Urdu Back Translation Text Reuse Detection

Final Year Project

Session 2021-2025

A 4th Year Student

A project submitted in partial fulfilment of the
COMSATS University Degree
of
BS in Software Engineering (CUI)

Department of Computer Science

COMSATS University Islamabad, Lahore Campus

1. Project Detail

Type (Nature of project)	[ ] **Development** [✓] **Research & Development**			
Area of specialization	Machine Learning, Natural Language Processing			
Project Group Members				
Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)	FA21-BSE-114	Muhammad Umer Aamir	FA21-BSE-114@cuilahore.edu.pk	Umer Aamir
(ii)	FA21-BSE-120	Malik Ashas	FA21-BSE-120@cuilahore.edu.pk	Ashas
(iii)	FA21-BSE-053	Usama Tufail	FA21-BSE-053@cuilahore.edu.pk	Usama

*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others.

Plagiarism Free Certificate

This is to certify that, I am Muhammad Umer Aamir S/o Chaudhary Aamir Riaz group leader of FYP under registration no CIIT/FA21-BSE-114/LHR at Computer Science Department, COMSATS University Islamabad, Lahore Campus. I declare that my FYP report is checked by my supervisor and the similarity index is 7% that is less than 20%, an acceptable limit by HEC. Report is attached herewith as Appendix A.

Date: Jun 12, 2025 Name of Group Leader: Muhammad Umer Aamir Signature: Umer Aamir

Name of Supervisor: Dr. Muhammad Sharjeel
Designation: Assistant Professor
Signature: _____

HoD: _____

Signature: _____

Abstract

It is now very easy in today's world to plagiarize or copy text as well as rephrase it using free information on the web because of huge free information base available on World wide web in different formats. The risk of content misuse without appropriate credit or acknowledgement to owner also arises. There are some tools available for some languages, Urdu has been at the low end due to its being considered a low-resource language. Consequently, the detection of reused texts and plagiarism in Urdu has become difficult for writers and rundown them of the appropriate credit.

This project addresses the problem by developing resources and techniques specifically for detecting Urdu text reuse and plagiarism. Our primary goal is to create Urdu text reuse corpora using a semi-automated back-translation approach, filling an important gap in current Natural language resources. Furthermore, we will design and test a variety of machine learning and deep learning algorithms on these newly created corpora to assess their effectiveness in recognizing Urdu text reuse.

Lastly, our work will integrate these detection methods into web applications for practical, user friendly access. We believe this project will be a valuable contribution to the Urdu Natural Language Processing community and will provide real-world benefits in academic and professional environments where original Urdu content and proper attribution for writers are essential.

Table of Contents

1 Introduction 14

1.1 Introduction 15

1.2 Objectives 15

1.2.1 Creation of Text Reuse Corpora 15
1.2.2 Apply Machine Learning, Deep Learning Techniques 16
1.2.3 Develop Web-Based Application 16
1.2.4 Contribution in NLP Resources 16

1.3 Problem statement 16

1.3.1 Digitally low resource language 16
1.3.2 Linguistically complex language 16
1.3.3 Weakness of Conventional methods 16
1.3.4 Limited Research on Urdu text reuse detection and plagiarism detection 16

1.4 Assumptions & Constraints 16

1.4.1 Dataset 16
1.4.2 Multi language translation tool 17
1.4.3 Dataset limitation 17
1.4.4 Computational Cost 17
1.4.5 Translation Errors 17

1.5 Project Scope 17

1.5.1 Creation of Text Reuse Corpora 17
1.5.2 Experimentation using ML, DL Techniques 17
1.5.3 Develop Web Application 17
1.5.4 Development of REST API 17
1.5.4 Integration deep learning model into web-based application 18
1.5.5 Multi-lingual tool 18
1.5.6 Use of unsupervised techniques 18

2 Requirements Analysis 19

2.1 Literature review 20

2.1.1 Counter Corpus 20
2.1.2 Novel DNN Approaches for Urdu Paraphrase Detection 20
2.1.3 Urdu Sentential Paraphrases (USP) Corpus and Paraphrase Detection Techniques 21
2.1.4 Urdu Short Text Reuse Detection and Corpus Development 21
2.1.5 Architecture Comparison 22

2.2 Stakeholders list 22

2.2.1 System Admin 22
2.2.2 End-Users 22

2.3 Requirements elicitation 23

2.3.1 FR-01 Login 23
2.3.2 FR-02 Create Account 23
2.3.3 FR-03 Forgot Password 24
2.3.4 FR-04 Change Password 25
2.3.5 FR-05 Profile Management 26
2.3.6 FR-06 API Services 26

2.3.7 FR-07 Rate Limiting and Throttling 27
2.3.8 FR-08 Authorization for API 27
2.3.9 FR-09 API Logging 27
2.3.10 FR-10 User History 27
2.3.11 FR-11 API Key 28
2.3.12 FR-12 API Monitoring 28
2.3.13 FR-13 Text Reuse Detection 28
2.3.14 FR-14 Plagiarism Detection 29
2.3.15 FR-15 Contact 30
2.3.16 FR-16 Notifications 31
2.3.17 FR-17 Admin Dashboard 31
2.3.18 FR-18 System Management 31
2.3.19 FR-19 Report Generation 32
2.3.20 FR-20 Analytics 32
2.3.21 NFR-01 Performance 32
2.3.22 NFR-02 Scalability 33
2.3.23 NFR-03 Usability 33
2.3.24 NFR-04 Availability 33
2.3.25 NFR-05 Security 34
2.3.26 NFR-06 Maintenance 34
2.3.27 NFR-07 Robustness 35
2.3.28 NFR-08 Backup and Recovery 35
2.3.29 NFR-09 Monitoring & logging 36
2.3.30 NFR-10 Interoperability 36
2.4 Requirements traceability matric (RTM) 36
2.5 Use case descriptions 39
2.5.1 Login 39
2.5.2 Create Account 40
2.5.3 Forgot Password 43
2.5.4 Change Password 45
2.5.5 Profile Management 47
2.5.6 API Services 48
2.5.7 Rate Limiting for API 50
2.5.8 Authorization for API 51
2.5.9 API Logging 53
2.5.10 User History 54
2.5.11 API Key Management 55
2.5.12 API Monitoring 57
2.5.13 Reuse Detection 59
2.5.14 Plagiarism Detection 60
2.5.15 Contact From 63
2.5.16 Notifications 64
2.5.17 Admin Dashboard 66
2.5.18 System Management 68
2.5.19 Report Generation 70
2.5.20 Analytics 71
2.6 Use case design 73
2.6.1 Login 73
2.6.2 Create Account 73

2.6.3 Forget Password....74
2.6.4 Change Password....74
2.6.5 Update Profile....75
2.6.6 API Services....75
2.6.7 Rate Limiting for API....76
2.6.8 Authorization for API....76
2.6.9 API logging....77
2.6.10 User history....77
2.6.11 API key management....78
2.6.12 API Monitoring....78
2.6.13 Text Reuse Detection....79
2.6.14 Plagiarism Detection....80
2.6.15 Contact....81
2.6.16 Notifications....82
2.6.17 Admin Dashboard....82
2.6.18 System Management....83
2.6.19 Report Generation....83
2.6.20 Analytics....84
2.7 Software development life cycle model....84
3 System Design....85
3.1 Work breakdown structure (WBS)....86
3.2 Activity diagram....87
3.2.1 Login....87
3.2.2 Create Account....87
3.2.3 Forgot Password....88
3.2.4 Change Password....89
3.2.5 Edit Profile....90
3.2.6 API services....91
3.2.7 Rate Limiting for API....92
3.2.8 Authorization for API....92
3.2.9 API logging....93
3.2.10 User History....94
3.2.11 API Key management....95
3.2.12 API monitoring....96
3.2.13 Text Reuse Detection....97
3.2.14 Plagiarism Detection....98
3.2.15 Contact....99
3.2.16 Notification....100
3.2.17 Admin Dashboard....101
3.2.18 System Management....102
3.2.19 Report Generation....103
3.2.20 Analytics....104
3.3 Sequence diagram....105
3.3.1 Login....105
3.3.2 Create Account....106
3.3.3 Forgot Password....107
3.3.4 Change Password....108
2.1.1. Edit Profile....109

3.3.5 API services 110
3.3.6 Rate Limiting for API 111
3.3.7 Authorization for API 112
3.3.8 API logging 113
3.3.9 User History 114
3.3.10 API Key management 115
3.3.11 API monitoring 116
3.3.12 Text Reuse Detection 117
2.1.2. Plagiarism Detection 118
3.3.13 Contact 119
3.3.14 Notification 120
3.3.15 Admin Dashboard 121
3.3.16 System Management 122
3.3.17 Report Generation 123
3.3.18 Analytics 123
3.4 Software architecture 124
3.5 Model 125
3.6 Class diagram 125
3.7 Database diagram 126
3.8 Network diagram (Gantt chart) 127
3.9 Collaboration Diagram 127
3.9.1 Login 127
3.9.2 Create Account 128
3.9.3 Forgot Password 128
3.9.4 Change Password 129
3.9.5 Edit Profile 129
3.9.6 API services 130
3.9.7 Rate Limiting for API 130
3.9.8 Authorization for API 131
3.9.9 API logging 131
3.9.10 User History 132
3.9.11 API Key management 132
3.9.12 API monitoring 133
3.9.13 Text Reuse Detection 133
3.9.14 Plagiarism Detection 134
3.9.15 Contact 134
3.9.16 Notification 135
3.9.17 Admin Dashboard 135
3.9.18 System Management 136
3.9.19 Report Generation 136
3.9.20 Analytics 137
4 System Testing 138
4.1 Test cases 139
4.1.1 Login 139
4.1.2 Create Account 140

4.1.3 Forget Password 140
4.1.4 Change Password 142
4.1.5 Edit Profile 142
4.1.6 API Services 143
4.1.7 Rate Limit for API 144
4.1.8 Authorization for API 145
4.1.9 API logging 146
4.1.10 User History 147
4.1.11 API Key Management 148
4.1.12 API Monitoring 149
4.1.13 Text Reuse Detection 150
4.1.14 Plagiarism Detection 151
4.1.15 Contact & Query Form 152
4.1.16 Notifications 153
4.1.17 Admin Dashboard 154
4.1.18 System Management 155
4.1.19 Report Generation 156
4.1.20 Analytics Report 157
4.2 Testing 158
4.2.1 Unit Testing 158
4.2.2 Black Box Testing 158
4.2.3 White Box Testing 158
4.2.4 Integration Testing 158
4.2.5 Frontend and Backend 159
4.2.6 Backend and AI Model Server 159
4.2.7 Backend and Database Server 159
4.2.8 Acceptance testing 159
4.2.9 Function Testing 159
4.2.10 Performance Testing 159
4.2.11 Security Testing 159
4.2.12 Usability Testing 159
4.2.13 Compatibility Testing 159
5 Conclusion 160
5.1 Problems Faced & Lessons Learned 161
5.1.1 Dataset Challenges 161
5.1.2 Language Complexity 161
5.1.3 Computational Resources 161
5.1.4 Limited literature 161
5.1.5 Lessons Learned 161
5.1.6 Machine Learning techniques 161
5.1.7 Deep Learning (DL) 161
5.1.8 Transformers and Large Language Models (LLMs) 161
5.2 Project Summary 162
5.3 Future Work 162
5.3.1 Dataset Expansion 162
5.3.2 Cross-Language Detection 162
5.3.3 Model Optimization 162
5.3.4 Mobile Application Development 162

6 Implementation 163
6.1 Home Page: 164
6.2 Contact Us: 165
6.3 About: 166
6.4 Login: 167
6.5 Text Reuse Detection: 168
6.6 API Service: 168
6.7 Plagiarism Detection: 169
6.8 User Management: 169
6.9 Admin Dashboard: 170
7 References 171

List of Tables

Table 1 Comparison of Architectures ..... **Error! Bookmark not defined.**
Table 2 FR-01 Login ..... **Error! Bookmark not defined.**
Table 3 FR-02 Create Account ..... **Error! Bookmark not defined.**
Table 4 FR-03 Forgot Password ..... **Error! Bookmark not defined.**
Table 5 FR-04 Change Password ..... **Error! Bookmark not defined.**
Table 6 FR-05 Profile Management ..... **Error! Bookmark not defined.**
Table 7 FR-06 API Services ..... **Error! Bookmark not defined.**
Table 8 FR-07 Rate Limiting and Throttling ..... **Error! Bookmark not defined.**
Table 9 FR-08 Authorization for API ..... **Error! Bookmark not defined.**
Table 10 FR-09 API Logging ..... **Error! Bookmark not defined.**
Table 11 FR-10 User History ..... **Error! Bookmark not defined.**
Table 12 FR-11 API Key ..... **Error! Bookmark not defined.**
Table 13 FR-12 API Monitoring ..... **Error! Bookmark not defined.**
Table 14 FR-13 Text Reuse Detection ..... **Error! Bookmark not defined.**
Table 15 FR-14 Plagiarism Detection ..... **Error! Bookmark not defined.**
Table 16 FR-15 Contact ..... **Error! Bookmark not defined.**
Table 17 FR-16 Notifications ..... **Error! Bookmark not defined.**
Table 18 FR-17 Admin Dashboard ..... **Error! Bookmark not defined.**
Table 19 FR-18 System Management ..... **Error! Bookmark not defined.**
Table 20 FR-19 Report Generation ..... **Error! Bookmark not defined.**
Table 21 FR-20 Analytics ..... **Error! Bookmark not defined.**
Table 22 NFR-01 Performance ..... **Error! Bookmark not defined.**
Table 23 NFR-02 Scalability ..... **Error! Bookmark not defined.**
Table 24 NFR-03 Usability ..... **Error! Bookmark not defined.**
Table 25 NFR-04 Availability ..... **Error! Bookmark not defined.**
Table 26 NFR-05 Security ..... **Error! Bookmark not defined.**
Table 27 NFR-06 Maintenance ..... **Error! Bookmark not defined.**
Table 28 NFR-07 Robustness ..... **Error! Bookmark not defined.**
Table 29 NFR-08 Backup and Recovery ..... **Error! Bookmark not defined.**
Table 30 NFR-09 Monitoring & logging ..... **Error! Bookmark not defined.**
Table 31 NFR-10 Interoperability ..... **Error! Bookmark not defined.**
Table 32 Requirement Traceability Matrix ..... **Error! Bookmark not defined.**
Table 33 UC-01 Login ..... **Error! Bookmark not defined.**
Table 34 UC-02 Create Account ..... **Error! Bookmark not defined.**
Table 35 UC-03 Forgot Password ..... **Error! Bookmark not defined.**
Table 36 UC-04 Change Password ..... **Error! Bookmark not defined.**
Table 37 UC-05 Profile Management ..... **Error! Bookmark not defined.**
Table 38 UC-06 API Services ..... **Error! Bookmark not defined.**
Table 39 UC-07 Rate Limiting for API ..... **Error! Bookmark not defined.**
Table 40 UC-08 Authorization for API ..... **Error! Bookmark not defined.**
Table 41 UC-09 API Logging ..... **Error! Bookmark not defined.**
Table 42 UC-10 User History ..... **Error! Bookmark not defined.**
Table 43 UC-11 API Key Management ..... **Error! Bookmark not defined.**
Table 44 UC-12 API Monitoring ..... **Error! Bookmark not defined.**
Table 45 UC-13 Reuse Detection ..... **Error! Bookmark not defined.**
Table 46 UC-14 Plagiarism Detection ..... **Error! Bookmark not defined.**
Table 47 UC-15 Contact From ..... **Error! Bookmark not defined.**
Table 48 UC-16 Notifications ..... **Error! Bookmark not defined.**
Table 49 UC-17 Admin Dashboard ..... **Error! Bookmark not defined.**
Table 50 UC-18 System Management ..... **Error! Bookmark not defined.**

Table 51 UC-19 Report Generation ..... **Error! Bookmark not defined.**
Table 52 UC-20 Analytics ..... **Error! Bookmark not defined.**
Table 53 Test Case 01 Login ..... **Error! Bookmark not defined.**
Table 54 Test Case 02 Create Account ..... **Error! Bookmark not defined.**
Table 55 Test Case 03 Forget Password ..... **Error! Bookmark not defined.**
Table 56 Test Case 04 Change Password ..... **Error! Bookmark not defined.**
Table 57 Test Case 05 Edit Profile ..... **Error! Bookmark not defined.**
Table 58 Test Case 06 API Services ..... **Error! Bookmark not defined.**
Table 59 Test Case 07 Rate Limiting ..... **Error! Bookmark not defined.**
Table 60 Test Case 08 Authorization ..... **Error! Bookmark not defined.**
Table 61 Test Case 09 API logging ..... **Error! Bookmark not defined.**
Table 62 Test Case 10 User History ..... **Error! Bookmark not defined.**
Table 63: Test Case 11 API Key Management ..... **Error! Bookmark not defined.**
Table 64 Test Case 12 API Monitoring ..... **Error! Bookmark not defined.**
Table 65 Test Case 13 Text Reuse Detection ..... **Error! Bookmark not defined.**
Table 66 Test Case 14 Plagiarism Detection ..... **Error! Bookmark not defined.**
Table 67 Test Case 15 Contact & Query Form ..... **Error! Bookmark not defined.**
Table 68 Test Case 16 Notifications ..... **Error! Bookmark not defined.**
Table 69 Test Case 17 Admin Dashboard ..... **Error! Bookmark not defined.**
Table 70 Test Case 18 System Management ..... **Error! Bookmark not defined.**
Table 71 Test Case 19 Report Generation ..... **Error! Bookmark not defined.**
Table 72 Test Case 20 Analytics Report ..... **Error! Bookmark not defined.**

