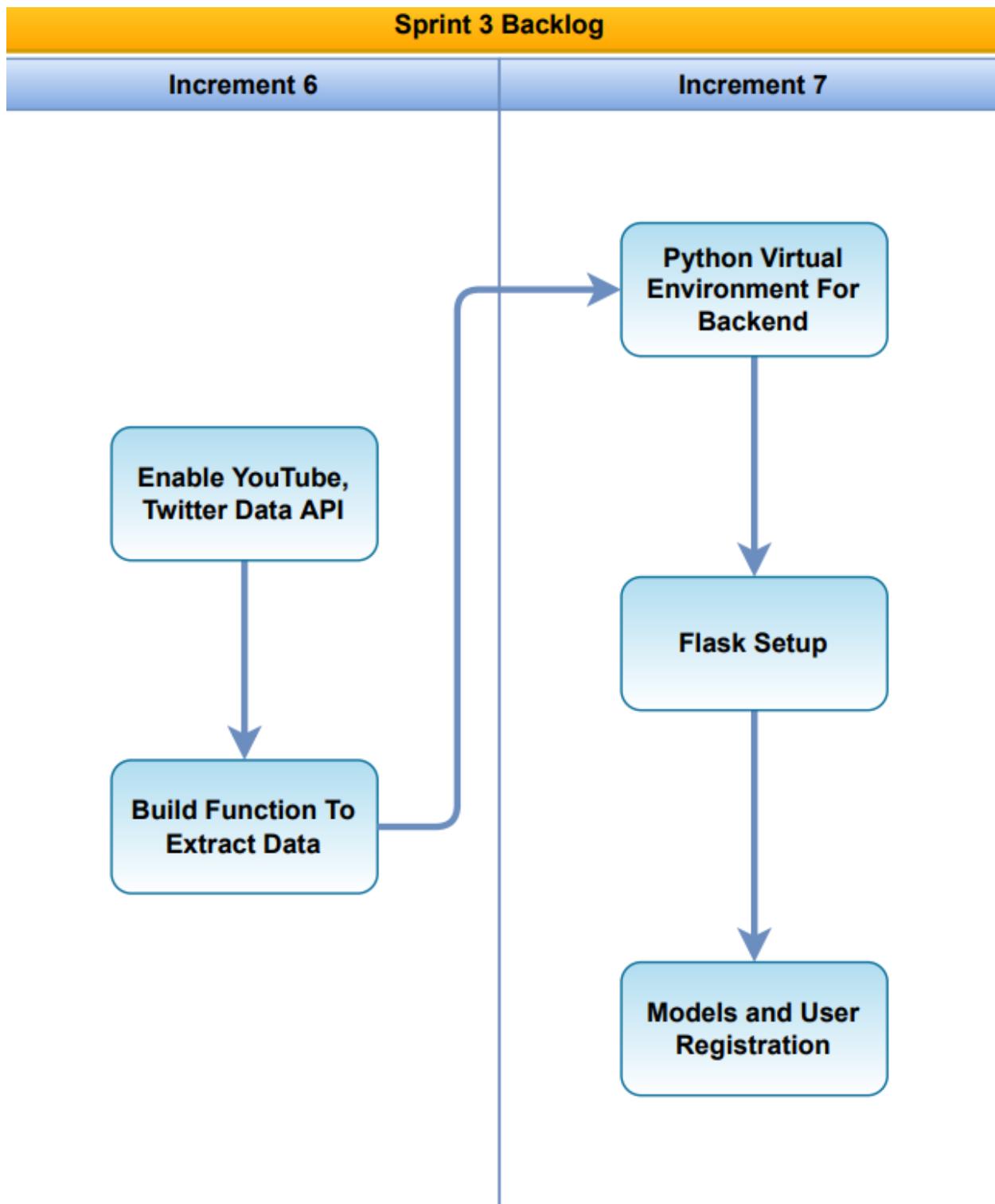


Figure 107 Sprint 3 Backlog

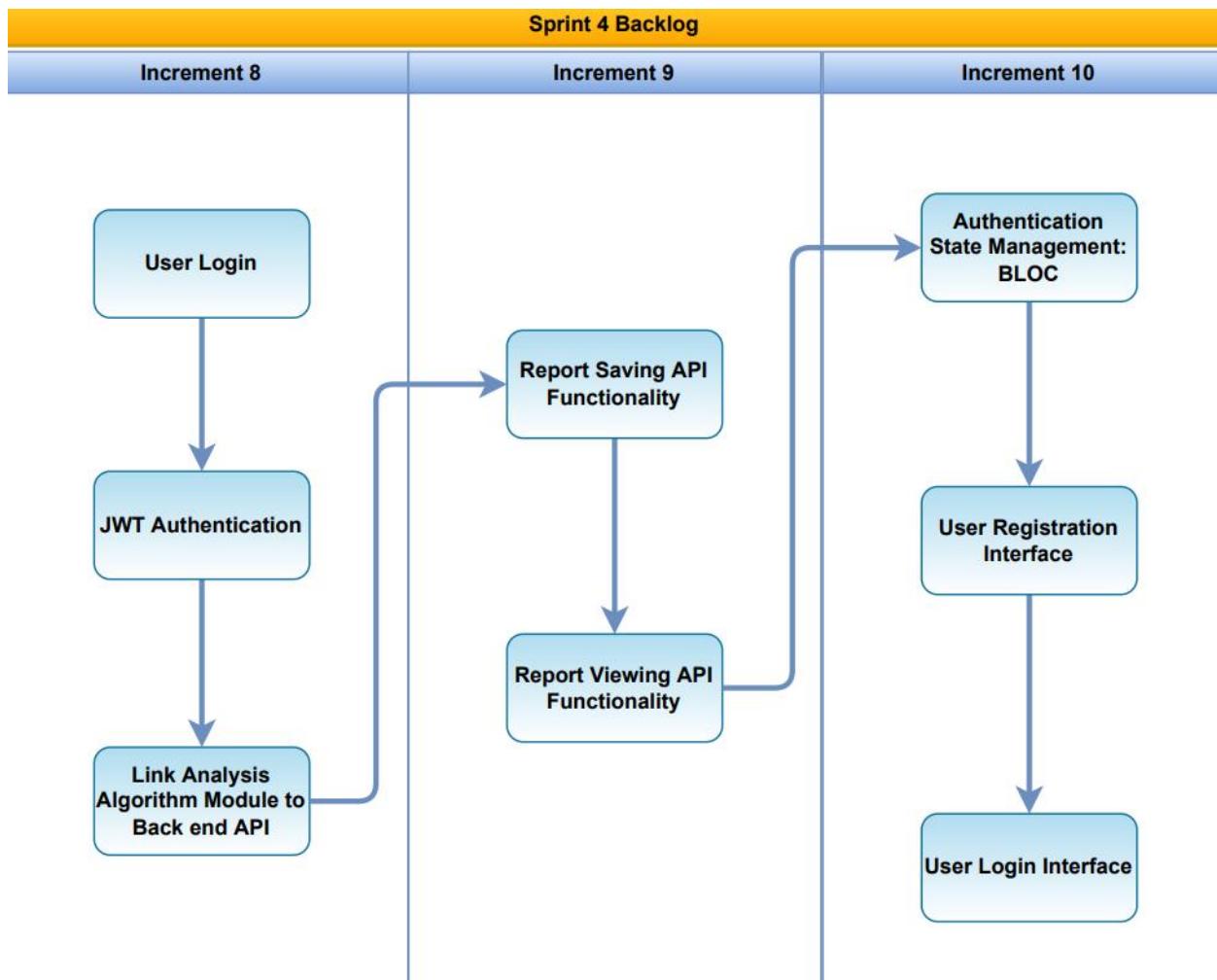


Figure 108 Sprint 4 Backlog

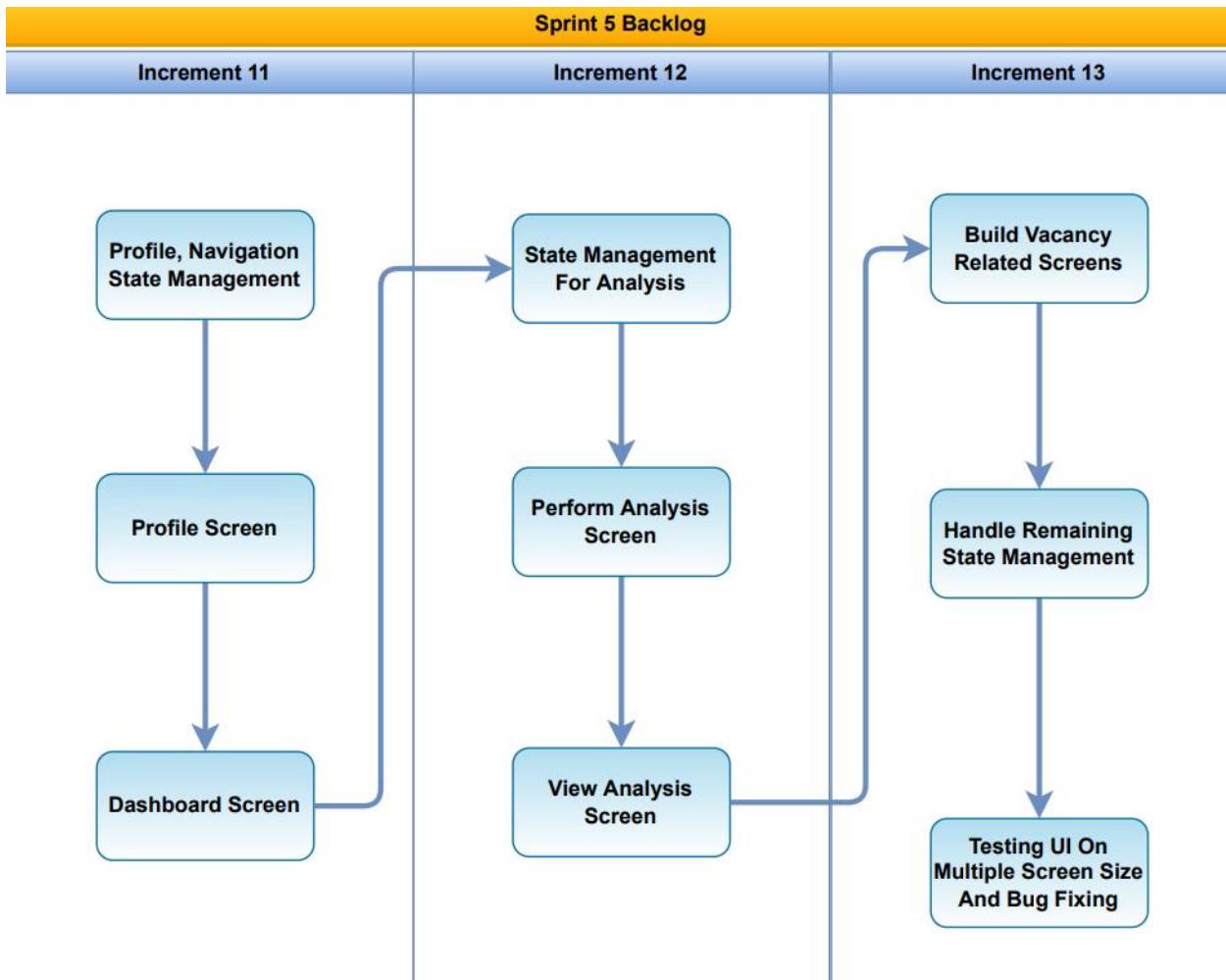


Figure 109 Sprint 5 Backlog

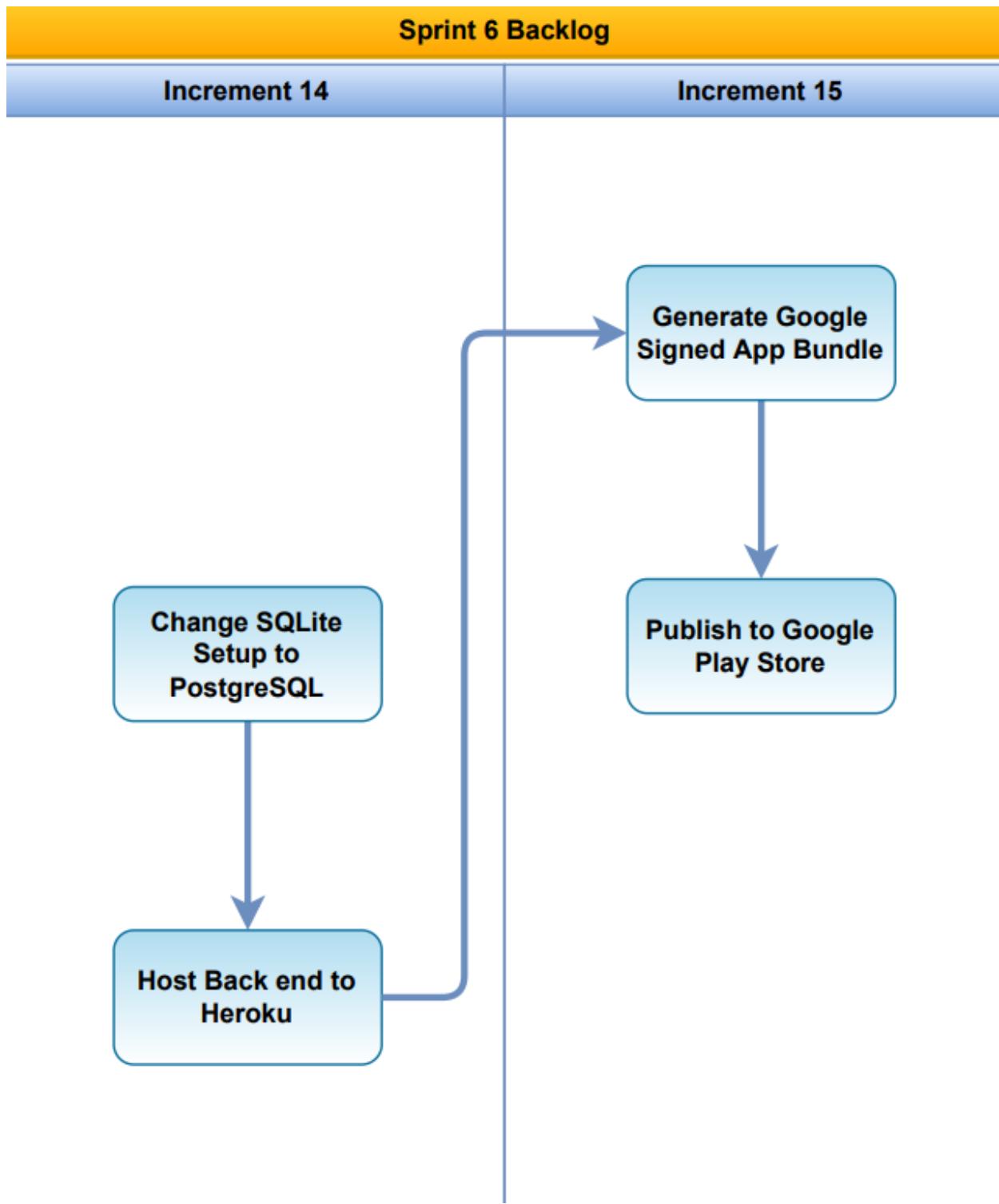


Figure 110 Sprint 6 Backlog

## Appendix 2: Reasons to use consumed libraries/framework

### i. Using BLOC for state management

Business Logic Component (BLOC) is recommended by Google itself for any state management related tasks in Flutter application. There is a need to use state management for quality applications where managing state becomes significant action. There is a need of tracking the current state of the application to navigate through multiple user interfaces and passing data using their models. Multiple developers can work in the same project as workloads are divided into multiple modules. The interface design codes and logical codes are separately written in BLOC pattern (Bloc Library, 2021).

### ii. Using Flask over fully featured framework like Django

For this project, we need backend to develop only API excluding any views like web pages. If Django or other fully fetched framework was used, it would only cause extra space which is not a good practise. And Django provides most of the work done by itself with its REST framework. Using such pre-built method, there would be less learning opportunities. Instead, I found it better to write own functions while using Flask as it is a microframework and there is a need to import other modules as per need.

### iii. Multinomial Naive Bayes algorithm over available package like TextBlob

Packages like TextBlob, Vader has the functionality to calculate polarity by itself. There would be no learning opportunities on supervised machine learning if the project was fully depended on such libraries. So, the plan was made to break down analysis into spam, sarcasm labels. Labelled data models were used from Kaggle (data model source) and using Multinomial Naïve Bayes of Ski-kit learn, the labels of testing datasets were performed. This provided me the learning opportunity and certain level of experience on supervised learning.

## Appendix 3: Software Requirement Specification

**Project: Sentimento**

“Social media assisting platform with sentiment analysis”

**Introduction**

The SRS document is a non-technical report clarifying the features/capabilities of the software, system or application. It contains detailed information on functional and non-functional requirements of the project. This document is prepared identifying and evaluating the client's requirement. If the project is not being built for specific client, the survey feedbacks of the potential users of the system is used to prepare the software requirement specification document (Sommerville, 2011).

The qualities of a clear and in-depth SRS document are as follows (GeeksforGeeks, 2021):

- i. The requirements stated by client or feedbacks from surveys are clearly addressed in the document.

- ii. All the must have functional and non-functional requirements are properly attached.
- iii. Preciseness over ambiguousness is another essential quality for a good SRS document.
- iv. It is better for a SRS document to be easily modifiable as requirements of a project in SCRUM or Agile are prone to have changes or improvements in the further iterations of the project.
- v. The SRS document is also made for client or non-technical viewers. It is a good practise to make this document easily understandable with use of simpler language and terms.

## Purpose

Primary objective of this SRS document is to provide substantial information about the functional and non-functional requirements of the Sentimento platform. This document clearly highlights must include features of the platform to function. The other tool and technologies requirements for the development of the project is also defined in this document.

The core purpose of this mobile application is to provide a user-friendly platform where non-technical and technical (with understanding of Data Analytics) users can perform sentiment analysis within same interface.

### Intended Audiences of this document

This document is primarily intended for the development team of the Sentimento project. Developers from algorithm team, backend and frontend including other team members engaged in technical and non-technical areas are the key readers/viewers of this document. The client of the project or potential users can also read this document to verify their demands are included in the document.

### Why this document for the developer team?

Viewing all the features to include in the platform, developers can identify and plan their moves accordingly. The vocal hearing from the client is not consistent and official but this document with client's approval provides clear vision on what to do and how to implement those essential features in the platform. This document can also be a proof of record of tasks to carry out for both the parties (client and development team).

## Contents

### User's need:

From the feedbacks of pre-survey, a mobile application with easy-to-use interfaces is to be developed where they can just provide video URL or topic of tweet. And just with this little information, users should be able to do their sentiment analysis task. The demand of a social media related job vacancies posting/viewing feature is also considered to include in the system.

### Assumptions and dependencies:

The development work of this project is first divided into four sections and carried out sequentially. The four sections are as follows:

i. Sentiment analysis functionality

Two functions are to be made to extract comments of a chosen video of YouTube and tweets using each of the respective third-party documentation. After that, a function to clean the data is made and using Multinomial Naive Bayes functionality of scikit-learn and training data models, sentiment analysis module is developed.

ii. Backend development

Flask framework will be used to make the application programming interface (API) of the system. The CRUD operations and sentiment analysis module will be integrated with this part of the development and it interacts with the frontend part of the system to complete the overall usefulness of the system.

iii. Frontend development

Flutter, cross-platform application framework is to be used to effectively developed all the features of the designed platform. For state management task, Business Logic Component (BLOC) will be used. The packages from the Flutter community will be utilized to perform basic functionalities quickly and efficiently. The folder structure will be maintained as recognized by Flutter developer community. The user interfaces will be broken down into smaller components to make the particular code easily accessible when needed to make any changes.

iv. Hosting and taking to production level

Heroku, a cloud hosting service is to be used to host the Python backend. The free account with zero payment to make service will be utilized to make the platform's API live.

The mobile application is estimated to publish on Google Play Store if the developed application at the end fulfils all the policy requirements of publishing application on Play Store.

## Requirements

### Functional requirements:

#### i. Primary Features

<u>Req.ID</u>	<u>Requirement Description</u>					
FR. 1	User can register to system with unique and valid registration detail <div style="border: 1px solid black; padding: 5px; text-align: center;">System Requirement</div> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">SR. 1</td><td>User can provide their correct registration information</td></tr> <tr> <td>SR. 2</td><td>Backend responds with registration success or fail according to user provided information</td></tr> </table>		SR. 1	User can provide their correct registration information	SR. 2	Backend responds with registration success or fail according to user provided information
SR. 1	User can provide their correct registration information					
SR. 2	Backend responds with registration success or fail according to user provided information					
FR. 2	User can login to system with their credentials used on registration <div style="border: 1px solid black; padding: 5px; text-align: center;">System Requirement</div> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">SR. 3</td><td>User can provide their correct login credentials</td></tr> <tr> <td>SR. 4</td><td>Backend responds with login fail in case of invalid information or else user will be forwarded to dashboard</td></tr> </table>		SR. 3	User can provide their correct login credentials	SR. 4	Backend responds with login fail in case of invalid information or else user will be forwarded to dashboard
SR. 3	User can provide their correct login credentials					
SR. 4	Backend responds with login fail in case of invalid information or else user will be forwarded to dashboard					
FR. 3	User can perform sentiment analysis on YouTube comments and tweets <div style="border: 1px solid black; padding: 5px; text-align: center;">System Requirement</div> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">SR. 5</td><td>User can provide YouTube video URL or topic of tweet to perform sentiment analysis</td></tr> <tr> <td>SR. 6</td><td>Backend responds with sentiment report if valid input is given by user.</td></tr> </table>		SR. 5	User can provide YouTube video URL or topic of tweet to perform sentiment analysis	SR. 6	Backend responds with sentiment report if valid input is given by user.
SR. 5	User can provide YouTube video URL or topic of tweet to perform sentiment analysis					
SR. 6	Backend responds with sentiment report if valid input is given by user.					