List of Figures

Figure 1 Hybrid Architecture using DNN Approach 19
Figure 2 USP Architecture 20
Figure 3 Short Text Reuse Detection Architecture 20
Figure 4 Login Use Case Diagram 66
Figure 5 Create Account Use Case 67
Figure 6 Forget Password Use Case Diagram 67
Figure 7 Change Password Use Case Diagram 68
Figure 8 Update Profile Use Case Diagram 68
Figure 9 API Services Use Case Diagram 69
Figure 10 Rate Limiting for API Use Case Diagram 69
Figure 11 API Authorization Use Case Diagram 70
Figure 12 API logging Use Case Diagram 70
Figure 13 User History Use case Diagram 71
Figure 14 API key management Use case Diagram 71
Figure 15 API Monitoring Use Case Diagram 72
Figure 16 Text Reuse Detection Use Case Diagram 73
Figure 17 Plagiarism Detection Use case Diagram 74
Figure 18 Contact Use Case Diagram 75
Figure 19 Notifications Use Case Diagram 76
Figure 20 Admin Dashboard Use Case Diagram.....76
Figure 21 System Management Use Case Diagram 77
Figure 22 Report Generation Use Case Diagram 77
Figure 23 Analytics Use Case Diagram.....78
Figure 24 Agile Diagram 78
Figure 25 Work Break Down Structure 80
Figure 26 Login Activity Diagram 81
Figure 27 Create Account Activity Diagram 81
Figure 28 Forgot Password Activity Diagram 82
Figure 29 Change Password Activity Diagram 83
Figure 30 Edit Profile Activity Diagram 84
Figure 31 API Services Activity Diagram 85
Figure 32 API Rate Limiting Activity Diagram 86
Figure 33 API Authorization Activity Diagram 86
Figure 34 API logging Activity Diagram 87
Figure 35 User history Activity Diagram 88
Figure 36 API key Management Activity Diagram 89
Figure 37 API monitoring Activity Diagram 90
Figure 38 Text Reuse Detection Activity Diagram 91
Figure 39 Plagiarism Detection Activity Diagram 92
Figure 40 Contact Activity Diagram 93
Figure 41 Notifications Activity Diagram 94
Figure 42 Admin Dashboard Activity Diagram 95
Figure 43 System Management Activity Diagram 96
Figure 44 Report Generation Activity Diagram 97
Figure 45 Analytics Activity Diagram 98
Figure 46 Login Sequence Diagram 99
Figure 47 Create Account Sequence Diagram 100
Figure 48 Forgot Password Sequence Diagram 101
Figure 49 Change Password Sequence Diagram 102

Figure 50 Update Profile Sequence Diagram 103
Figure 51 API Services Sequence Diagram 104
Figure 52 Rate Limiting Sequence Diagram .....105 Figure 53 Authorization Sequence Diagram .....106 Figure 54 API logging Sequence Diagram 107
Figure 55 User History Sequence Diagram 108
Figure 56 API Key Management Sequence Diagram 109
Figure 57 API Monitoring Sequence Diagram 110
Figure 58 Text Reuse Detection Sequence Diagram 111
Figure 59 Plagiarism Detection Sequence Diagram 112
Figure 60 Contact Sequence Diagram 113
Figure 61 Notification Sequence Diagram 114
Figure 62 Admin Dashboard Sequence Diagram 115
Figure 63 System Management Sequence Diagram 116
Figure 64 Report Generation Sequence Diagram 117
Figure 65 Analytics Sequence Diagram 117
Figure 66 Software Architecture Diagram 118
Figure 67 Machine Learning Model Process 119
Figure 68 Class Diagram 119
Figure 69 Database Diagram 120
Figure 70 Network diagram (Gantt chart) 121
Figure 71 Login Collaboration Diagram 121
Figure 72 Create Account Collaboration Diagram 122
Figure 73 Forgot Password Collaboration Diagram 122
Figure 74 Change Password Collaboration Diagram 123
Figure 75 Edit Profile Collaboration Diagram 123
Figure 76 API services Collaboration Diagram 124
Figure 77 Rate Limiting for API Collaboration Diagram 124
Figure 78 Authorization for API Collaboration Diagram 125
Figure 79 API logging Collaboration Diagram 125
Figure 80 User History Collaboration Diagram 126
Figure 81 API Key management Collaboration Diagram 126
Figure 82 API monitoring Collaboration Diagram 127
Figure 83 Text Reuse Detection Collaboration Diagram 127
Figure 84 Plagiarism Detection Collaboration Diagram 128
Figure 85 Contact Collaboration Diagram 128
Figure 86 Notification Collaboration Diagram 129
Figure 87 Admin Dashboard Collaboration Diagram 129
Figure 88 System Management Collaboration Diagram 130
Figure 89 Report Generation Collaboration Diagram 130
Figure 90 Analytic Collaboration Diagram 131
Figure 91 Home Page 158
Figure 92 Contact us page 159
Figure 93 About Page 160
Figure 94 Login Page 161
Figure 95 Text Reuse Detection 162
Figure 96 API Service 162
Figure 97 Plagiarism Detection 163
Figure 98 User Management 163

Figure 99 Admin Dashboard 164
Figure 100 - Report 166
Figure 101- AI Report 166

Chapter # 01

Introduction

1 Introduction

1.1 Introduction

Nowadays, it is very easy to reuse, rephase or copy text of others due to the availability of large amount of content on internet. Text reuse detection plays a very important role in detecting whether two texts or documents giving the same interpretation using different words, phrases, grammatical and syntactical structures. This task is important in many fields, like Academia, content writing and professional documentation. Availability of the large amount of content on the internet, the need for effective text reuse detection tools has become very significant. For high resource languages like English, Spanish, and French, many advanced text reuses and plagiarism detection tools already exist. But for Urdu language there is not a single tool available on internet due to digitally low resource language. Urdu is a linguistically rich language with unique syntactical structures [1] and vocabulary making it challenging to identify the reuse of text. The traditional approaches to text reuse detection and plagiarism detection are based on lexical [2] and syntactical similarities which cannot handle the contextual variations in text or documents due to linguistically rich language. Recent developments in Natural Language Processing and Machine Learning can address these challenges.

In this FYP Project, we aim to develop the nine text reuse detection datasets for the Urdu language using semi-automated approach. The process includes translation of Urdu text into several different languages and then back translation to Urdu language. Five corpora will be created by translating Urdu text into widely spoken languages, while the remaining four corpora will be created by translating Urdu text into different language families. The back translation technique includes translating a source language A into a target language B and then translating it back into the source language A. By using this method, we can capture diverse contextual and semantic meanings, which are essential for training models. We will apply Machine Learning, Deep Learning and LLM-based techniques to evaluate the strength of multiple Urdu text reuse corpora that shall be created during this project. Additionally, we will create a web application which will detect text reuse and plagiarism of documents.

1.2 Objectives

The following are the Aims and Objectives of our final year project:

1.2.1 Creation of Text Reuse Corpora

We will create of 9 Urdu text reuse corpora. We will construct the corpora using back translation technique. The 5 corpora will be created by applying a 1-3-5-7-9 languages translation method. This process involves translating text from one language to other and then back to corpora will be created its original (e.g., Urdu to Arabic, Arabic to English, and then back in Urdu). The 4by the back translating text using 4 most popular language families. These corpora will have source and reused text classified into 3 classes such as Partially Derived (PD), Wholly Derived (WD), and Non-Derived (ND).

1.2.2 Apply Machine Learning, Deep Learning Techniques

We Will apply Machine Learning (e.g., SVM, KNN, Navie Bayes, etc.) and Deep Learning methods (e.g., Transformers, CNN, LSTM, etc.) to predict text reuse in Urdu language documents.

1.2.3 Develop Web-Based Application

We will develop the web application and integrate our best result model into that application for text reuse detection. This helps user to detect Urdu language text reuse and plagiarism easily and effectively.

1.2.4 Contribution in NLP Resources

By creating on the corpora and experiments on it, we can propose the Urdu text reuse detection architecture with effective results. In this way we can contributing to Natural Language Processing NLP Resources.

1.3 Problem statement

The following are the major problems in our final year project:

1.3.1 Digitally low resource language

For some other languages e.g. English, and French, there are multiple large scaled annotated benchmark datasets available for text reuse detection and plagiarism detection. But for Urdu language, it's not like that [3]. One the significant problem in Urdu language is the absence of large scale publicly available benchmark dataset for the task of Urdu text reuse detection and plagiarism detection. There is not even a single large scaled annotated benchmark dataset for our task available publicly.

1.3.2 Linguistically complex language

Urdu is a linguistically rich language with unique syntactical structures (grammatical arrangement of words) and vocabulary which makes it challenging to detect reuse of text.

1.3.3 Weakness of Conventional methods

The conventional approaches for text reuse detection and plagiarism detection are based on lexical (vocabulary) and syntactical (grammatical arrangement of words) similarities which can't handle the contextual variations in text or documents due to linguistically rich language.

1.3.4 Limited Research on Urdu text reuse detection and plagiarism detection

Due to the unavailability of large-scaled benchmark dataset, linguistically complex language and less community of NLP researchers in Urdu, there is not much literature present on internet for text reuse detection and plagiarism detection. So, there is very limited literature available for the task of Urdu text reuse detection and plagiarism detection.

1.4 Assumptions & Constraints

1.4.1 Dataset

For this Final Year Project, we are using COUNTER corpus which consists of 600 Urdu paraphrased documents pairs and each document is annotated as Partially Derived (PD), Wholly Derived (WD),

or Non-Derived (ND). By using COUNTER corpus, we are creating 9 more corpus using back translation technique.

1.4.2 Multi language translation tool

We are using already build language translation tool for the creation of 9 corpora using back translation technique.

1.4.3 Dataset limitation

As COUNTER corpus consists of only 600 documents pairs. So, we have very limited corpus for the task of Urdu text reuse detection and plagiarism detection.

1.4.4 Computational Cost

Feature extraction like word embeddings/sentence embedding etc. and training deep learning models may require high computational resources.

1.4.5 Translation Errors

Translation of original COUNTER corpus into 9 corpora may introduce the errors to capture the original context of Urdu text.

1.5 Project Scope

1.5.1 Creation of Text Reuse Corpora

We will create of 9 Urdu text reuse corpora. We will construct the corpora using back translation technique. The 5 corpora will be created by applying a 1-3-5-7-9 languages translation method. This process involves translating text from one language to other and then back to its original (e.g., Urdu to Arabic, Arabic to English, and then back in Urdu). The 4 corpora will be created by the back translating text using 4 most popular language families. These corpora will have source and reused text classified into 3 classes such as Partially Derived (PD), Wholly Derived (WD), and Non-Derived (ND).

1.5.2 Experimentation using ML, DL Techniques

Apply Machine Learning (e.g., SVM, KNN, Navie Bayes, etc.) and Deep Learning methods (e.g., Transformers, CNN, LSTM, etc.) to predict text reuse in Urdu documents.

1.5.3 Develop Web Application

We will develop the web application and integrate Machine Learning and Deep Learning model into that application for text reuse detection.

1.5.4 Development of REST API

We will deploy our model and make API of our trained model. So that users/organizations can access our trained model using API.

1.5.4 Integration deep learning model into web-based application

We will integrate the trained model into web-based application. So that users can use trained model using our web-based application.

1.5.5 Multi-lingual tool

Our Machine Learning and Deep Learning model will only train for the task of Urdu text reuse detection and plagiarism detection dataset. So, that's why it's not multi-lingual model. It's only for detection of Urdu documents.

1.5.6 Use of unsupervised techniques

We are using COUNTER corpus which is annotated in 3 classes. Partially Derived (PD), Wholly Derived (WD), and Non-Derived (ND). As classes are already defined so, we are using supervised machine learning approach. That's why unsupervised learning approach is not in our scope.

Chapter # 02

Requirements Analysis

2 Requirements Analysis

2.1 Literature review

2.1.1 Counter Corpus

COUNTER corpus is the Urdu text reuse detection benchmark dataset. It consists of 1200 journalism related documents and is annotated into 3 classes such as Partially Derived (PD), Wholly Derived (WD), and Non-Derived (ND). COUNTER corpus helps the Urdu NLP researchers to develop and evaluate the models which make it an important resource for Natural Language Processing.

2.1.2 Novel DNN Approaches for Urdu Paraphrase Detection

In this study, they address a significant challenge in very digitally low resource language Urdu paraphrase reuse detection task. The authors use a semi-automatic pipeline for generating a paraphrase corpus in Urdu, resulting in a dataset of 3,147 Urdu sentence pairs, with manually tagged 854 paraphrased and 2,293 non-paraphrased pairs. They proposed two deep neural network (DNN)-based approaches that were introduced: Word Embeddings n-gram Overlap (WENGO) and Deep Text Reuse and Paraphrase Plagiarism Detection (D-TRAPPD). They test these methods on Urdu paraphrase detection and text reuse/plagiarism detection task, where D-TRAPPD surpasses WENGO and state-of-the-art techniques in both the approaches. The authors made their Urdu text reuse corpus, deep learning architecture, which is freely available, a valuable resource for Urdu text reuse detection tasks [4]. Their Deep Learning Architecture:

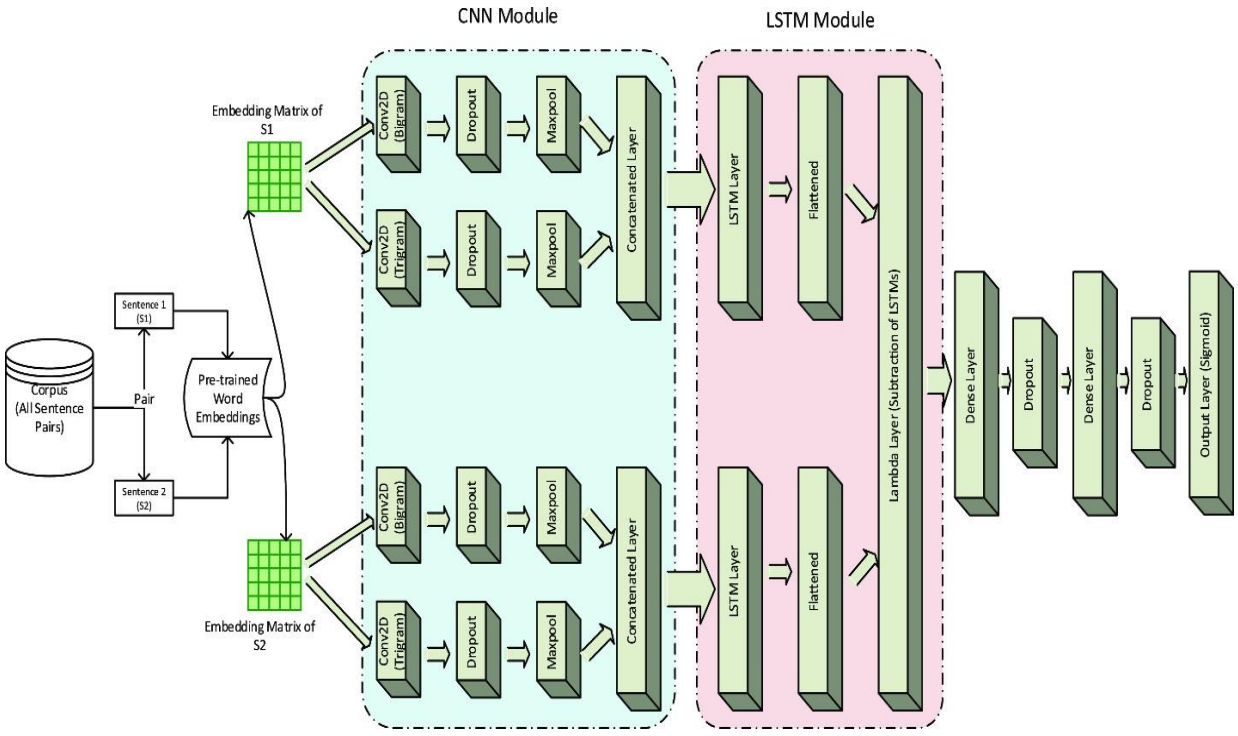


Figure 1 *Hybrid Architecture using DNN Approach*

2.1.3 Urdu Sentential Paraphrases (USP) Corpus and Paraphrase Detection Techniques

Paraphrase detection systems classify the sentences of the text into paraphrased or non-paraphrased on whether they give the same interpretation. Even though immense work has been done in the English language, such kind of significant research gaps in South Asian languages, including Urdu, are there. This work bridges that gap by making the large-scale manually annotated benchmark dataset-USP Corpus with 4,900 sentences of both paraphrased (2,941) and non-paraphrased sentences (1,959), collected from Urdu newspapers. This work also discusses the different techniques for paraphrase detection, such as Word Embedding (WE), Sentence Transformers (ST), and feature-fusion approaches with N-grams as baseline. Their experimental results prove that feature-fusion techniques combining the proposed ST with baseline N-gram features yield the best results at F1 = 0.855 [5]. Their Architecture:

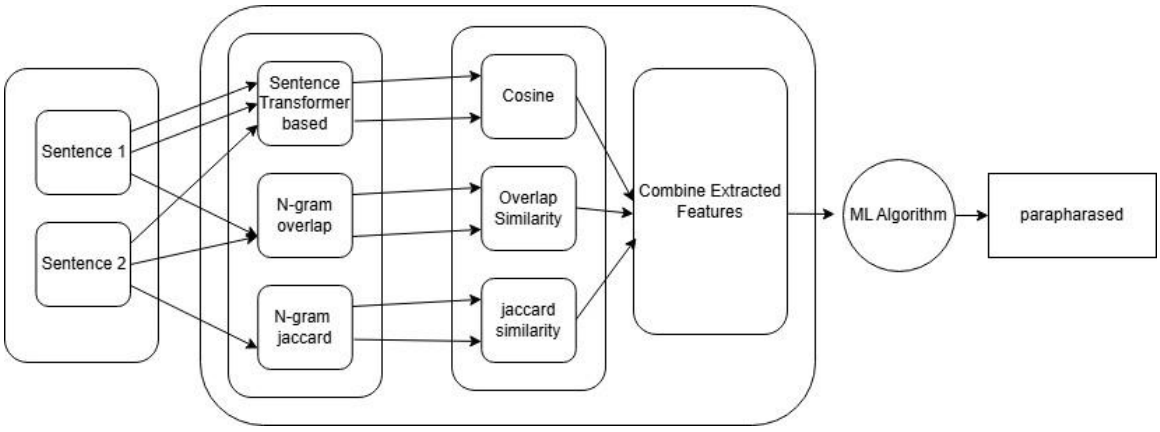


Figure 2 *USP Architecture*

2.1.4 Urdu Short Text Reuse Detection and Corpus Development

Text reuse is the act of taking over text, either word-to-word or word-to-many-words. This article proposes the Urdu Short Text Reuse Corpus (USTRC), which has 2,684 short text pairs labelled manually as verbatim (496), paraphrased (1,329), and independently written (859). The corpus was tested in binary class and multiclass classification with the help of various state-of-the-art algorithms. Results teach that character n-gram overlaps with the J48 classifier do well for Urdu text reuse detection [6].

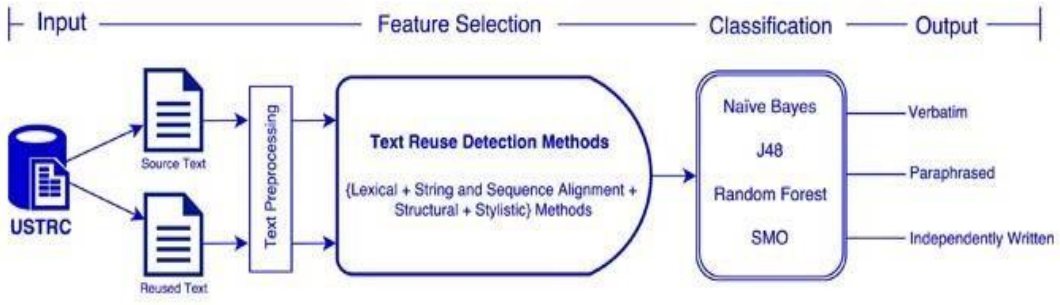


Figure 3 *Short Text Reuse Detection Architecture*

2.1.5 Architecture Comparison

Table 1 *Comparison of Architectures*

Study	Corpus	Categories	Methodology	Key Findings	Architecture
COUNTER Corpus	Urdu Text Reuse Detection	Partially Derived, Wholly Derived, NonDerived	Annotated dataset for text reuse detection	Helps Urdu NLP research, a crucial resource for text reuse tasks	-
Novel DNN Approaches for Urdu Paraphrase Detection	Urdu Paraphrase Corpus (3,147 pairs)	Paraphrased, nonparaphrased	Semi-automatic pipeline, DNN approaches (WENGO, D-TRAPPD)	D-TRAPPD outperforms WENGO and state-of-the-art methods	WENGO , D-TRAPPD
USP Corpus Paraphrase Detection Techniques	Urdu Sentential Paraphrases (4,900 sentences)	Paraphrased, nonparaphrased	Feature-fusion techniques combining ST and N-grams	Best performance with feature-fusion techniques	ST + N-gram features
Urdu Short Text Reuse Detection & Corpus	Urdu Short Text Reuse Corpus (2,684 pairs)	Verbatim, Paraphrased, Independently Written	Character n-gram overlap integrated with the J48 decision tree classifier	Character n-gram overlap with J48 classifier yields good results	J48 classifier, Character n-gram overlap

2.2 Stakeholders list

2.2.1 System Admin

The user who manages and look after the operation, security, and maintenance of the paraphrase detection tool.

2.2.2 End-Users

Individuals who use the application for Urdu text reuse detection, including students, teachers, writers, and organizations.

1. Student/Teacher
2. Writer/Author
3. Organization (which are getting API from us)

2.3 Requirements elicitation

2.3.1 FR-01 Login

Table 2 *FR-01 Login*

ID	Description
FR01-01	The system shall allow authorized users to log into their accounts for access.
FR01-02	The authorized user shall input their email and password on the login page.
FR01-03	The system shall validate the authorized user's email and password against the database records.
FR01-04	The system shall display an error message if login fails; otherwise, it shall log in the user and redirect to the home page.
FR01-05	The system shall log all login attempts, including timestamps and IP addresses.

2.3.2 FR-02 Create Account

Table 3 *FR-02 Create Account*

ID	Description
FR02-01	The system shall allow users to access the account creation page from login interface if user is not logged in.
FR02-02	The system shall provide input fields for the user's name, phone number, email, password, and date of birth.

FR02-03	The system shall validate that the phone number is in a valid format and does not already exist in the system.
FR02-04	The system shall validate that the email address is in a proper format and is not already registered.
FR02-05	The system shall ensure passwords are at least 8 characters long and include uppercase and lowercase letters, digits, and special characters
FR02-06	The system should output clear error messages when input fields are invalid or incomplete.
FR02-07	The system will send a verification email to the user's email address once the account is successfully created.
FR02-08	The system shall log all account creation attempts, including timestamps and IP addresses.

2.3.3 FR-03 Forgot Password

Table 4 *FR-03 Forgot Password*

ID	Description
FR03-01	The system will allow users to initiate a password reset process by choosing the "Forgot Password" option on the login interface.
FR03-02	The system shall prompt users to input their registered email address to request a password reset.
FR03-03	The system will generate and send a One-Time Password (OTP) to the user's email address when requested.
FR03-04	The system shall prompt users to enter the received OTP for verification.

FR03-05	The system shall verify the entered OTP against the one sent to the user's email address.
FR03-06	After successful OTP verification, the system will enable a user to set a new password and confirm it by providing it twice.
FR03-07	The system shall validate that the new password is different from the old password.
FR03-08	The system shall allow users to resend the OTP after a 60-second cooldown period to prevent recalls.

2.3.4 FR-04 Change Password

Table 5 *FR-04 Change Password*

ID	Description
FR04-01	The system shall allow users only to access the change password feature if they are logged onto the system.
FR04-02	The system shall require users to verify their current password before allowing them to set a new password.
FR04-03	The system shall provide input fields for the new password and confirm password during the password change process.
FR04-04	The system shall ensure that the new password is different from the current (old) password.
FR04-05	The system shall require the new passwords to meet the complexity criteria, such as being at least 8 characters long and containing uppercase letters, lowercase letters, digits, and special characters.
FR04-06	The system shall display appropriate error messages if the current password is not correct or if the new password is not according to the complexity requirements.

FR04-07	The system shall remove all the session of user on password change
FR04-08	The system shall log all password change attempts, including timestamps and user identifiers.

2.3.5 FR-05 Profile Management

Table 6 *FR-05 Profile Management*

ID	Description
FR05-01	The system will enable a user to view his/her profile information such as name, email, phone number, and profile picture.
FR05-02	The system shall allow users to update their phone number.
FR05-03	The system shall allow users to change their profile picture by uploading a new image.
FR05-04	The system shall notify users via email upon successful changes to their profile information.
FR05-05	The system shall log all profile management activities, including updates and changes, with timestamps and user identifiers.
FR05-06	The system shall enforce authentication, allowing users to manage their profiles only when they are logged in.

2.3.6 FR-06 API Services

Table 7 *FR-06 API Services*

ID	Description

FR06	The system shall provide API services to other platforms (e.g., blog websites, teachers) to identify text reuse.

2.3.7 FR-07 Rate Limiting and Throttling

Table 8 *FR-07 Rate Limiting and Throttling*

ID	Description
FR07	The system shall implement rate limiting that restricts the number of API calls per user within a specified time frame.

2.3.8 FR-08 Authorization for API

Table 9 *FR-08 Authorization for API*

ID	Description
FR08	The system shall require API users to authenticate using provided API keys.

2.3.9 FR-09 API Logging

Table 10 *FR-09 API Logging*

ID	Description
FR09	The system shall log all API requests, responses, and user activities, ensuring traceability for each request made by the user.

2.3.10 FR-10 User History

Table 11 *FR-10 User History*

ID	Description
FR10	The system shall track and save user history, including API calls and actions performed via the API.

2.3.11 FR-11 API Key

Table 12 *FR-11 API Key*

ID	Description
FR11-01	The system shall allow the users to generate or regenerate their API keys for security purposes.
FR11-02	The system shall allow the users to create multiple API keys for different applications or teams, with shared quotas.
FR11-03	The system shall allow the users to download their API keys in CSV or PDF format for backup and documentation purposes.

2.3.12 FR-12 API Monitoring

Table 13 *FR-12 API Monitoring*

ID	Description
FR12-01	The system shall provide users with detailed usage analytics, including total API calls, remaining quota, accessible through their dashboard, displaying current consumption relative to their package quota.
FR12-02	The system shall automatically suspend API service for users who exceed their quota notifying them by using email.

2.3.13 FR-13 Text Reuse Detection

Table 14 *FR-13 Text Reuse Detection*

ID	Description
FR13-01	The system shall allow users to input two separate Urdu paragraphs for comparison.

FR13-02	The system shall process the two texts to identify reused content using proposed AI Models (e.g., Sentence Transformer).
FR13-03	The system shall calculate the percentage of content reused between the two texts.
FR13-04	The system shall classify the degree of text reuse into predefined categories (e.g., PD, WD, ND).
FR13-05	The system shall allow users to upload images containing text or use a mobile device's camera to capture images, or upload images directly for OCR extraction.
FR13-06	The system shall display the extracted text to the user for review and allow editing before proceeding to text reuse detection.
FR13-07	The system shall allow users to upload two or more documents for text reuse detection.
FR13-08	The system shall generate a comprehensive report detailing the findings of the documentbased text reuse detection, including reuse percentages and sources. allow users to download

2.3.14 FR-14 Plagiarism Detection

Table 15 *FR-14 Plagiarism Detection*

ID	**Description**
FR14-01	The system shall allow users to upload files containing text for plagiarism checking.

FR14-02	The system shall process the entered or extracted text to identify potential plagiarism by comparing it against online sources using search engines and databases.
FR14-03	The system shall calculate the percentage of text that matches external sources to determine the level of plagiarism.
FR14-04	The system shall identify and retrieve the URLs or sources from where the content is potentially copied.
FR14-05	The system shall generate a comprehensive plagiarism report that includes the percentage of matched content and the corresponding source URLs.
FR14-06	The system shall provide an option for users to download the plagiarism report in formats such as PDF

2.3.15 FR-15 Contact

Table 16 *FR-15 Contact*

ID	Description
FR15-01	The system shall allow the users to create and submit queries or complaints via a contact form.
FR15-02	The system shall allow the users to select a message category (e.g., billing, technical support, general) for their query.
FR15-03	The system shall allow the users to view their submitted queries/complaints and the corresponding admin responses.

FR15-04	The system shall allow the users to track the status of their queries (e.g., pending, responded, resolved).
FR15-05	The system shall send email notifications to users whenever the status of their contact form query is updated.

2.3.16 FR-16 Notifications

Table 17 *FR-16 Notifications*

ID	Description
FR16-01	The system shall notify users about important events, actions and task such as exceeding their API Limit, account creation, Completion of task via email.
FR16-02	The system shall allow admins to send notifications to users.

2.3.17 FR-17 Admin Dashboard

Table 18 *FR-17 Admin Dashboard*

ID	Description
FR17	The system shall provide an admin dashboard accessible only to admin users. Users must log in the dashboard, which shows details (e.g., total users, API usage, recent activities, Logs).

2.3.18 FR-18 System Management

Table 19 *FR-18 System Management*

ID	Description

FR18	The system shall allow admins, to view, create, update, system users, complains, queries and other task accessible through admin dashboard.

2.3.19 FR-19 Report Generation

Table 20 *FR-19 Report Generation*

ID	Description
FR19-01	The system shall allow admins to generate various. Reports may be filtered by date range, user, or activity type. Reports shall be exportable in common formats (e.g., CSV, PDF).
FR19-02	The system shall generate detailed reports for API usage, including the most-used endpoints and peak usage times.

2.3.20 FR-20 Analytics

Table 21 *FR-20 Analytics*

ID	Description
FR20-01	The system shall provide users with usage analytics, including token usage, remaining quota, and document history.
FR20-02	The system shall provide admins with insights, user activity, and service usage.

2.3.21 NFR-01 Performance

Table 22 *NFR-01 Performance*

ID	Description
NFR01-01	The system shall respond to user interactions (e.g., login, account creation, API requests) within 10 seconds under normal operating conditions.

NFR01-02	The system must be able to accommodate 25 concurrent users without affecting its performance.
NFR01-03	The system shall ensure that API responses have a maximum latency of 15 seconds to maintain real-time interactions.

2.3.22 NFR-02 Scalability

Table 23 *NFR-02 Scalability*

ID	Description
NFR02-01	The system architecture shall have ability to scale to accommodate increased load and user base growth.
NFR02-02	The system shall automatically allocate and deallocate resources based on real-time demand to optimize performance and cost.

2.3.23 NFR-03 Usability

Table 24 *NFR-03 Usability*

ID	Description
NFR03-01	The system shall provide an perceptive and user-friendly interface that minimize the new learning for new users.

2.3.24 NFR-04 Availability

Table 25 *NFR-04 Availability*

ID	Description

NFR04-01	The system shall maintain an uptime of 99%, ensuring high availability for users.
NFR04-02	The system shall report failure to admin that results in minimize downtime and ensure continuous service availability.

2.3.25 NFR-05 Security

Table 26 *NFR-05 Security*

ID	Description
NFR05-01	The system should implement robust authentication methods and role-based access control to guarantee authorized access.
NFR05-02	All sensitive data, including user passwords and API keys, shall be encrypted.
NFR05-03	The system shall maintain comprehensive logs for all operations (e.g., login attempts, password changes, API usage) with secure storage.
NFR05-04	The system shall be safeguarded against common security vulnerabilities like SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).

2.3.26 NFR-06 Maintenance

Table 27 *NFR-06 Maintenance*

ID	Description
NFR06-01	The system shall follow a modular architecture to facilitate easier updates, maintenance, and scalability.

NFR06-02	Detailed documentation for users and developers shall be available, in the form of API documentation and system architecture diagrams.
NFR06-03	The system's codebase shall follow industry-standard coding practices and be subject to regular code reviews to ensure high quality and maintainability.

2.3.27 NFR-07 Robustness

Table 28 *NFR-07 Robustness*

ID	Description
NFR07-01	The system shall be designed to handle failures gracefully, with automatic failover mechanisms in place to maintain service continuity.
NFR07-02	The system shall implement comprehensive error handling strategies to manage unexpected issues without compromising user experience or data integrity.

2.3.28 NFR-08 Backup and Recovery

Table 29 *NFR-08 Backup and Recovery*

ID	Description
NFR08-01	The system shall perform backups(monthly) of all critical data.
NFR08-02	The system shall have a disaster recovery plan that guarantees service restoration within 4 hours after a major failure.

NFR08-03	The system shall regularly test data recovery processes to ensure reliability and effectiveness in restoring data after a loss event.

2.3.29 NFR-09 Monitoring & logging

Table 30 *NFR-09 Monitoring & logging*

ID	Description
NFR09-01	The system shall log all the action performed by adding time stamp and user id(if user is logged in).
NFR09-02	The system shall check security of failed login attempts or unauthorized access should be properly logged and checked.