FR. 4	<p>User can view open job vacancies and skill offering from other users of platform</p> <table border="1" data-bbox="349 297 1405 582"> <thead> <tr> <th colspan="2" data-bbox="349 297 458 361">System Requirement</th></tr> </thead> <tbody> <tr> <td data-bbox="349 361 458 477">SR. 7</td><td data-bbox="458 361 1405 477">User can choose either job vacancies or freelancing services to view</td></tr> <tr> <td data-bbox="349 477 458 582">SR. 8</td><td data-bbox="458 477 1405 582">Backend responds with currently available vacancies posted by other users of platform</td></tr> </tbody> </table>	System Requirement		SR. 7	User can choose either job vacancies or freelancing services to view	SR. 8	Backend responds with currently available vacancies posted by other users of platform
System Requirement							
SR. 7	User can choose either job vacancies or freelancing services to view						
SR. 8	Backend responds with currently available vacancies posted by other users of platform						
FR. 5	<p>User can post social media related job vacancies and their skill offering in the platform</p> <table border="1" data-bbox="349 737 1405 1022"> <thead> <tr> <th colspan="2" data-bbox="349 737 458 800">System Requirement</th></tr> </thead> <tbody> <tr> <td data-bbox="349 800 458 916">SR. 9</td><td data-bbox="458 800 1405 916">User can provide information about their vacancy or freelancing skill.</td></tr> <tr> <td data-bbox="349 916 458 1022">SR. 10</td><td data-bbox="458 916 1405 1022">Backend stores the data in the database that is accessible by other users later</td></tr> </tbody> </table>	System Requirement		SR. 9	User can provide information about their vacancy or freelancing skill.	SR. 10	Backend stores the data in the database that is accessible by other users later
System Requirement							
SR. 9	User can provide information about their vacancy or freelancing skill.						
SR. 10	Backend stores the data in the database that is accessible by other users later						
FR. 6	<p>User can view their detailed profile</p> <table border="1" data-bbox="349 1115 1405 1400"> <thead> <tr> <th colspan="2" data-bbox="349 1115 458 1178">System Requirement</th></tr> </thead> <tbody> <tr> <td data-bbox="349 1178 458 1294">SR. 11</td><td data-bbox="458 1178 1405 1294">Profile viewing REST API request can go from frontend to backend with user's authentication detail</td></tr> <tr> <td data-bbox="349 1294 458 1400">SR. 12</td><td data-bbox="458 1294 1405 1400">Backend responds with user's profile data if authentic request is provided.</td></tr> </tbody> </table>	System Requirement		SR. 11	Profile viewing REST API request can go from frontend to backend with user's authentication detail	SR. 12	Backend responds with user's profile data if authentic request is provided.
System Requirement							
SR. 11	Profile viewing REST API request can go from frontend to backend with user's authentication detail						
SR. 12	Backend responds with user's profile data if authentic request is provided.						
FR. 7	<p>User can save and view their past sentiment analysis reports</p> <table border="1" data-bbox="349 1469 1405 1776"> <thead> <tr> <th colspan="2" data-bbox="349 1469 458 1533">System Requirement</th></tr> </thead> <tbody> <tr> <td data-bbox="349 1533 458 1649">SR. 13</td><td data-bbox="458 1533 1405 1649">User can have the option to save sentiment report for further uses.</td></tr> <tr> <td data-bbox="349 1649 458 1776">SR. 14</td><td data-bbox="458 1649 1405 1776">Backend stores the report in the database.</td></tr> </tbody> </table>	System Requirement		SR. 13	User can have the option to save sentiment report for further uses.	SR. 14	Backend stores the report in the database.
System Requirement							
SR. 13	User can have the option to save sentiment report for further uses.						
SR. 14	Backend stores the report in the database.						

ii. Authentication

Bearer authentication tokens will be used by the system to authenticate user's request to backend.

iii. Reporting/Documentation requirements

Use case diagram, system architecture, flowchart and similar diagrams are designed before the start of development of the project.

Non-functional requirements:

i. Performance: To reduce the memory usage and shorten sentiment analysis algorithm processing time, the machine learning model is to be serialized into pickle file.

ii. Scalability: The backend of the project is to be hosted in cloud for larger scale accessibility of the project.

iii. Security: For user's security reasons, the platform should not store third party access tokens. Considering it, the tokens will be locally saved in user's device.

## Approval

The platform is designed and developed to fit all the potential users like data analyst, marketing and news media professionals and social media related job seeker/provider. It clarifies that this project is not made for any particular client. The platform is developed keeping in mind to fulfil the professional needs of potential users. The user interface and database architecture are structured for all the general users of the platform.

This project is designed to suit the interest of multiple or large-scale users, there is no any particular client of this project to approve this software requirement specification. Instead, our supervisors of the project have approved the proposed system to carry out the development works of this project.

## Appendix 4: User Feedbacks

## i. Google Play Store Reviews

*User reviews***Samira Kharel**

★★★★★ 17 April 2022



Very nice user interface and very efficient app to perform sentiment analysis on tweets that I need in my day to day activity.

**Saurab Tharu**

★★★★★ 12 April 2022



Great app with nice UI Wish you all the best team for further improvements

**Bimsara Shrestha**

★★★★★ 19 April 2022



All the functionalities works smoothly. Thank you for providing such valuable app for free.

**sudip shrestha**

★★★★★ 19 April 2022



Vry useful with simple to use UI

**Keshab Acharya**

★★★★★ 20 April 2022



I feel lazy reading YouTube comments but I want to know what people are saying. This app is just perfect for my need to know what majority of people are saying about the video. Thank you developer team for making such wonderful app for free.

**Benay kafle**

★★★★★ 19 April 2022



Simple but usefull application.

**Sandesh Shrestha**

★★★★★ 19 April 2022



Useful

**Seleshma Prasai**

★★★★★ 20 April 2022



Incredible exprience, Amazing application

Figure 111 User feedback from Google Play Store reviews

## ii. User Feedback Survey Report

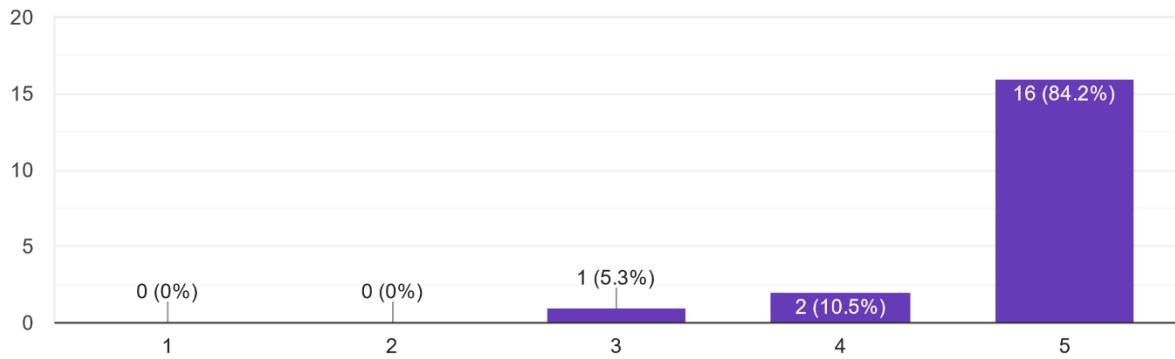
Are you happy that there is alternative of API tokens by using coupons?

19 responses



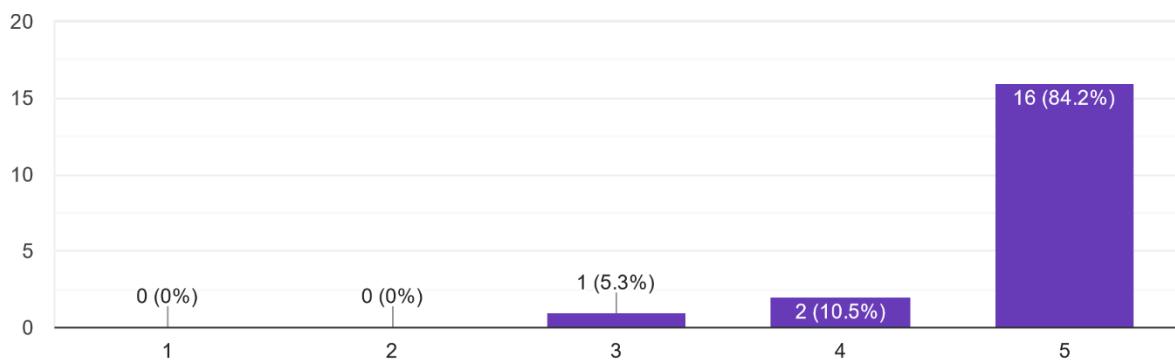
Rate the user interface for profile

19 responses



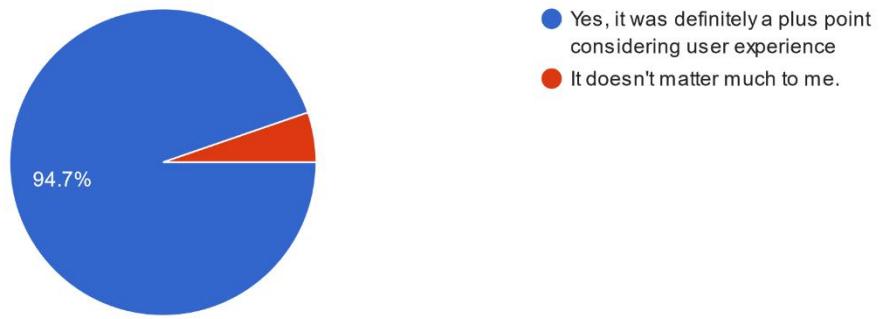
## Rate the data visualization of the application

19 responses



## Is showing video thumbnail while performing YouTube analysis better?

19 responses



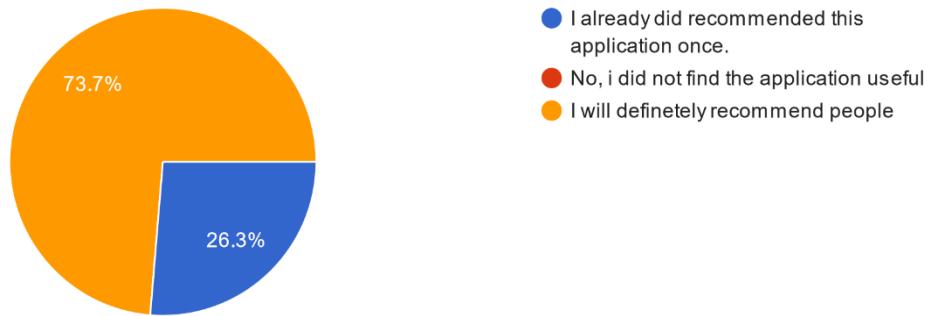
Are you satisfied that user can only have five open vacancies at a time?

19 responses



Will you recommend this application to anyone who likes to perform sentiment analysis on social media data?

19 responses



## Appendix 5: Weekly/Increment task information for development

### (15<sup>th</sup> Increment)

Week20: Published Flutter application to Google Play Store.

### (14th Increment)

Week19: Database migrated from SQLite to PostgreSQL, Deployed API on Heroku, made live and stable

### (13th Increment)

Week18: Vacancy viewing/adding/deleting UI feature and their bloc, fixes widget pixel oversizing issue in smaller width devices

### (12th Increment)

Week17: YouTube, Twitter, manual sentiment analysis UI, report saving feature and their bloc

### (11th Increment)

Week16: Profile, Home Page Menus, Keys Manage and their bloc

### (10th Increment)

Week15: Login/Signup Screens with auth BLOC and API request handler for auth task

### (9th Increment)

Week14: Implemented sentiment report and vacancy (saving/viewing) backend API part

(8th Increment)

Week13: User Login, JWT authentication to allow only authenticated users to perform sentiment analysis, YouTube/Twitter sentiment algorithm linked

(7th Increment)

Week12: Flask backend setup and done up to user registration

(6th Increment)

Week11: Data integration part code using YouTube, Twitter API documentation

(5th Increment)

Week10: Implemented pickle (serializing trained ML model) concept in my system that increases calculation speed

(4th Increment)

Week9: Made a working sentiment analysis model that works by providing both training, testing data

(3rd Increment)

Week8: Worked on data pre-processing like tokenizing, stemming, stop words part

(2nd Increment)

Week7: Started to write code for development work after initial level research and learning NLTK, TextBlob

(1st Increment)

Week6: Initial UI prototype is made ready to implement it properly later after completion of backend work

Week1-5: Research, interim report, brainstorming on development work, gathering development work guidance source

## Appendix 6: Essential code snippets

## i. Backend

```

from flask import Flask
import os
from flask_jwt_extended import JWTManager
from src.register import register_blueprint
from src.login import login_blueprint
from src.report import report_blueprint
from src.profile import profile_blueprint
from src.vacancy import vacancy_blueprint
from src.youtube import youtube_blueprint
from src.twitter import twitter_blueprint
from src.manual import manual_blueprint
from src.database import db

def create_app(test_config=None):
    app = Flask(__name__, instance_relative_config=True)

    if test_config is None:
        app.config.from_mapping(
            SECRET_KEY=os.environ.get("SECRET_KEY"),
            SQLALCHEMY_DATABASE_URI=os.environ.get("DATABASE_URI"),
            SQLALCHEMY_TRACK_MODIFICATIONS=False,
            JWT_SECRET_KEY=os.environ.get('JWT_SECRET_KEY')
        )
    else:
        app.config.from_mapping(test_config)

    app.register_blueprint(register_blueprint)
    app.register_blueprint(login_blueprint)
    app.register_blueprint(report_blueprint)
    app.register_blueprint(profile_blueprint)
    app.register_blueprint(vacancy_blueprint)
    app.register_blueprint(youtube_blueprint)
    app.register_blueprint(twitter_blueprint)
    app.register_blueprint(manual_blueprint)

    # decorator for index page of the backend
    @app.route("/")

```

```

def index():
    return {
        "Registration": "/register/ [POST]",
        "Login": "/login/ [POST]",
        "User profile": "/profile/ [GET]",
        "YouTube comments analysis": "/red/ [POST]",
        "Tweets analysis": "/blue/ [POST]",
        "User reports": "/report/ [GET/POST]",
        "Vacancy": "/vacancy/ [GET/POST/DELETE] &
/vacancy/all?filter=freelancer or vacancy [GET]",
        "Sentence/Paragraph polarity": "/manual/ [POST]"
    }

db.app = app
db.init_app(app)

JWTManager(app)
# with app.app_context():
#     db.create_all()
return app

```

The above code file is the backend logic that listens the requests received from API

```

from flask_sqlalchemy import SQLAlchemy
from datetime import datetime

db = SQLAlchemy()

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(55), nullable=False)
    email = db.Column(db.String(80), unique=True, nullable=False)
    password = db.Column(db.Text(), nullable=False)
    purpose = db.Column(db.String(30), nullable=False)
    analysis_count = db.Column(db.Integer, default=0)
    vacancy_count = db.Column(db.Integer, default=0)
    reports = db.relationship('Report', backref="user")
    vacancies = db.relationship('Vacancy', backref="user")

    def __init__(self, **kwargs):
        super().__init__(**kwargs)

    def __repr__(self) -> str:

```

```

        return 'User>>> {self.email}'

class Report(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    analysis_datetime = db.Column(db.DateTime, default=datetime.now())
    platform = db.Column(db.String(20), nullable=False)
    data = db.Column(db.String(200), nullable=False)
    data_count = db.Column(db.Integer, nullable=False)
    spam_count = db.Column(db.Integer, nullable=False)
    sarcasm_count = db.Column(db.Integer, nullable=False)
    positive_count = db.Column(db.Integer, nullable=False)
    negative_count = db.Column(db.Integer, nullable=False)
    neutral_count = db.Column(db.Integer, nullable=False)
    overall_polarity = db.Column(db.String(6), nullable=False)

    def __init__(self, **kwargs):
        super().__init__(**kwargs)

    def __repr__(self) -> str:
        return 'Report>> {self.data}'


class Vacancy(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    title = db.Column(db.String(50), nullable=False)
    contact_email = db.Column(db.String(80), nullable=False)
    contact_number = db.Column(db.String(20), nullable=False)
    description = db.Column(db.String(600), nullable=False)
    work_hr = db.Column(db.Integer, nullable=False)
    wage_hr = db.Column(db.Text, nullable=False)
    is_negotiable = db.Column(db.Boolean, default=True, nullable=False)
    is_vacancy = db.Column(db.Boolean, default=True, nullable=False)

    def __init__(self, **kwargs):
        super().__init__(**kwargs)

    def __repr__(self) -> str:
        return 'Vacancy>> {self.title}'