2.3.30 NFR-10 Interoperability

Table 31 *NFR-10 Interoperability*

ID	Description
NFR10-01	The system must ensure that all web pages, UI components, and functionality are displayed and work correctly across these browsers.
NFR10-02	The application must be responsive and provide a consistent experience across all major mobile and tablet devices

2.4 Requirements traceability matric (RTM)

Table 32 *Requirement Traceability Matrix*

Requirement ID	Module	Requirement Description	Module	Implementation	Test Case ID	Test Status
FR-01	Authentication	Login functionality for users	Auth Module	Login API	TC01	Passed

FR-02	Authentication	Create new user accounts	Auth Module	Create Account API	TC02	Passed
FR-03	Authentication	Forgot Password feature	Auth Module	Forgot Password API	TC03	Passed
FR-04	Authentication	Change Password functionality	Auth Module	Change Password API	TC04	Passed
FR-05	Profile Management	Manage user profiles	Profile Module	Update Profile API	TC05	Passed
FR-06	API Services	Provide various API services	API Services Module	API Services Implementation	TC06	Passed
FR-07	API Services	Implement rate limiting and throttling for APIs	API Services Module	Rate Limiting & Throttling	TC07	Passed
FR-08	API Services	Authorization mechanisms for APIs	API Services Module	API Authorization	TC08	Passed
FR-09	API Services	Logging for all API interactions	API Services Module	API Logging Implementation	TC09	Passed
FR-10	User Management	Track and display user history	User Managem	User History API	TC10	Passed

			ent Module			
FR-11	API Services	Manage API keys for users	API Services Module	API Key Management API	TC11	Passed
FR-12	API Services	Monitor API performance and usage	API Services Module	API Monitoring API	TC12	Passed
FR-13	Text Analysis	Detect text reuse within documents	Text Analysis Module	Text Reuse Detection Module	TC13	Passed
FR-14	Plagiarism Detection	Detect plagiarism in user submissions	Plagiarism Module	Plagiarism Detection Module	TC14	Passed
FR-15	Communication	Provide a contact form for user queries	Communication Module	Contact Form Implementation	TC15	Passed
FR-16	Notifications	Send notifications to users	Notifications Module	Notifications Service	TC16	Passed
FR-17	Admin Dashboard	Provide an admin dashboard for system management	Admin Module	Admin Dashboard Implementation	TC17	Passed
FR-18	System Management	Manage system settings and configurations	System Management Module	System Management API	TC18	Passed
FR-19	Reporting	Generate various system reports	Reporting Module	Report Generation API	TC19	Passed

FR-20	Analytics	Provide analytics and	Analytics Module	Analytics API	TC20	Passed
		insights based on system data				

2.5 Use case descriptions

2.5.1 Login

Table 33 *UC-01 Login*

Use case ID 01	**Use case Name: Login**
Priority	**High**
Actors:	End-User (Student/Teacher, Writer/Author, Organization), Admin
Use Case Summary	Allows all registered users, including admins, to log into the system by providing valid email and password credentials. If a user fails to log in after multiple attempts, the system suggests using the forgot password feature.
Pre-condition:	1. No user is currently logged into the system. 2. User must have a registered account with a valid email and password.

Normal Flow of Events	**Alternative Path**
1.User navigates to the login page.	1a. If user is logged in already, then system redirect to home page.
2. User input their email and password in the input fields respectively.	
3. User submits the login form by pressing submit button.	

4. System validates the email format and process for credentials verification against the database.	4a In case the user forgets its password, he/she can click the "Forgot Password" link to start the forgot password step.
5. The system authenticates the user if the given credentials are valid, then shows a success message and redirects to the homepage.	5a. If credentials are invalid, the system displays an error message indicating login failure and highlighting the error fields with appropriate message .
6. System logs the successful login attempt with a timestamp and IP address.	
Exceptions / Alerts	
None	
Post Conditions	
Step#	**Description**
1- Successful Login	User (including admins) is authenticated, logged into the system, and redirected to the home page.
2- Failed Login	User gets an error message and is not allowed to leave the login page. If there are several failed attempts, then the user is asked to reset the password.
Use Case Cross References	
Includes	Validate Input Data, logging table
Extends	Display Error Message, Forgot Password

2.5.2 Create Account

Table 34 *UC-02 Create Account*

Use case ID 02	**Use case Name: Create Account**
Priority	**High**
Actors:	End-User (Student/Teacher, Writer/Author, Organization)
Use Case Summary	Enables new users to create an account by giving necessary personal information, validate input, verify email using, and ensuring that the email and phone number are unique in the system.
Pre-condition:	1. User is not currently signed in into the system. 2. User does not have an existing account associated with the provided email or phone number.

Normal Flow of Events	**Alternative Path**
1. User accesses the account creation page from the login interface or from home.	
2. User inputs their name, email, password, date of birth, phone number, and selects user type.	2a. If email or phone number already exists, system displays an error message indicating duplication by highlighting the specific input field with red border.
3. System validates the input fields for correct format email and phone number.	
4. If email and phone number are valid and unique, system generates a 6-digit OTP and sends it to user's given email address.	
5. User will get the OTP via email.	

6. User inputs the received OTP in the verification field.	
7. System validates the entered OTP.	
8. If OTP is valid, the system creates user account and redirects user to the login page.	8a. If OTP is invalid, system outputs an error message and allows user to retry.
9. User can also request new OTP after 60 seconds.	
9. System logs the account creation attempt with timestamp and IP address.	
Exceptions / Alerts	
1. If the email fails to send, system notifies the user and prompts to retry the account creation process. 2. If the user requests to resend the OTP, system sends a new OTP after a 60-second cooldown period.	
Post Conditions	
Step#	**Description**
1- Successful Account Creation	User's account is created, and user is redirected to login page.
2- Failed Account Creation	User receives appropriate error messages and remains on the account creation page to correct inputs or retry OTP verification.
Use Case Cross References	

Includes	Send Email, OTP Verification
Extends	Resend OTP

2.5.3 Forgot Password

Table 35 *UC-03 Forgot Password*

Use case ID 03	**Use case Name: Forgot Password**
Priority	**Medium**
Actors:	End-User (Student/Teacher, Writer/Author, Organization), System
Use Case Summary	Allows registered users who are not currently logged in to reset their password by verifying their identity through a One-Time Password (OTP) sent to their registered email address.
Pre-condition:	1. User is currently not logged into the system. 2. User must have a registered account with a valid email address in the system.

Normal Flow of Events	**Alternative Path**
1. The user selects the "Forgot Password" option on the login screen.	
2. User inputs their registered email address.	
3. System validates the email format and checks if it exists in the database.	

4. If the email is correct and exists, system will generate a 6-digit OTP.	
5. System sends the OTP code to the user's registered email address.	
6. User receives the OTP via email.	
7. User inputs the received OTP into the verification field.	
8. System verifies the entered OTP against the one sent to the email.	
9. Once the OTP is verified, the system asks the user to input a new password and enter it again for confirmation.	
10. The System verify the new password for complexity and ensures it is different from the old password.	
11. System updates the user's password in the database.	
12. System logs the password reset attempt with timestamp and IP address.	
13. The system successfully redirects the user to the login page with a success message.	
Exceptions / Alerts	
User can request new OTP after 60 seconds of generation of old OTP.	

Post Conditions	
Step#	**Description**
Successful Password Reset	The user's password is updated successfully in the system, and the user is sent to the login page.
Failed Password Reset	User receives appropriate error messages and remains on the password reset page to correct inputs or retry OTP verification.
Use Case Cross References	
Includes	OTP Generation and Sending, Password Validation
Extends	Resend OTP

2.5.4 Change Password

Table 36 *UC-04 Change Password*

Use case ID 04	**Use case Name: Change Password**
Priority	**Medium**
Actors:	End-User (Student/Teacher, Writer/Author, Organization), System
Use Case Summary	Allows authenticated users to change their current password by verifying their existing password and ensuring the new password meets complexity requirements and is different from the old password.
Pre-condition:	1. The user must be logged into the system. 2. The user must know their current password.
Normal Flow of Events	**Alternative Path**
1. User navigates to the profile to change password and select change password.	

2. The user input their current password.	
3. User input the new password and confirm password.	
4. System verifies the current password.	4a. If the user enter the an invalid current password system delivers a message of error.
5. System validates the new password for complexity and ensures it is different from the old password.	5a. If the new password doesn't meet the complexity requirements, the system outputs an appropriate error message and highlight the input field.
6. System will update the password in the database.	
7. System logs the password change attempt with timestamp and user identifier.	
8. System removes all active user sessions.	
9. System notifies the user of the successful password change.	
Exceptions / Alerts	
None	
Post Conditions	
Step#	**Description**
Successful Password Change	User's password is updated, all sessions are terminated, and the user is notified.
Failed Password Change	User remains on the change password page with error messages prompting corrective actions.

Use Case Cross References	
Includes	Password Complexity Enforcement, Session Termination
Extends	Password History Check

2.5.5 Profile Management

Table 37 *UC-05 Profile Management*

Use case ID 05	**Use case Name: Profile Management**
Priority	**Medium**
Actors:	End-User (Student/Teacher, Writer/Author, Organization), System

Use Case Summary	Enables users to view and update their profile information, including personal details and profile picture, while ensuring that changes are authenticated and logged.
Pre-condition:	1. User must be logged into the system. 2. User has an existing profile with updatable fields.

Normal Flow of Events	**Alternative Path**
1. The user proceeds to the profile management section.	
2. The system outputs the current profile information.	
3. User click on to update button to update information. (e.g., phone number, profile picture).	
4. User inputs the new information.	4a. If the current password is incorrect, system displays an error message.
5. System validates the input fields.	5a. If the input data is invalid, system displays appropriate error messages.

6. If validation passes, system updates the profile in the database.	6a. If the user cancels the update, system retains the existing profile information.
7. System sends a notification email about the successful update.	
8. System logs the profile management activity with timestamp and user identifier.	
Exceptions / Alerts	
None	
Post Conditions	
Step#	**Description**
Successful Update	User's profile information is updated, and a notification is sent.
Failed Update	User remains on the change password page with error messages prompting corrective actions.
Use Case Cross References	
Includes	Email Notification, Input Validation
Extends	Upload Profile picture

2.5.6 API Services

Table 38 *UC-06 API Services*

Use case ID 06	**Use case Name: API Services**
Priority	**Low**
Actors:	API Users, System

Use Case Summary	Provides API services to external platforms, enabling them to identify text reuse and plagiarism in Urdu documents by sending requests and receiving processed results.
Pre-condition:	1. External platform must have a valid API key. 2. External platform must comply with rate limiting policies. 3. User has access to the API documentation.

Normal Flow of Events	**Alternative Path**
1. External platform sends a request to the API with the necessary data (e.g., text documents).	
2. System authenticates the API key.	2a. If the API key is invalid, system returns an authentication error.
3. System processes the request using AI models to detect text reuse.	
4. System returns the results to the external platform.	4a. If the request exceeds rate limits, system throttles the request and returns a rate limit exceeded error.
5. System logs the API request and response with timestamp and user identifier.	
6. If validation passes, system updates the profile in the database.	
7. System sends a notification email about the successful update.	
8. System logs the profile management activity with timestamp and user identifier.	

Exceptions / Alerts
If the request data is malformed, system returns a validation error.

Post Conditions	
Step#	**Description**
Successful Request	External platform receives processed results indicating text reuse and plagiarism detection.
Failed Request	External platform receives appropriate error messages indicating the reason for failure.
Use Case Cross References	
Includes	Authorization for API, API Logging
Extends	Rate Limiting and Throttling

2.5.7 Rate Limiting for API

Table 39 *UC-07 Rate Limiting for API*

Use case ID 07	**Use case Name: Rate Limiting for API**
Priority	**Medium**
Actors:	API Users, System
Use Case Summary	Implement the rate limiting to restrain the number of API calls from a single user within the specified time frame to restrict abuse and ensure fair utilization of the API resources.
Pre-condition:	1. API user must have a valid API key. 2. API user is making requests to the API services.

Normal Flow of Events	**Alternative Path**
1. API user sends a request to the API.	
2. System checks the number of requests made by the user in the current time frame.	2a. If the user is nearing the rate limit, system may send a warning notification.
3. If the user is within the limit, system processes the request.	

4. If the user exceeds the limit, system throttles the request and returns a rate limit exceeded error.	4a. If the user requests to exceed the rate limit (if permitted), system evaluates and grants access based on predefined policies.
5. System logs the rate limiting event with timestamp and user identifier.	
Exceptions / Alerts	
None	
Post Conditions	
Step#	**Description**
Within Limit	API request is processed successfully.
Exceeded Limit	API request is throttled, and the user receives a rate limit exceeded error message.
Use Case Cross References	
Includes	Logging Rate Limits, Notification of Nearing Limits
Extends	Dynamic Rate Limiting

2.5.8 Authorization for API

Table 40 *UC-08 Authorization for API*

Use case ID 08	**Use case Name: Authorization for API**
Priority	**Medium**
Actors:	API Users, System
Use Case Summary	Requires API users to authenticate using provided API keys to securely access API services, confirming that access is available only to authorized users can utilize the API functionalities.

Pre-condition:	1. API user must possess a valid API key. 2. API user is making a request to the API services.

Normal Flow of Events	**Alternative Path**
1. API user includes their API key in the request headers or parameters.	
2. System retrieves and verifies the API key.	
3. If the API key is valid, system grants access to the requested API service.	3a. If the API key is invalid or expired, system denies access and returns an authentication error.
4. System processes the request and returns the appropriate response to the API user.	4a. If the API key has insufficient permissions for the requested service, system returns an authorization error.
5. System logs the authorization attempt with timestamp and user identifier.	

Exceptions / Alerts
If the API key is missing from the request, system returns an authentication error.

Post Conditions	
Step#	**Description**
Authorized Access	API user gains access to the requested service.
Unauthorized Access	API user is denied access and receives an error message indicating the reason for denial.

Use Case Cross References	
Includes	API Logging, Permission Checks
Extends	API Key Revocation

2.5.9 API Logging

Table 41 *UC-09 API Logging*

Use case ID 09	**Use case Name: API Logging**
Priority	**Medium**
Actors:	API Users, System, Admin
Use Case Summary	Logs all API requests, responses, and user activities to ensure traceability, accountability, and security for each interaction with the API services.
Pre-condition:	1. API users must have access to the API services. 2. System must have logging capabilities enabled.

Normal Flow of Events	**Alternative Path**
1. API user sends a request to the API.	
2. System processes the request and generates a response.	2a. If logging fails (e.g., due to storage issues), system alerts the admin and retries logging.
3. System logs the request details, including timestamp, API key, request parameters, and response status.	3a. If sensitive information is included in the logs, system ensures data masking or encryption to protect user privacy.
4. System stores the log securely to prevent tampering.	
5. Admin can access and review the logs through the admin dashboard.	

Exceptions / Alerts
If the logging service is unavailable, system queues the logs and attempts to log them once the service is restored.
Post Conditions

Step#	Description
API Integration	All API interactions are recorded and stored for auditing and monitoring purposes.
Admin Access	Admins have access to detailed logs for analysis and troubleshooting.
Use Case Cross References	
Includes	Log Retrieval, Secure Storage of Logs
Extends	Log Analysis

2.5.10 User History

Table 42 *UC-10 User History*

Use case ID 10 Use case Name: User History	
Priority **Medium**	
Actors: End-User (Student/Teacher, Writer/Author, Organization), System	
Use Case Summary	Tracks and saves user history, including API calls and actions performed via the API, allowing users to review their past activities and providing admins with insights for monitoring and analysis.
Pre-condition:	1. User must be authenticated and logged into the system. 2. System must have tracking capabilities enabled.
Normal Flow of Events	**Alternative Path**
1. User performs an action or makes an API call.	
2. System records the action details, including timestamp, action type, and parameters.	
3. System stores the history in the user's profile.	

4. User can access their history through their dashboard.	4a. If the user requests to clear their history, system deletes the relevant records after confirmation. 4b. If the user has extensive history, system provides filtering and search options for easier navigation.
5. System logs the history tracking event.	
Exceptions / Alerts	
If the system fails to record an action, it notifies the user and retries logging the action.	
Post Conditions	
Step#	**Description**
Successful Tracking	User's actions and API calls are recorded and accessible for review.
Failed Tracking	User is informed of the unsuccessful tracking attempt and can retry or report the issue.
Use Case Cross References	
Includes	Action Logging, History Retrieval
Extends	Advanced Analytics, Privacy Controls

2.5.11 API Key Management

Table 43 *UC-11 API Key Management*

Use case ID 11	**Use case Name: API Key Management**
Priority	**Medium**
Actors:	API Users, System

Use Case Summary	Allows users to generate, regenerate, and manage their API keys for secure access to API services, including the ability to create multiple keys for different applications or teams and download them for backup and documentation.
Pre-condition:	1. User must be logged into the system. 2. User has the necessary permissions to manage API keys.