```

The above code file contains all the three models of the system

```

from flask import Blueprint
from flask_jwt_extended import jwt_required, get_jwt_identity
from src.database import User
from src.functions.field_validators import success_false
from src.senti_bot import notifyProfileView

profile_blueprint = Blueprint('profile', __name__)

@profile_blueprint.route('/profile/', methods=['GET'])
@jwt_required()
def profile():
    try:
        user_email = get_jwt_identity()
        current_user = User.query.filter_by(email=user_email).first()
        return {
            "success": "true",
            "name": current_user.name,
            "email": current_user.email,
            "purpose": current_user.purpose,
            "analysis_count": str(current_user.analysis_count),
            "vacancy_count": str(current_user.vacancy_count)}
    except:
        return success_false()

```

The above code file handles the user's profile viewing request

```

from flask import Blueprint, request
from werkzeug.security import generate_password_hash
from src.database import User, db
from src.functions.field_validators import validate_email, success_false
from flask_jwt_extended import create_access_token

from src.senti_bot import notifyRegisterToAdmin, notifyRegisterTry

register_blueprint = Blueprint('register', __name__)

@register_blueprint.route('/register/', methods=['POST'])
def register():
    request_data = request.get_json()
    try:
        name = request_data['name']
        email = request_data['email']

```

```

password = request_data['password']
purpose = request_data['purpose']

# to carry further process only if data in post requests are valid
# first will check if any unwanted format of data is provided
if len(name) > 55:
    return success_false(msg="Name should be less than 55chars")
if len(name) < 5:
    return success_false(msg="Name should be at least 5chars")
if len(password) < 6:
    return success_false(msg="Password should be at least 6chars")
if len(email) > 80:
    return success_false(msg="Email should be less than 80chars")
if len(purpose) > 30:
    return success_false(msg="Purpose should be less than 30chars")
if not validate_email(email):
    return success_false(msg="Please enter a valid email address")

# now moving to database querying if provided registration data format is
correct
if User.query.filter_by(email=email).first() is not None:
    # if user with this email already exist
    return success_false(msg="This email is already in use")

# if registration request is valid, moving to registration stage from
here
hashed_password = generate_password_hash(password)
user_to_register = User(name=name, email=email,
                        password=hashed_password, purpose=purpose)
# inserting user data to user table
db.session.add(user_to_register)
db.session.commit() # saving database instance

access_token = create_access_token(
    identity=email, expires_delta=False)

# responding user that user is now registered
return {"success": "true", "msg": "User is now registered", "email": email, "name": name, "access_token": access_token}

except:
    return success_false()

```

The above code file handles the user's register request

```

from flask import Blueprint, request
from flask_jwt_extended import jwt_required, get_jwt_identity
from src.database import User, db
from src.algo.extract_data import youtube_comments_extract
from src.algo.Sentimento import Sentimento
from src.senti_bot import notifyYouTube

youtube_blueprint = Blueprint('youtube', __name__)

@youtube_blueprint.route('/red/', methods=['POST'])
@jwt_required()
def youtube_sentiment():
    request_data = request.get_json()
    try:
        user_email = get_jwt_identity()
        current_user = User.query.filter_by(email=user_email).first()
        video_id = request_data['video_id']
        api_key = request_data['api_key']
        approx_comments = 0 # declaring outside so that it can be used at this
space
        try:
            approx_comments = int(request_data['approx_comments'])
        except:
            return {"success": "false", "msg": "Approx comments should be
numeric"}
    if (len(video_id) < 5 or len(video_id) > 50):
        return {"success": "false", "msg": "Please provide valid video
source"}
    if (len(api_key) < 5 or len(api_key) > 250):
        return {"success": "false", "msg": "Please provide valid key/coupan"}
    if ((approx_comments < 20) or (approx_comments > 1000)):
        return {"success": "false", "msg": "Comments between 20 to 1000 can
only be set at the moment"}

    # after post req validation, moving to youtube sentiment analysis
    youtube_data = youtube_comments_extract(
        video_id, api_key, approx_comments)

    # making object of sentiment result
    sentiment = Sentimento(youtube_data[0])

```

```
# after getting sentiment result, need to increase user's analysis count
record

    current_user.analysis_count += 1
    db.session.commit()

    # after adding the user's analysis count, need to return sentiment result
    to user
    return {
        "success": "true",
        "msg": "Analysed given data successfully",
        "title": youtube_data[2],
        "thumbnail": youtube_data[1],
        "total_data": sentiment.data_count(),
        "spam_count": sentiment.layer_count()[0],
        "sarcasm_count": sentiment.layer_count()[1],
        "positive_count": sentiment.overall_polarity()["positive_count"],
        "negative_count": sentiment.overall_polarity()["negative_count"],
        "neutral_count": sentiment.overall_polarity()["neutral_count"],
        "overall_polarity":
round((sentiment.overall_polarity()["overall_polarity"]), 3)
    }

except:
    return {"success": "false", "msg": "Invalid API request"}
```

The above code file handles user's YouTube comment sentiment analysis request

## ii. Frontend

```

import 'package:flutter/material.dart';
import 'package:flutter_neumorphic/flutter_neumorphic.dart';
import 'package:flutter_secure_storage/flutter_secure_storage.dart';
import 'package:sentimento/screens/analysis_screen/comments_screen.dart';
import 'package:sentimento/screens/analysis_screen/manual_analysis.dart';
import 'package:sentimento/screens/analysis_screen/tweets_screen.dart';
import 'package:sentimento/screens/authscreen_controller.dart';
import 'package:sentimento/screens/getstarted_screen/getstarted_screen.dart';
import 'package:sentimento/screens/home_screen/home_screen.dart';
import 'package:sentimento/screens/login_screen/login_screen.dart';
import 'package:sentimento/screens/mynetwork_screen/post_service.dart';
import 'package:sentimento/screens/mynetwork_screen/post_vacancy.dart';
import 'package:sentimento/screens/mynetwork_screen/user_posts.dart';
import 'package:sentimento/screens/mynetwork_screen/view_freelancers.dart';
import 'package:sentimento/screens/mynetwork_screen/view_vacancies.dart';
import 'package:sentimento/screens/profile_screen/profile_screen.dart';
import 'package:sentimento/screens/register_screen/register_screen.dart';
import 'package:sentimento/screens/reports_screen/reports_screen.dart';
import 'package:sentimento/utilities/routes.dart';

FlutterSecureStorage? secureStorage;

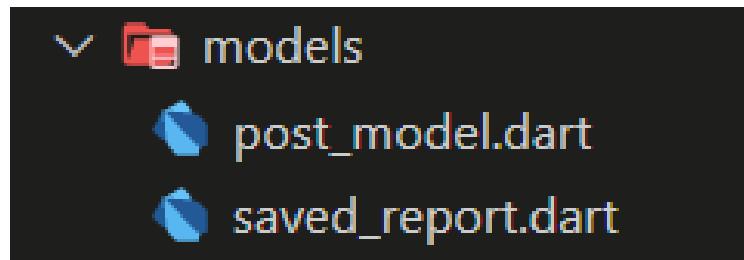
void main() {
    //making single instance so that it can be used again by other methods from
    //multiple pages
    secureStorage = const FlutterSecureStorage();

    runApp(MaterialApp(
        debugShowCheckedModeBanner: false,
        theme: ThemeData(primarySwatch: Colors.teal, brightness: Brightness.light),
        home: const AuthScreenController(),
        routes: {
            AppRoutes.getstartedscreen: (context) => const GetStartedScreen(),
            AppRoutes.loginscreen: (context) => const LoginScreen(),
            AppRoutes.registerscreen: (context) => const RegisterScreen(),
            AppRoutes.homescreen: (context) => const HomeScreen(),
            AppRoutes.profilescreen: (context) => const ProfileScreen(),
            AppRoutes.commentsAnalysis: (context) => const CommentsScreen(),
            AppRoutes.tweetsAnalysis: (context) => const TweetsScreen(),
            AppRoutes.manualAnalysis: (context) => const ManualAnalysis(),
            AppRoutes.reports: (context) => const ReportsScreen(),
        }
    ));
}

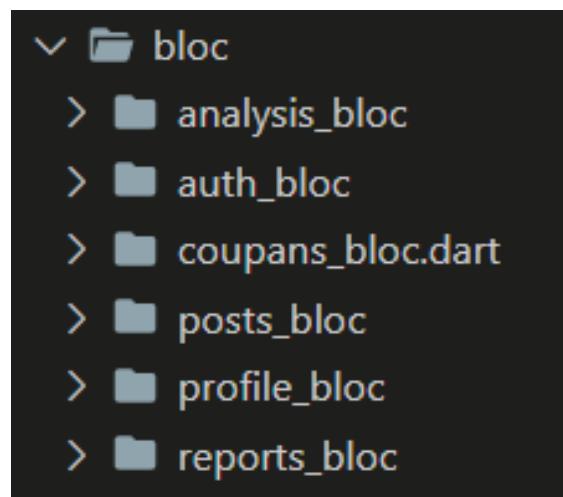
```

```
AppRoutes.myPosts: (context) => const UserPosts(),
AppRoutes.viewVacancy: (context) => const ViewVacancies(),
AppRoutes.viewFreelancer: (context) => const ViewFreelancers(),
AppRoutes.postVacancy: (context) => const PostVacancy(),
AppRoutes.postSkill: (context) => const PostService()
},
));
}
```

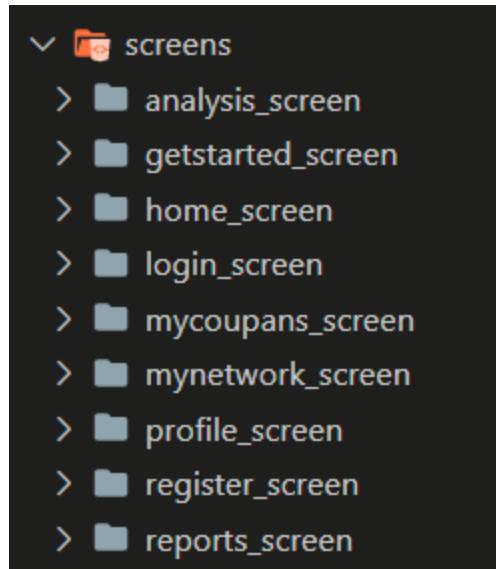
Code content of main.dart file



Models used in built Flutter application



Used state management contents



Screens used in the Flutter application

```

import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:sentimento/utilities/securestorage.dart';

const String domain = "https://sentimentofyp.herokuapp.com";

const requestHeader = <String, String>{
  'Content-Type': 'application/json; charset=UTF-8',
};

//to get response from backend (POST request)
getResponse(endPoint, header, body) async {
  final response =
    await http.post(Uri.parse(endPoint), headers: header, body: body);
  return response;
}

//to get response from JWT protected POST request
getPostResponse(endPoint, body) async {
  String? token = await secureStorage?.read(key: 'access-token');
  var headerwithToken = <String, String>{
    'Content-Type': 'application/json; charset=UTF-8',
    'Authorization': 'Bearer $token',
  };
  final response = await http.post(Uri.parse(endPoint),

```

```

        headers: headerwithToken, body: body);
    return response;
}

//to get (GET request)
getGETResponse(endPoint) async {
    String? token = await secureStorage?.read(key: 'access-token');
    var headerwithToken = <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
        'Authorization': 'Bearer $token',
    };
    final response =
        await http.get(Uri.parse(endPoint), headers: headerwithToken);
    return response;
}

//to handle post/vacancy delete request of user
getDELETEResponse(endPoint) async {
    String? token = await secureStorage?.read(key: 'access-token');
    var headerwithToken = <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
        'Authorization': 'Bearer $token',
    };
    final response =
        await http.delete(Uri.parse(endPoint), headers: headerwithToken);
    return response;
}

//to respond back to bloc
blocResponder(response) {
    if (response.statusCode == 200) {
        var decodedResponse = json.decode(response.body);
        return decodedResponse;
    }
}

```

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:sentimento/bloc/auth_bloc/auth_bloc.dart';
import 'package:sentimento/screens/getstarted_screen/getstarted_screen.dart';
import 'package:sentimento/screens/home_screen/home_screen.dart';
import 'package:sentimento/widgets/minor_ui_parts.dart';

```

```

class AuthScreenController extends StatelessWidget {
  const AuthScreenController({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return BlocProvider(
      create: (context) => AuthBloc(LoadingState())..add(SessionCheckerEvent()),
      child: BlocBuilder<AuthBloc, AuthState>(
        builder: (context, state) {
          if (state is LoggedInState) {
            return const HomeScreen();
          } else if (state is MessageState) {
            manualToastMsg(state.message);
          } else if (state is LoggedOutState) {
            return const GetStartedScreen();
          }
          return showLoading(context);
        },
      ),
    );
  }
}

```

Dart functions to communicate with backend API's

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:sentimento/bloc/auth_bloc/auth_bloc.dart';
import 'package:sentimento/screens/getstarted_screen/getstarted_screen.dart';
import 'package:sentimento/screens/home_screen/home_screen.dart';
import 'package:sentimento/widgets/minor_ui_parts.dart';

class AuthScreenController extends StatelessWidget {
  const AuthScreenController({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return BlocProvider(
      create: (context) => AuthBloc(LoadingState())..add(SessionCheckerEvent()),
      child: BlocBuilder<AuthBloc, AuthState>(
        builder: (context, state) {
          if (state is LoggedInState) {
            return const HomeScreen();
          }
        },
      ),
    );
  }
}

```

```

        } else if (state is MessageState) {
            manualToastMsg(state.message);
        } else if (state is LoggedOutState) {
            return const GetStartedScreen();
        }
        return showLoading(context);
    },
),
);
}
}

```

Dart file that controls the navigation when application is opened.

```

class TextFieldValidator {
    static emailValidator(email) {
        bool emailValid = RegExp(
            r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+")
            .hasMatch(email);
        if (emailValid) {
            return null;
        } else {
            return "Enter valid email address";
        }
    }

    static passwordValidator(password) {
        if (password!.isEmpty || password.length < 6) {
            return "Password length must be atleast 6";
        }

        return null;
    }

    static nameValidator(name) {
        if (name!.isEmpty) {
            return "Name cannot be empty";
        }
        if (name.length < 5) {
            return "Name should be more than 5chars";
        }
        if (name.length > 55) {
            return "Name should be less than 55chars";
        }
    }
}