Normal Flow of Events	**Alternative Path**
1. User navigates to the API key management section.	
2. User chooses to generate a new API key.	
3. System generates a unique API key.	
4. User can create multiple API keys for different applications or teams.	3a. If the user attempts to generate more API keys than allowed, system outputs an error message. 4b. If the user requests to regenerate an existing API key, system invalidates the old key and generates a new one.
5. User can download their API keys in CSV or PDF format for backup and documentation purposes.	
6. System logs the API key generation/regeneration event with timestamp and user identifier.	

Exceptions / Alerts

- If the download fails, the system outputs the user and prompts to retry.
- If the API key generation service is unavailable, system alerts the user and suggests alternative actions.

Post Conditions

Step#	**Description**

Successful API Key Management	User successfully generates, regenerates, and downloads API keys.
Failed API Key Management	User receives error messages and can retry the actions as necessary.
Use Case Cross References	
Includes	API Logging, Permission Checks
Extends	API Key Revocation, Usage Quotas

2.5.12 API Monitoring

Table 44 *UC-12 API Monitoring*

Use case ID 12	**Use case Name: API Monitoring**
Priority	**Medium**
Actors:	Admin , API Users
Use Case Summary	Provides users with detailed usage analytics, including total API calls and remaining quota, and allows admins to monitor API usage, identify peak usage times, and manage API services based on usage patterns.
Pre-condition:	1. User must have an active account with API access. 2. Admin must have access to monitoring tools and dashboards. 3. Admin and User must be login.
Normal Flow of Events	**Alternative Path**
1. User accesses the API monitoring dashboard.	
2. System displays usage analytics, including total API calls, remaining quota.	
3. Admin accesses the dashboard and select user to monitor.	

4. System displays detailed API usage statistics, including the most-used endpoints and peak usage times.	3a. If the user has exceeded their quota but requests an extension, admin can review and adjust quotas as necessary.
5. Admin reviews the analytics to identify trends and potential issues.	
6. If a user exceeds their quota, system automatically suspends their API service and sends an email notification.	
7. System logs all monitoring activities with timestamps and user identifiers.	

Exceptions / Alerts
If the monitoring service fails, system alerts the admin and attempts to restore monitoring capabilities.

Post Conditions	
Step#	**Description**
User Analytics Accessed	Users can view their API usage statistics.
Admin Insights	Admins have access to comprehensive analytics for monitoring and managing API services.
Quota Enforcement	Users exceeding their quotas are notified and restricted from further API access until resolved.

Use Case Cross References	
Includes	Email Notification for Quota Exceeding, Logging Monitoring Activities
Extends	Automated Alerts, Customize Dashboards

2.5.13 Reuse Detection

Table 45 *UC-13 Reuse Detection*

Use case ID 13	**Use case Name: Reuse Detection**
Priority	**High**
Actors:	End-User (Student/Teacher, Writer/Author, Organization), System,

Use Case Summary	Enables users to detect text reuse in Urdu by inputting paragraphs manually or uploading documents/images, processing the texts using AI models to identify reused content, and generating comprehensive reports.
Pre-condition:	1. User must be logged into the system. 2. User has access to the reuse detection tool. 3. System has the necessary AI models and OCR capabilities integrated.

Normal Flow of Events	**Alternative Path**
1. User navigates to the reuse detection tool.	
2. User inputs two separate Urdu paragraphs manually or uploads images/documents containing text.	2a. If the uploaded documents are in unsupported formats, system prompts the user to upload compatible formats.
3. If images are uploaded, system performs OCR to extract text.	3a. If OCR fails to extract text from images, system notifies the user and suggests uploading clearer images.
4. System displays the extracted text for user review and allows editing.	
5. User initiates the text reuse detection process.	
6. System processes the texts using AI models (e.g., Sentence Transformer) to identify reused content.	

7. System calculates the percentage of content reused between the two texts.	
8. System classifies the reuse into predefined categories (PD, WD, ND).	
9. System generates a comprehensive report detailing the findings, including reuse percentages and sources.	
10. User can download the report in desired formats (e.g., PDF).	
11. System logs the reuse detection activity with timestamp and user identifier.	
Exceptions / Alerts	
If the AI model fails to process the texts, system will output an error message and logs the failure for troubleshooting.	
Post Conditions	
Step#	**Description**
Successful Detection	User receives a detailed report on text reuse, including percentages and sources, and can download the report.
Failed Detection	User is informed of the failure and prompted to retry or correct inputs.
Use Case Cross References	
Includes	OCR Processing, Report Generation
Extends	Multi-Document Detection, Editing Extracted Text

2.5.14 Plagiarism Detection

Table 46 *UC-14 Plagiarism Detection*

Use case ID 14	**Use case Name: Plagiarism Detection**
Priority	**High**
Actors:	End-User (Student/Teacher, Writer/Author, Organization), System,
Use Case Summary	Allows users to upload text files for plagiarism by comparing the content against online sources using search engines and databases, calculating the percentage of matched content, identifying source URLs, and generating downloadable reports.
Pre-condition:	1. User must be signed in into the system. 2. User has access to the plagiarism detection tool. 3. System has access to online search engines and databases for comparison.

Normal Flow of Events	**Alternative Path**
1. User navigates to the plagiarism detection tool.	
2. User uploads a text file containing Urdu content.	2a. If the format of uploaded files is unsupported, system prompts the user to upload a compatible format.
3. System extracts the text from the uploaded file.	
4. System processes the extracted text by comparing it against online sources using search engines and databases.	4a. If no matches are found, system informs the user that no plagiarism was detected.
5. System calculates the percentage of text that matches external sources to determine the level of plagiarism.	

6. System identifies and retrieves the URLs or sources from where the content is potentially copied.	
7. System generates a comprehensive plagiarism report that includes the percentage of matched content and corresponding source URLs.	
8. User can download the plagiarism report in formats such as PDF.	
9. System logs the plagiarism detection activity with timestamp and user identifier.	

Exceptions / Alerts
If the comparison process fails because of network issues, system notifies the user and logs the failure for troubleshooting.

Post Conditions	
Step#	**Description**
Successful Detection	User receives a plagiarism report indicating matched content percentages and source URLs, and can download the report.
Failed Detection	User is informed of the failure and prompted to retry or correct inputs.

Use Case Cross References	
Includes	Text Extraction, Report Generation

Extends	Source URL, Bulk Upload

2.5.15 Contact From

Table 47 *UC-15 Contact From*

Use case ID 15	**Use case Name: Contact Form**	
Priority	**Low**	
Actors:	Unregistered User, End-User (Student/Teacher, Writer/Author, Organization), System, Admin	
Use Case Summary	Enables users to create and submit queries or complaints via a contact form, categorize their messages, view submitted queries and admin responses, track the status of their queries, and receive email notifications upon status updates.	
Pre-condition:	1. User must be logged into the system (optional based on design). 2. User has access to the contact form tool.	
Normal Flow of Events		**Alternative Path**
1. User navigates to the contact form.		
2. User fills in the query or complaint details.		
3. User selects a message category (e.g., billing, technical support, general).		3a. If the user does not select a message category, system prompts the user to select one.
4. User submits the form.		4a. If the user attempts to submit an incomplete form, system displays appropriate error messages.
5. System sends the query to the admin.		
6. Admin reviews and responds to the query.		
7. User can view their submitted queries and corresponding admin responses.		

8. User can track the status of their queries (e.g., pending, responded, resolved).	
9. System sends email notifications to users whenever the status of their contact form query is updated.	
10. System logs the contact form activities with timestamps and user identifiers.	
Exceptions / Alerts	
1. If the system fails to send the query to the admin, it notifies the user and logs the failure for troubleshooting. 2. • If the admin is unavailable to respond immediately, system updates the query status accordingly.	
Post Conditions	
Step#	**Description**
Successful Submission	User's query is submitted, responded to by admin, and the status is updated and tracked.
Failed Submission	User receives error messages and remains on the contact form page to correct inputs or retry submission.
Use Case Cross References	
Includes	Email Notifications, Status Tracking
Extends	Message Categorization, Admin Response Management

2.5.16 Notifications

Table 48 *UC-16 Notifications*

Use case ID 16	**Use case Name: Notifications**
Priority	**Low**

Actors: End-User (Student/Teacher, Writer/Author, Organization), System, Admin	
Use Case Summary	Sends notifications to users about important events, actions, and tasks (e.g., exceeding API limits, account creation, task completion) via email, and allows admins to send custom notifications to users.
Pre-condition:	1. The user must be registered in the system. 2. The admin must have permissions to send notifications.

Normal Flow of Events	**Alternative Path**
1. System identifies an important event or action (e.g., user exceeds API limit, account creation, task completion).	
2. System generates a notification message.	2a. If the user has opted out of certain notifications, system respects user preferences and does not send those notifications.
3. System sends an email notification to the user.	3a. If the email fails to send, system retries and notifies the admin of the failure.
4. Admin can access the notification feature in the admin dashboard.	
5. Admin creates and sends custom notifications to selected users.	
6. Users receive and can view the notifications.	
7. System logs all notification activities with timestamps and user identifiers.	

Exceptions / Alerts
If the notification content is invalid or contains prohibited information, system rejects the notification and alerts the admin.

Post Conditions

Step#	Description
Successful Notification	Users receive timely and relevant notifications about important events and actions.
Failed Notification	Users are informed of the notification failure, and admins are alerted for troubleshooting.

Use Case Cross References	
Includes	Email Sending, Notification Logging
Extends	Custom Admin Notifications, User Preferences Management

2.5.17 Admin Dashboard

Table 49 *UC-17 Admin Dashboard*

Use case ID 17 Use case Name: Admin Dashboard	
Priority **Medium**	
Actors: Admin, System	
Use Case Summary	Provides admins with a centralized dashboard to monitor system metrics, user activities, API usage, recent activities, and access logs, enabling efficient management and oversight of the application.
Pre-condition:	1. Admin must be logged into the system with appropriate administrative privileges. 2. System must have dashboard capabilities enabled and populated with relevant data.

Normal Flow of Events	Alternative Path
1. Admin logs into the system.	
2. Admin navigates to the admin dashboard section.	

3. System displays various metrics such as total users, API usage statistics, recent user activities, and access logs.	3a. If the admin requests specific data not displayed on the dashboard, system provides options to customize the dashboard or access detailed reports. 3b. If the dashboard fails to load data, system notifies the admin and logs the failure for troubleshooting.
4. Admin interacts with dashboard widgets to drill down into specific metrics or data points.	
5. Admin can perform administrative tasks directly from the dashboard (e.g., managing users, viewing logs, generating reports).	
6. System updates the dashboard in real-time as new data becomes available.	
7. System logs all admin dashboard accesses and actions for security and auditing purposes.	
Exceptions / Alerts	
If the admin lacks necessary permissions for certain dashboard functionalities, system restricts access and displays an appropriate message.	
Post Conditions	
Step#	**Description**
Successful Dashboard Access	Admin has comprehensive visibility into system metrics and can perform necessary administrative tasks.

Failed Dashboard Access	Admin is informed of access issues and can attempt to resolve them or seek assistance.
Use Case Cross References	
Includes	Real-Time Data Display, Log Management
Extends	Customizable Widgets, Data Exporting

2.5.18 System Management

Table 50 *UC-18 System Management*

Use case ID 18	**Use case Name: System Management**
Priority	**Medium**
Actors:	Admin, System
Use Case Summary	Allows admins to manage system users, handle complaints and queries, and perform other administrative tasks through the admin dashboard, ensuring smooth operation, security, and maintenance of the paraphrase detection tool.
Pre-condition:	1. Admin must be logged into the system with appropriate administrative privileges. 2. System must have user and task management capabilities enabled.

Normal Flow of Events	**Alternative Path**
1. Admin accesses the system management section from the admin dashboard.	
2. Admin can view the list of system users, including their details and statuses.	
3. Admin can create new user accounts or update existing ones.	3a. If the admin encounters an issue while managing users or tasks, system provides troubleshooting options or error notifications.

4. Admin can delete or deactivate user accounts as necessary.	
5. Admin can view and manage complaints and queries submitted by users.	
6. Admin can assign, prioritize, and respond to user queries and complaints.	
7. Admin can perform other administrative tasks such as configuring system settings, managing API keys, and monitoring system health.	7a. If the admin attempts to perform an action without the necessary permissions, system displays an error message.
8. System logs all system management activities with timestamps and admin identifiers.	

Exceptions / Alerts
If the system fails to update user information or handle a complaint, system notifies the admin and logs the failure for troubleshooting.

Post Conditions	
Step#	**Description**
Successful System Management	Admin effectively manages users, complaints, queries, and system settings.
Failed System Management	Admin receives error messages and can retry or seek assistance to resolve issues.

Use Case Cross References	
Includes	User Management, Complaint and Query Management
Extends	Bulk User Operations, Automated Task Assignments

2.5.19 Report Generation

Table 51 *UC-19 Report Generation*

Use case ID 19	**Use case Name: Report Generation**
Priority	**Medium**
Actors:	Admin, System
Use Case Summary	Enables admins to generate various reports filtered by date range, user, or activity type, and export them in common formats (e.g., CSV, PDF). Additionally, generates detailed reports for API usage, including the most-used endpoints and peak usage times.
Pre-condition:	1. Admin must be logged into the system with appropriate administrative privileges. 2. System must have reporting capabilities enabled and access to relevant data sources.

Normal Flow of Events	**Alternative Path**
1. The admin proceeds to the report generation section in the admin dashboard.	
2. The admin selects the type of report to generate (e.g., user activity, API usage).	2a. If the admin requests a report type that requires additional permissions, system prompts for necessary access.
3. Admin applies desired filters such as date range, specific users, or activity types.	3a. If the selected filters return no data, system notifies the admin and suggests adjusting the filters.
4. The system retrieves, and the compiles relevant data based on the selected criteria.	
5. The system will generate the report in the mentioned format (e.g., CSV, PDF).	
6. Admin reviews the generated report.	
7. Admin exports or downloads the report for external use.	

8. System logs the report generation activity with timestamp and admin identifier.	
Exceptions / Alerts	
If the report generation process fails due to data retrieval issues, system notifies the admin and logs the failure for troubleshooting.	
Post Conditions	
Step#	**Description**
Successful Report Generation	Admin receives the generated report in the desired format and can use it for analysis, record-keeping, or decision-making.
Failed Report Generation	Admin is informed of the failure and can adjust filters or seek assistance.
Use Case Cross References	
Includes	Data Filtering, Report Exporting
Extends	Automated Report Scheduling, Custom Report Templates

2.5.20 Analytics

Table 52 *UC-20 Analytics*

Use case ID 20	**Use case Name: Analytics**
Priority	**Medium**
Actors:	Admin, System
Use Case Summary	Provides Admin with usage analytics, including token usage, remaining quota, and offers insights into user activity and service usage to inform system improvements and decision-making.
Pre-condition:	1. Admin must be logged into the system.

Normal Flow of Events	Alternative Path
1. Admin navigates to the analytics section.	1a. If the user requests analytics data without being admin permissions, system restricts access and notifies the user.
2. Admins Selects Analytics report type.	
3. System retrieves and displays usage analytics, including token usage, remaining quota, and document history	3a. If the selected filters return no data, system notifies the admin and suggests adjusting the filters.
4. Admin can apply filters.	
5. System can generate custom analytics reports based on specific metrics or criteria.	
6. System allows exporting of analytics data in formatted document for further analysis.	6a. This Document can also be mailed.
7. System logs all analytics access and actions with timestamps and user identifiers.	

Exceptions / Alerts
If the report generation process fails due to data retrieval issues, system notifies the admin and logs the failure for troubleshooting.

Post Conditions
Admins can export analytics data for external analysis.