```

```

    }

}

static keyValidator(key) {
  if (key!.isEmpty) {
    return "Key/coupan cannot be empty";
  }
  if (key.length < 5) {
    return "Key/coupan should be more than 5chars";
  }
  if (key.length > 250) {
    return "Key/coupan should be less than 250chars";
  }
}

static dataSourceValidator(source, platform) {
  if (platform == "youtube") {
    if (source!.isEmpty) {
      return "Video url cannot be empty";
    }
    if (source.length < 5) {
      return "Url cannot be this small";
    }
    if (source.length > 250) {
      return "Url cannot be longer than 250chars";
    }
  }
  if (platform == "twitter") {
    if (source!.isEmpty) {
      return "Tweet topic cannot be empty";
    }
    if (source.length < 2) {
      return "Tweet topic cannot be this small";
    }
    if (source.length > 250) {
      return "Tweet topic be longer than 250chars";
    }
  }
}

static numOfDataValidator(dataCount) {
  try {
    int approx = int.parse(dataCount);
    if ((approx < 20) || (approx > 1000)) {
      return "Range of 20 to 1000 is only allowed for now.";
    }
  }
}

```

```

        }
    } catch (e) {
        return "Enter numeric values only";
    }
}

static roleValidator(role) {
    if ((role.length < 3) || (role.length > 50)) {
        return "Length of role should be between 3 to 50";
    }
}

static phoneNumberValidator(number) {
    if ((number.length < 5) || (number.length > 20)) {
        return "Length of phone number should be between 5 to 20";
    }
}

static descriptionValidator(desc) {
    if (desc.length < 15) {
        return "Description should be atleast 15chars in length";
    }
}

static workHrValidator(hr) {
    try {
        int hrs = int.parse(hr);
        if ((hrs < 1) || (hrs > 8)) {
            return "Working hour should be in range of 1 to 8";
        }
    } catch (e) {
        return "Enter numeric values only";
    }
}

static wageValidator(wage) {
    if ((wage.length < 2) || (wage.length > 15)) {
        return "Invalid wage/hour format";
    }
}
}

```

Functions to validate text fields

```
import 'dart:convert';
import 'api_universal.dart';

const String loginEndPoint = "$domain/login/";

const String registerEndPoint = "$domain/register/";

Future requestProfile() async {
    return blocResponder(await getGETResponse("$domain/profile/"));
}

Future requestLogin(String email, String password) async {
    var requestBody =
        jsonEncode(<String, String>{"email": email, "password": password});
    return blocResponder(
        await getResponse(loginEndPoint, requestHeader, requestBody));
}

Future requestRegister(name, email, password, purpose) async {
    var requestBody = jsonEncode(<String, String>{
        "name": name,
        "purpose": purpose,
        "email": email,
        "password": password
    });
    return blocResponder(
        await getResponse(registerEndPoint, requestHeader, requestBody));
}
```

Asynchronous functions to perform login, register http request

```

import 'dart:convert';
import 'package:sentimento/models/post_model.dart';
import 'package:sentimento/widgets/minor_ui_parts.dart';

import 'api_universal.dart';

const String vacancyEndPoint = "$domain/vacancy/";
const String deletePostEndPoint = "$domain/vacancy/?id=";
const String vacanciesEndPoint = "$domain/vacancy/all/?filter=vacancy";
const String freelancersEndPoint = "$domain/vacancy/all/?filter=freelancer";

Future requestAllPosts(filter) async {
    if (filter == "vacancy") {
        var response = blocResponder(await getGETResponse(vacanciesEndPoint));
        return PostModel.fromJson(response);
    } else {
        var response = blocResponder(await getGETResponse(freelancersEndPoint));
        return PostModel.fromJson(response);
    }
}

Future requestUserPosts() async {
    var response = blocResponder(await getGETResponse(vacancyEndPoint));
    return PostModel.fromJson(response);
}

Future addNewPost(Vacancy post) async {
    var requestBody = jsonEncode(<String, dynamic>{
        "title": post.title,
        "contact_email": post.contactEmail,
        "contact_number": post.contactNumber,
        "description": post.description,
        "work_hr": post.workHr,
        "wage_hr": post.wageHr,
        "is_negotiable": post.isNegotiable,
        "is_vacancy": post.isVacancy
    });
    var response =
        blocResponder(await getPostResponse(vacancyEndPoint, requestBody));
    manualToastMsg(response["msg"]);

    return response;
}

Future deleteSelectedPost(postId) async {

```

```

var deleteResponse =
    blocResponder(await getDELETEResponse("$deletePostEndPoint$postId"));
manualToastMsg(deleteResponse["msg"]);

return deleteResponse;
}

```

Asynchronous functions to perform vacancy related http requests

```

import 'dart:async';
import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';
import 'package:sentimento/api/auth_requests.dart';
import 'package:sentimento/utilities/securestorage.dart';
part 'auth_event.dart';
part 'auth_state.dart';

class AuthBloc extends Bloc<AuthEvent, AuthState> {
    AuthBloc(LoadingState LoadingState) : super(LoadingState()) {
        on<SessionCheckerEvent>(_sessionChecker);
        on<LoginClickedEvent>(_loginClick);
        on<RegisterClickedEvent>(_registerClick);
        on<InitialEvent>(_initialEvent);
    }

    FutureOr<void> _initialEvent(
        InitialEvent initialEvent, Emitter<AuthState> emit) {
        emit.call(LoggedOutState());
    }

    FutureOr<void> _sessionChecker(
        SessionCheckerEvent sessionClickedEvent, Emitter<AuthState> emit) async {
        emit.call(LoadingState());
        //checks if bearer token is saved or not
        bool userLoggedIn = await isLoggedIn();
        if (userLoggedIn) {
            emit.call(LoggedInState());
        } else if (!userLoggedIn) {
            emit.call(LoggedOutState());
        }
    }

    FutureOr<void> _loginClick(
        LoginClickedEvent loginClickedEvent, Emitter<AuthState> emit) async {

```

```

emit.call(LoadingState());
try {
  var loginResponse = await requestLogin(
    LoginClickedEvent.email, LoginClickedEvent.password);
  authValidation(loginResponse, emit);
} catch (e) {
  emit.call(MessageState(message: "No internet or try again"));
}
}

FutureOr<void> _registerClick(RegisterClickedEvent registerClickedEvent,
  Emitter<AuthState> emit) async {
  emit.call(LoadingState());
  try {
    var registerResponse = await requestRegister(
      registerClickedEvent.fullName,
      registerClickedEvent.email,
      registerClickedEvent.password,
      registerClickedEvent.purpose);
    authValidation(registerResponse, emit);
  } catch (e) {
    emit.call(MessageState(message: "No internet or try again"));
  }
}

void authValidation(backendResponse, Emitter<AuthState> emit) async {
  if (backendResponse["success"] == "true") {
    //when login request is valid, will save token and take user to home screen
    saveUserInfo(backendResponse["access_token"], backendResponse["name"],
      backendResponse["email"]);
    emit.call(MessageState(message: backendResponse["msg"]));
    //await Future.delayed(const Duration(milliseconds: 500));
    emit.call(LoggedInState());
  } else {
    emit.call(MessageState(message: backendResponse["msg"]));
    // emit.call(LoggedOutState());
  }
}
}

```

User authentication related state management

```
class AppRoutes {  
    static String getstartedscreen = "/getstarted";  
    static String loginscreen = "/login";  
    static String registerscreen = "/register";  
    static String splashscreen = "/splash";  
    static String homescreen = "/home";  
    static String profilescreen = "/profile";  
    static String commentsAnalysis = "/comments";  
    static String tweetsAnalysis = "/tweets";  
    static String manualAnalysis = "/manual";  
    static String analysisResult = "/result";  
    static String reports = "/reports";  
    static String postVacancy = "/post/vacancy";  
    static String postSkill = "/post/skill";  
    static String viewVacancy = "/view/vacancies";  
    static String viewFreelancer = "/view/freelancers";  
    static String myPosts = "/myposts";  
}
```

Maintained static routes to access easily