Use Case Cross References	
Includes	Usage Tracking, Data Visualization
Extends	Real-Time Email Alerts

2.6 Use case design

2.6.1 Login

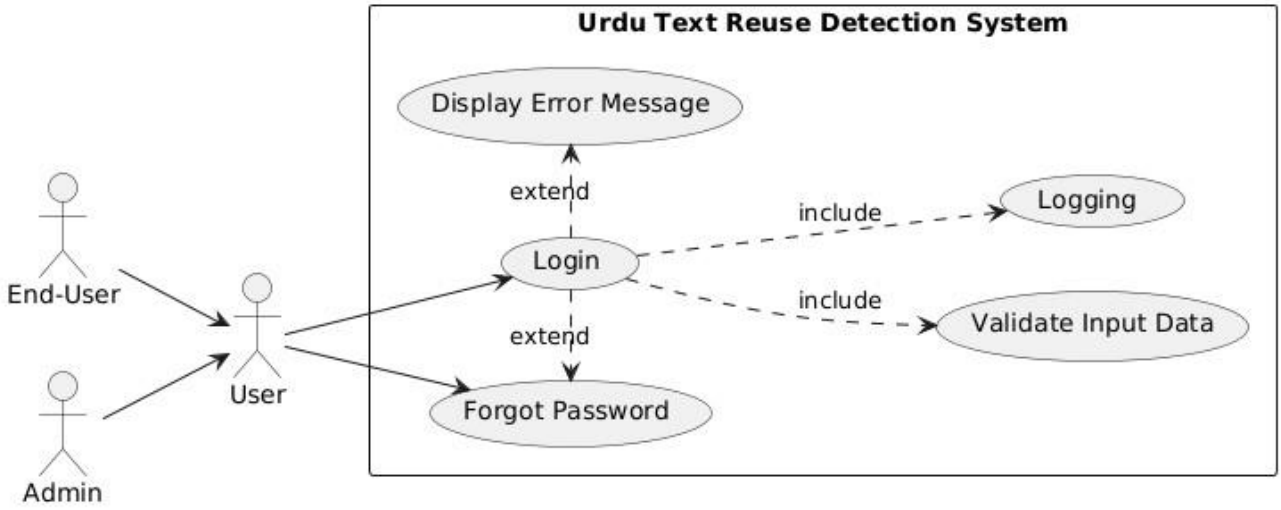


Figure 4 *Login Use Case Diagram*

2.6.2 Create Account

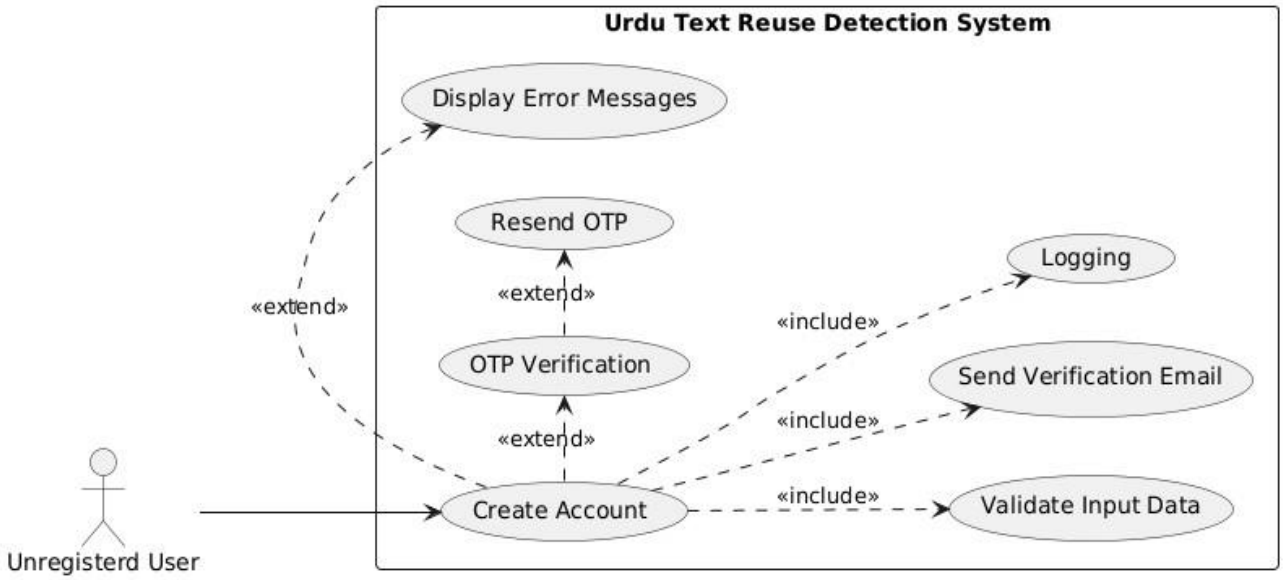


Figure 5 *Create Account Use Case*

2.6.3 Forget Password

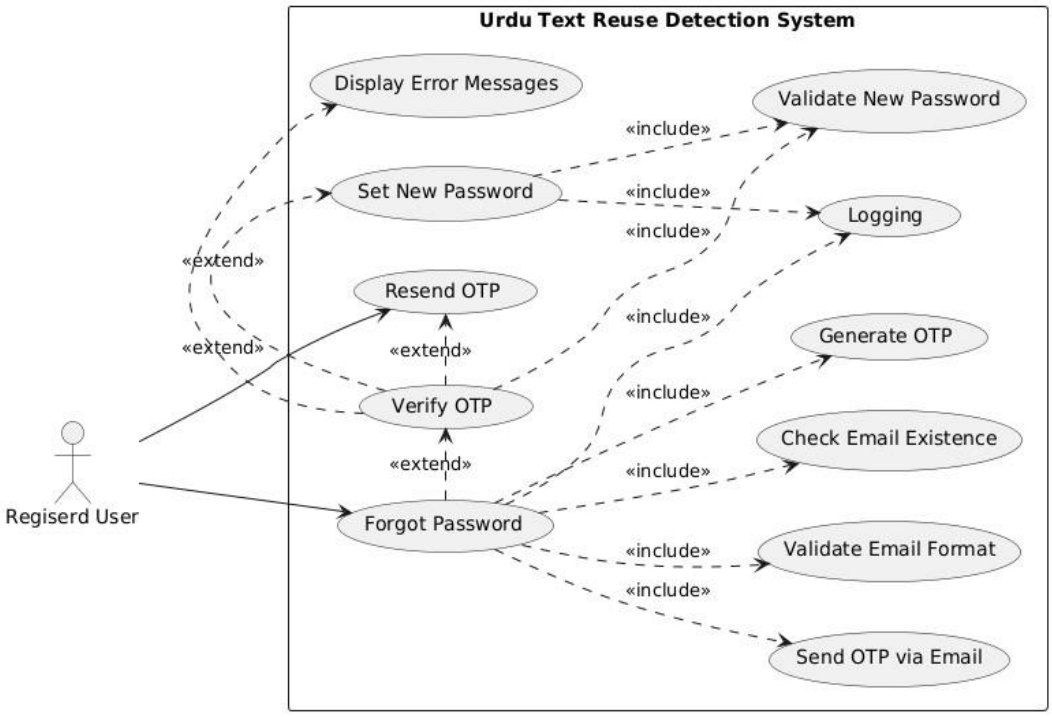


Figure 6 *Forget Password Use Case Diagram*

2.6.4 Change Password

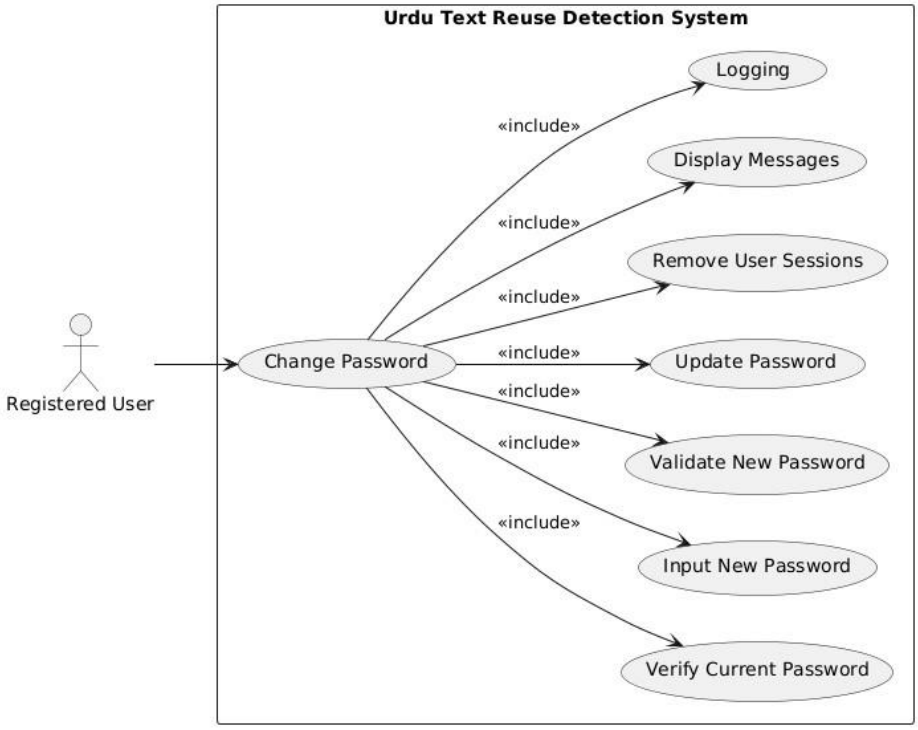


Figure 7 *Change Password Use Case Diagram*

2.6.5 Update Profile

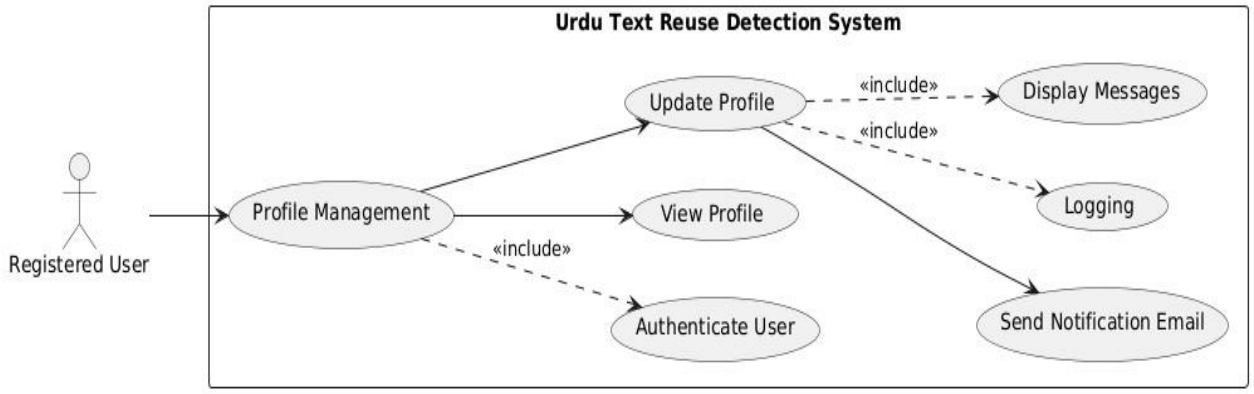


Figure 8 *Update Profile Use Case Diagram*

2.6.6 API Services

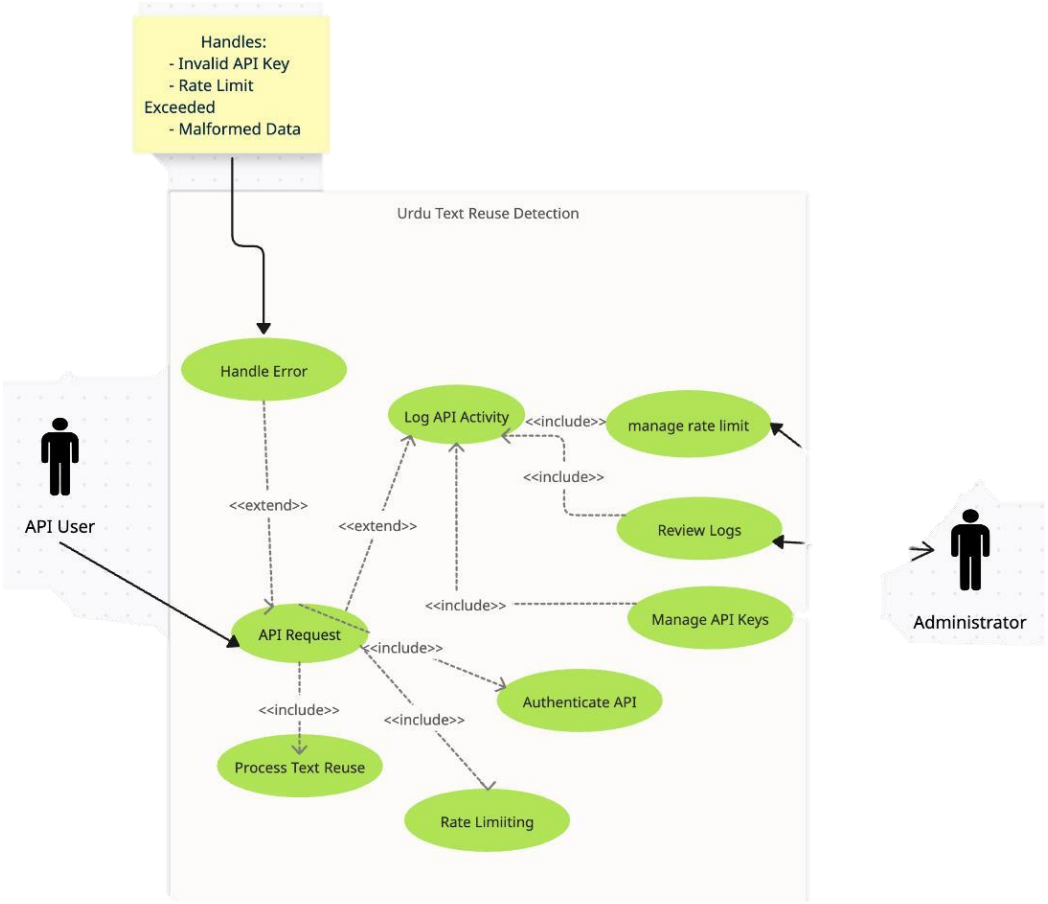


Figure 9 *API Services Use Case Diagram*

2.6.7 Rate Limiting for API

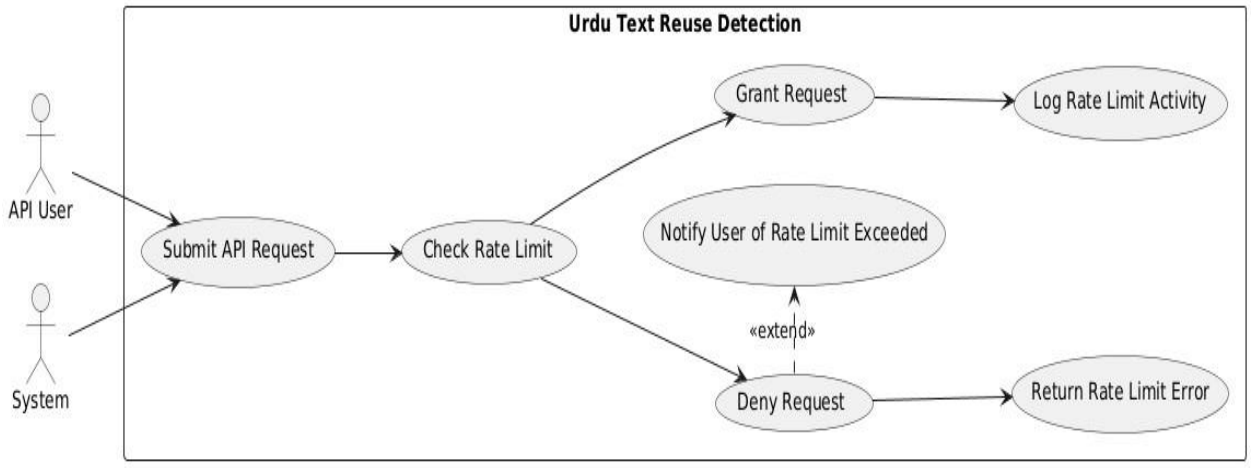


Figure 10 *Rate Limiting for API Use Case Diagram*

2.6.8 Authorization for API

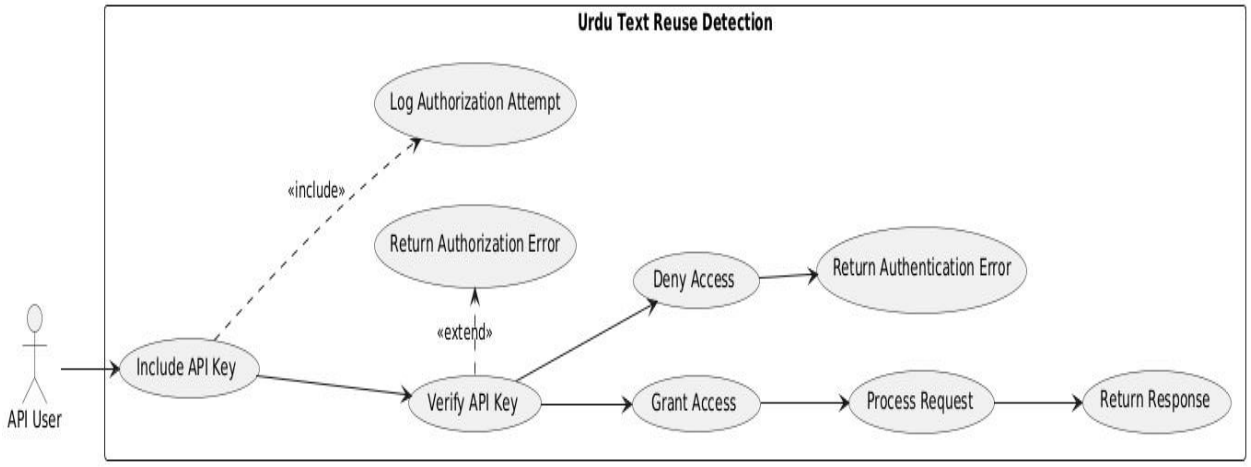


Figure 11 *API Authorization Use Case Diagram*

2.6.9 API logging

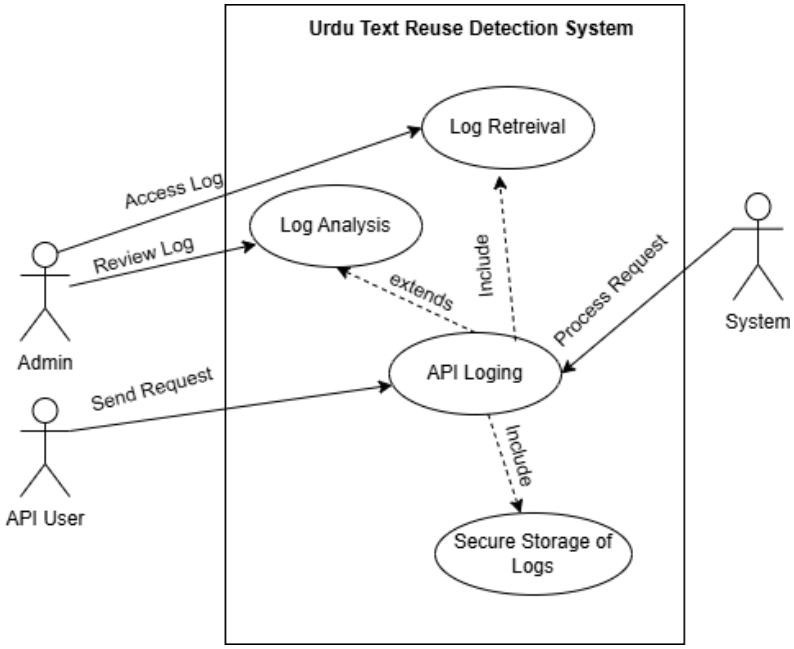


Figure 12 *API logging Use Case Diagram*

2.6.10 User history

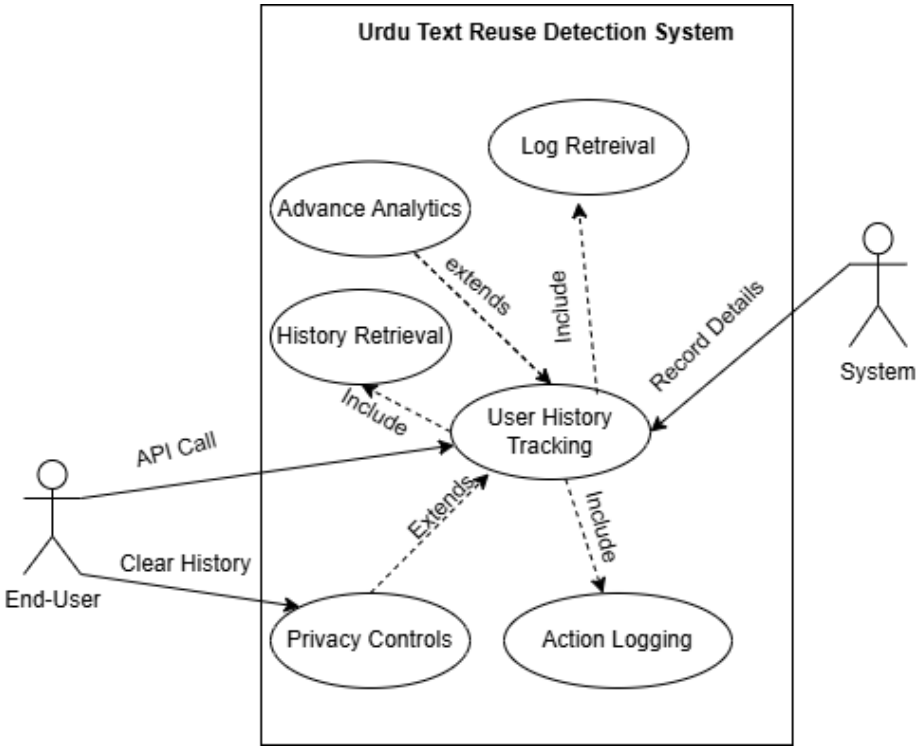


Figure 13 *User History Use case Diagram*

2.6.11 API key management

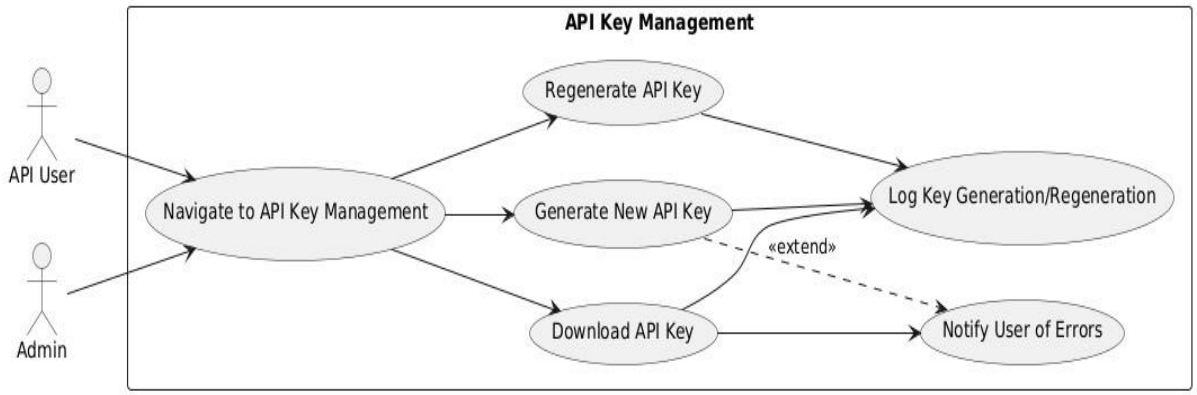


Figure 14 *API key management Use case Diagram*

2.6.12 API Monitoring

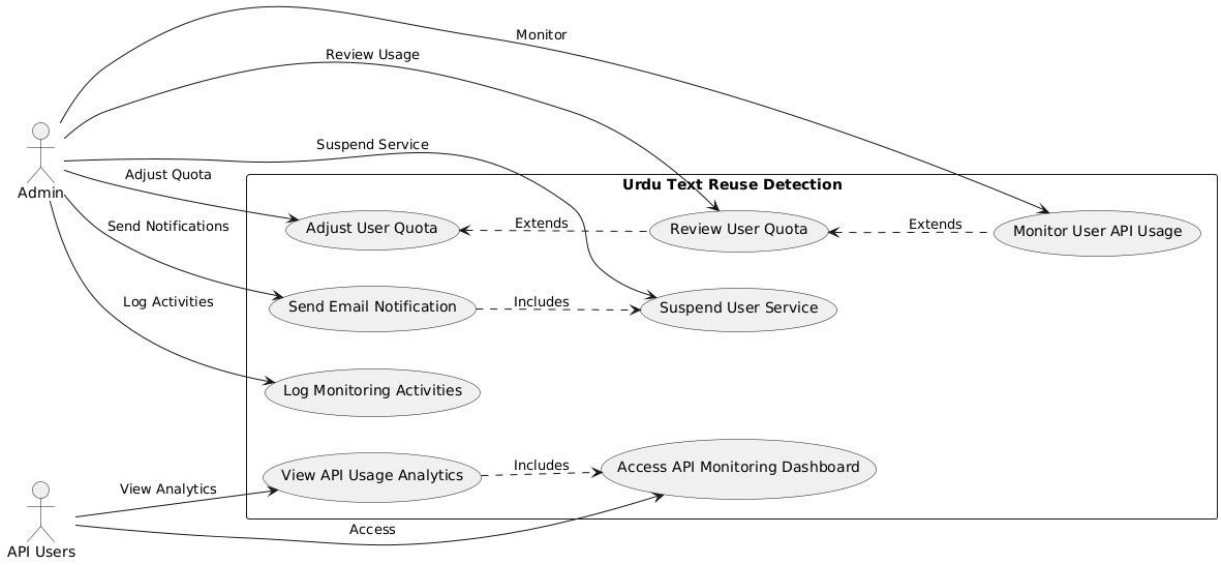


Figure 15 *API Monitoring Use Case Diagram*

2.6.13 Text Reuse Detection

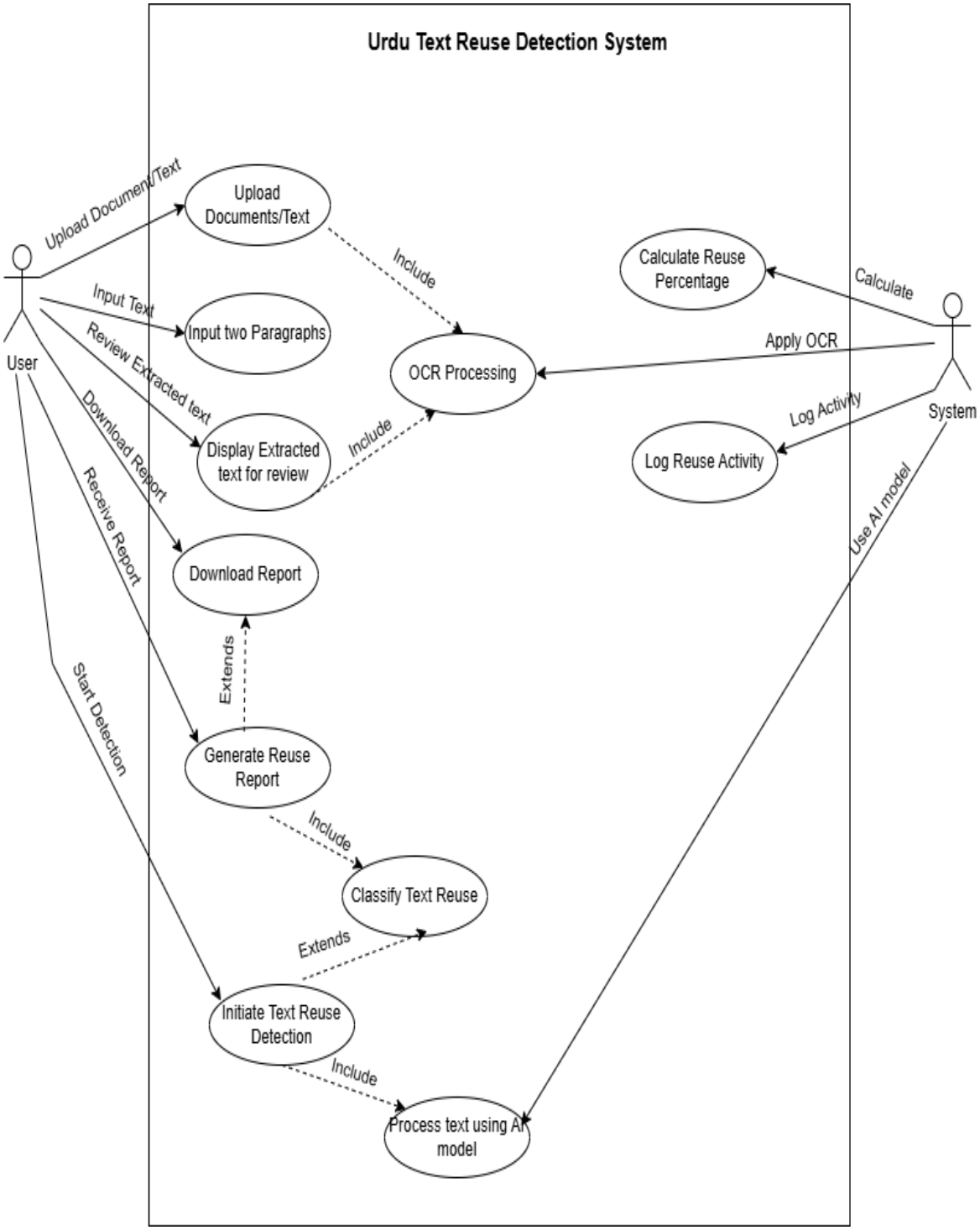


Figure 16 *Text Reuse Detection Use Case Diagram*

2.6.14 Plagiarism Detection

Figure 17 *Plagiarism Detection Use case Diagram*

2.6.15 Contact

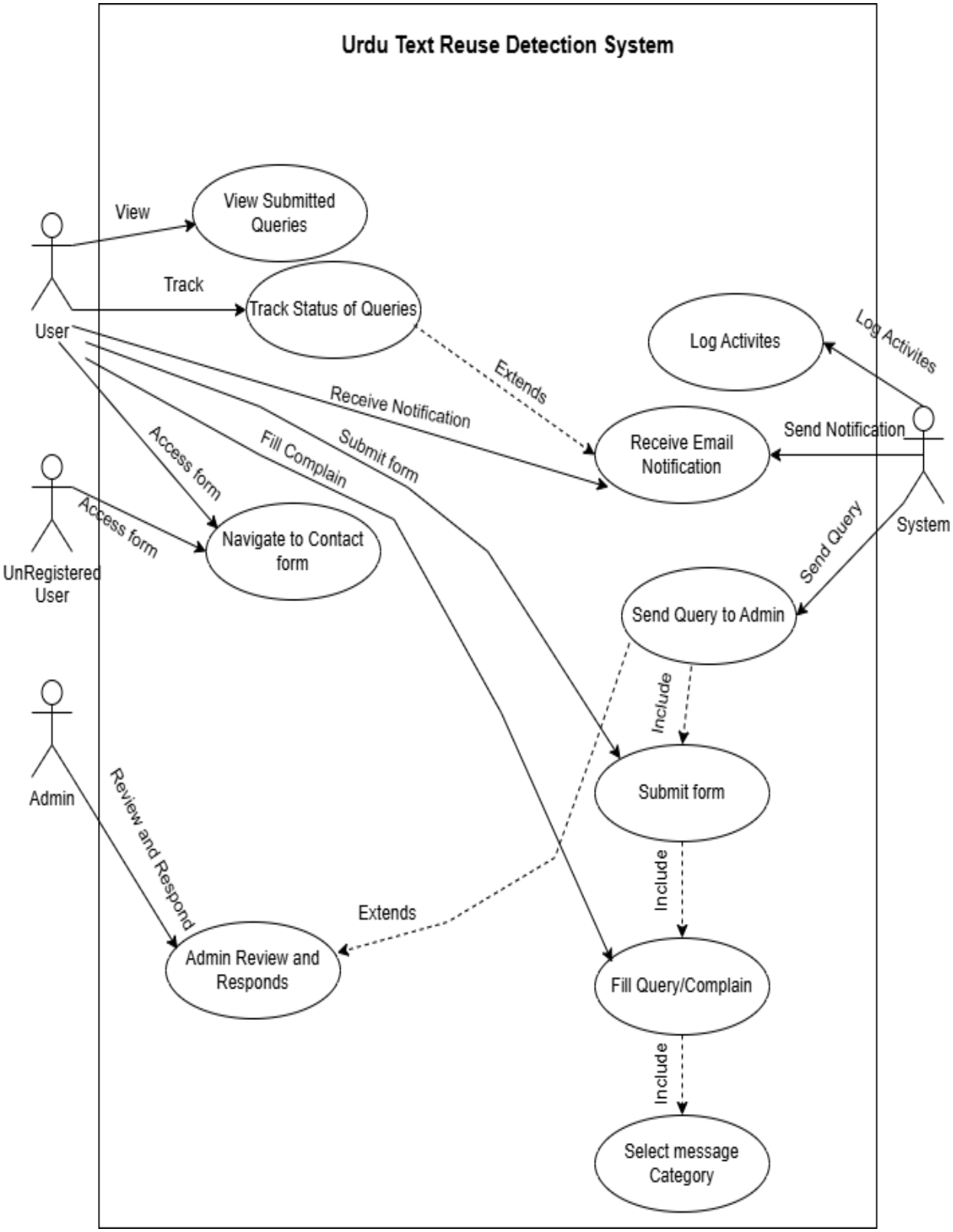


Figure 18 *Contact Use Case Diagram*

2.6.16 Notifications

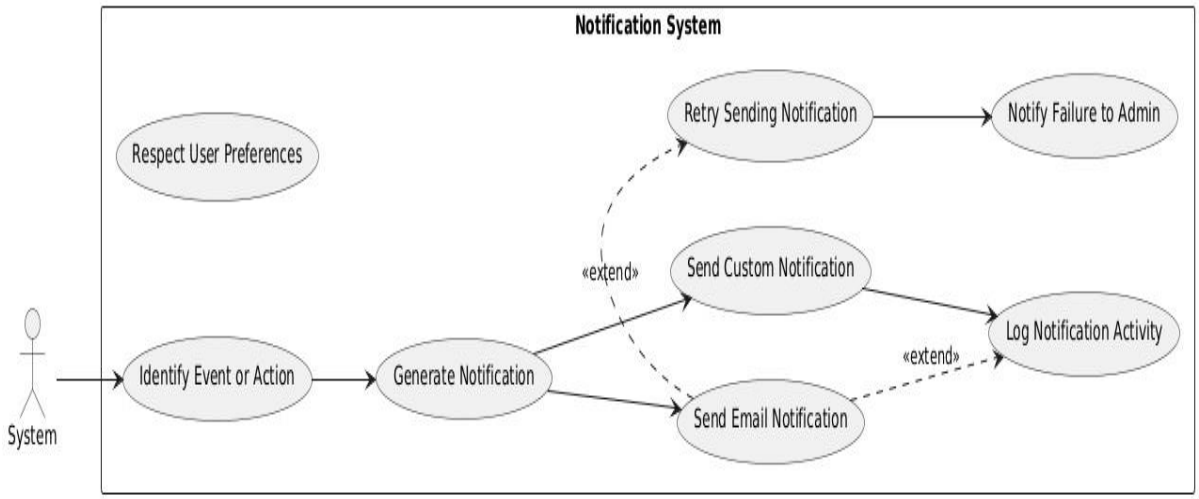


Figure 19 *Notifications Use Case Diagram*

2.6.17 Admin Dashboard

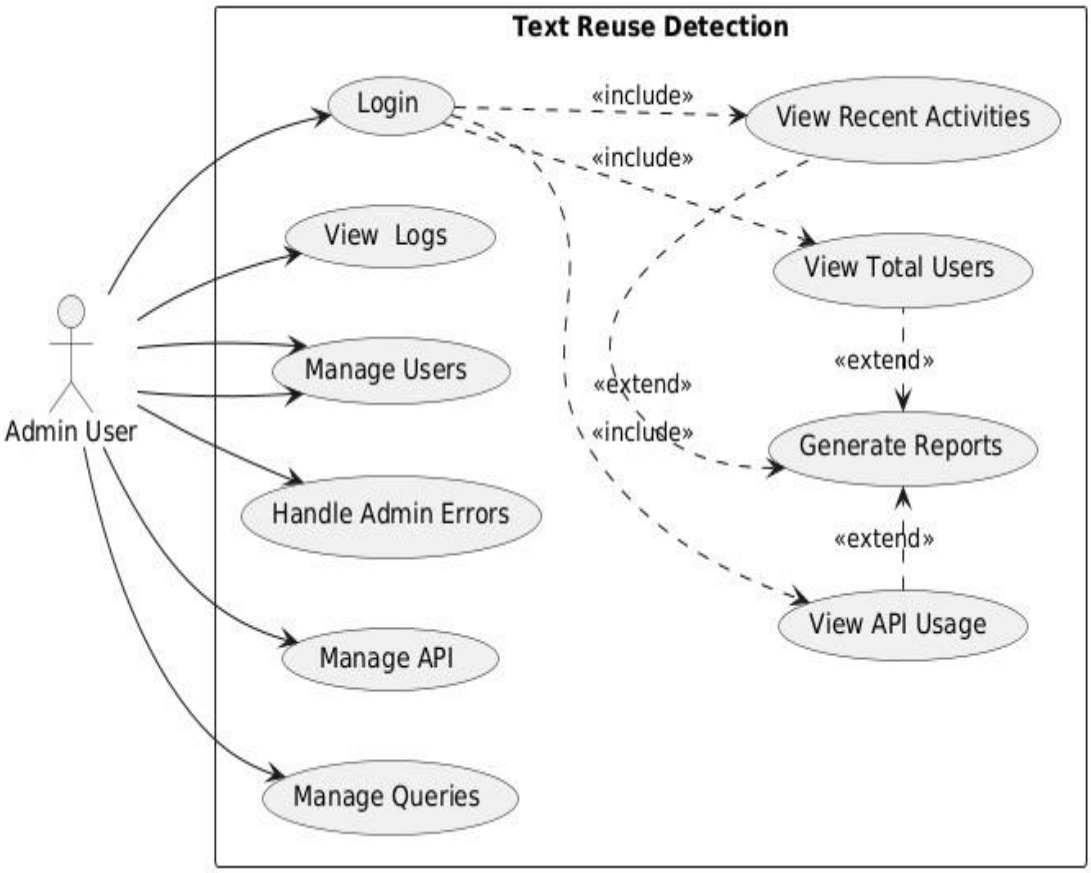


Figure 20 Admin Dashboard Use Case Diagram

2.6.18 System Management

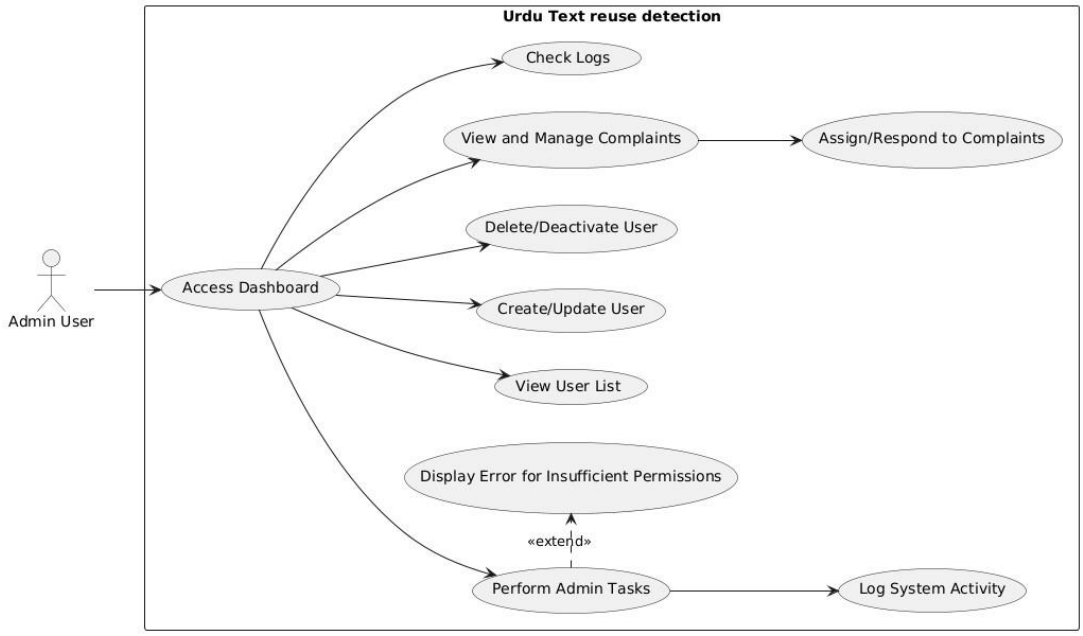


Figure 21 *System Management Use Case Diagram*

2.6.19 Report Generation

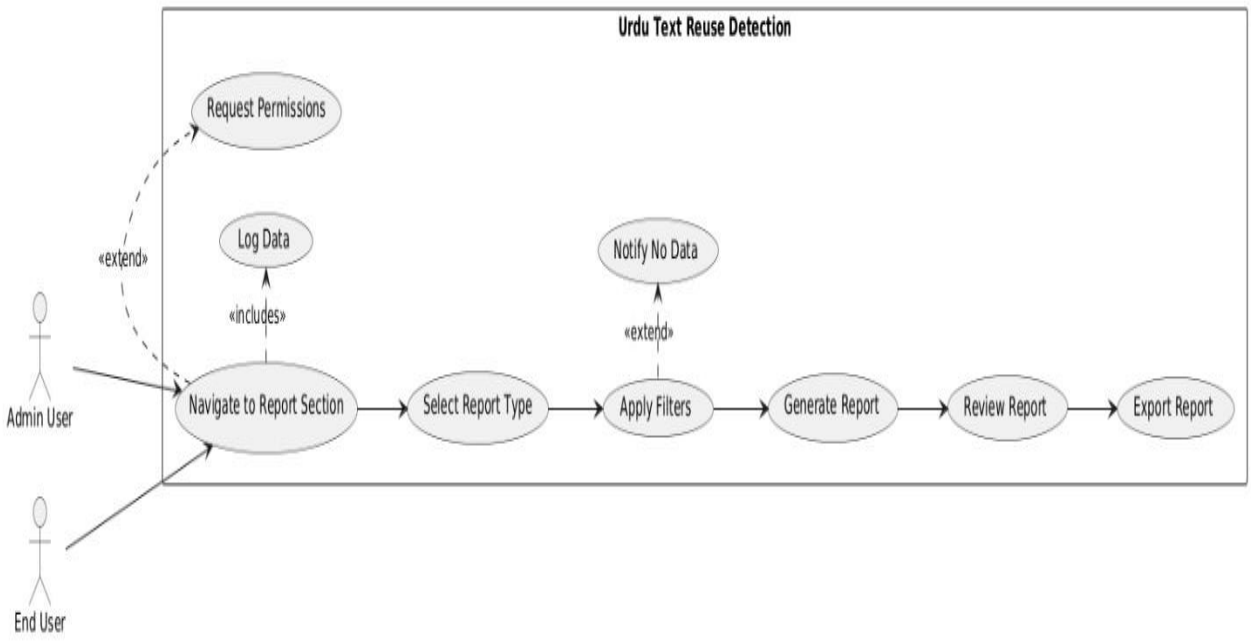


Figure 22 *Report Generation Use Case Diagram*

2.6.20 Analytics

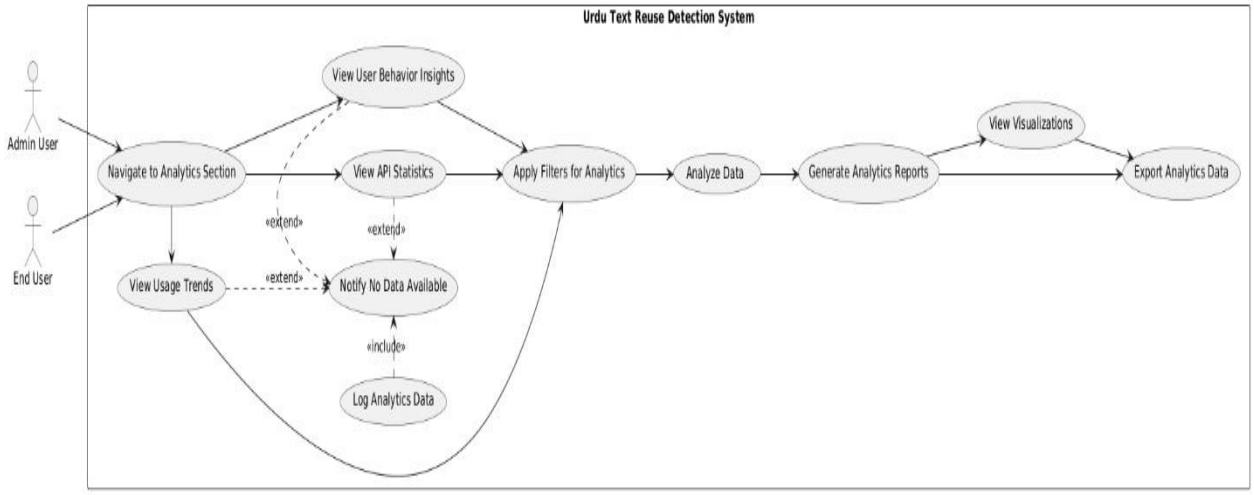


Figure 23 *Analytics Use Case Diagram*

2.7 Software development life cycle model

We Choose Agile SDLC model for this project because of its flexibility, adaptability, and iterative process due to the experimental nature of Urdu text reuse detection system. Agile allows use for incremental development, enables us to refine our model and application features in manageable sprints. Additionally, Agile is particularly suitable for AI and machine learning related projects, because it enables experimentation with various algorithms and architectures. Unlike traditional SDLC models like Waterfall, Agile gives the flexibility to address the challenges of text reuse detection's experimentation on of low-resource languages like Urdu, making it the suitable choice for delivering a user-friendly system.

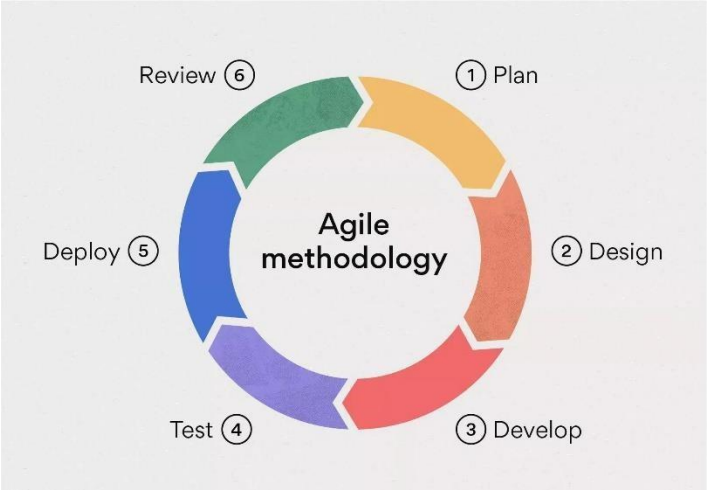


Figure 24 *Agile Diagram*

Chapter # 03

System Design

3 System Design

3.1 Work breakdown structure (WBS)

The following is the Work break down structure of this final year project:

Figure 25 Work Break Down Structure

3.2 Activity diagram

The following are the activity diagram of each user functional requirement of our final year project:

3.2.1 Login

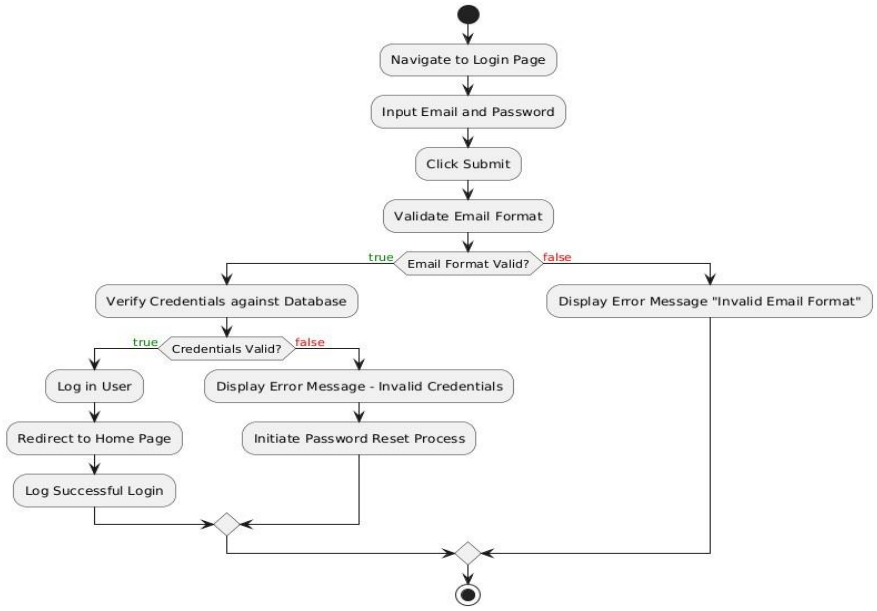


Figure 26 *Login Activity Diagram*

3.2.2 Create Account

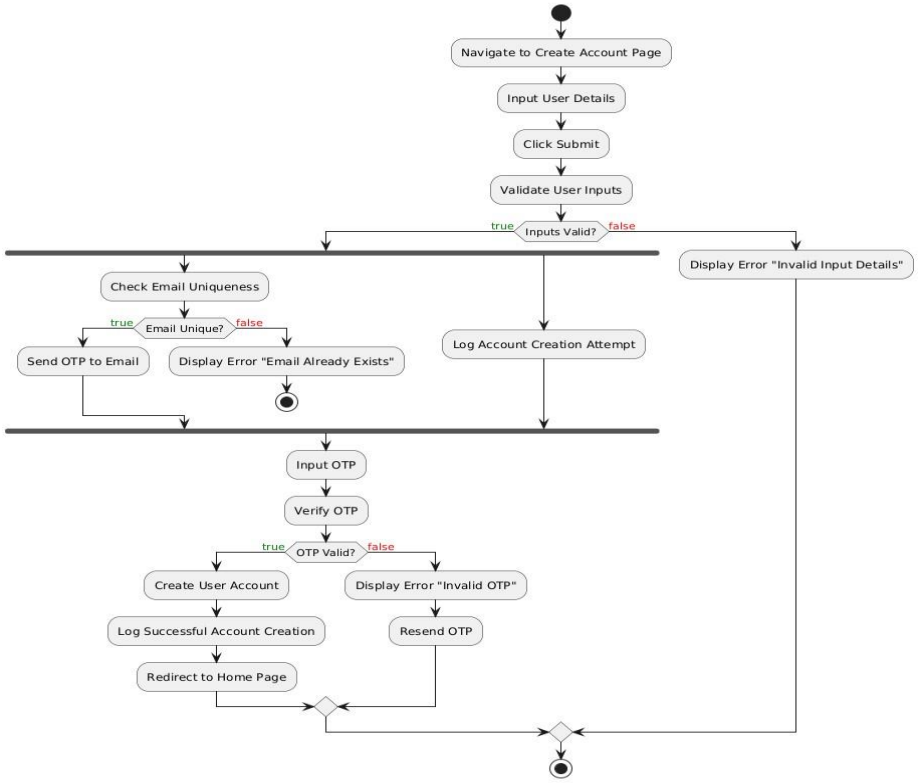


Figure 27 *Create Account Activity Diagram*

3.2.3 Forgot Password

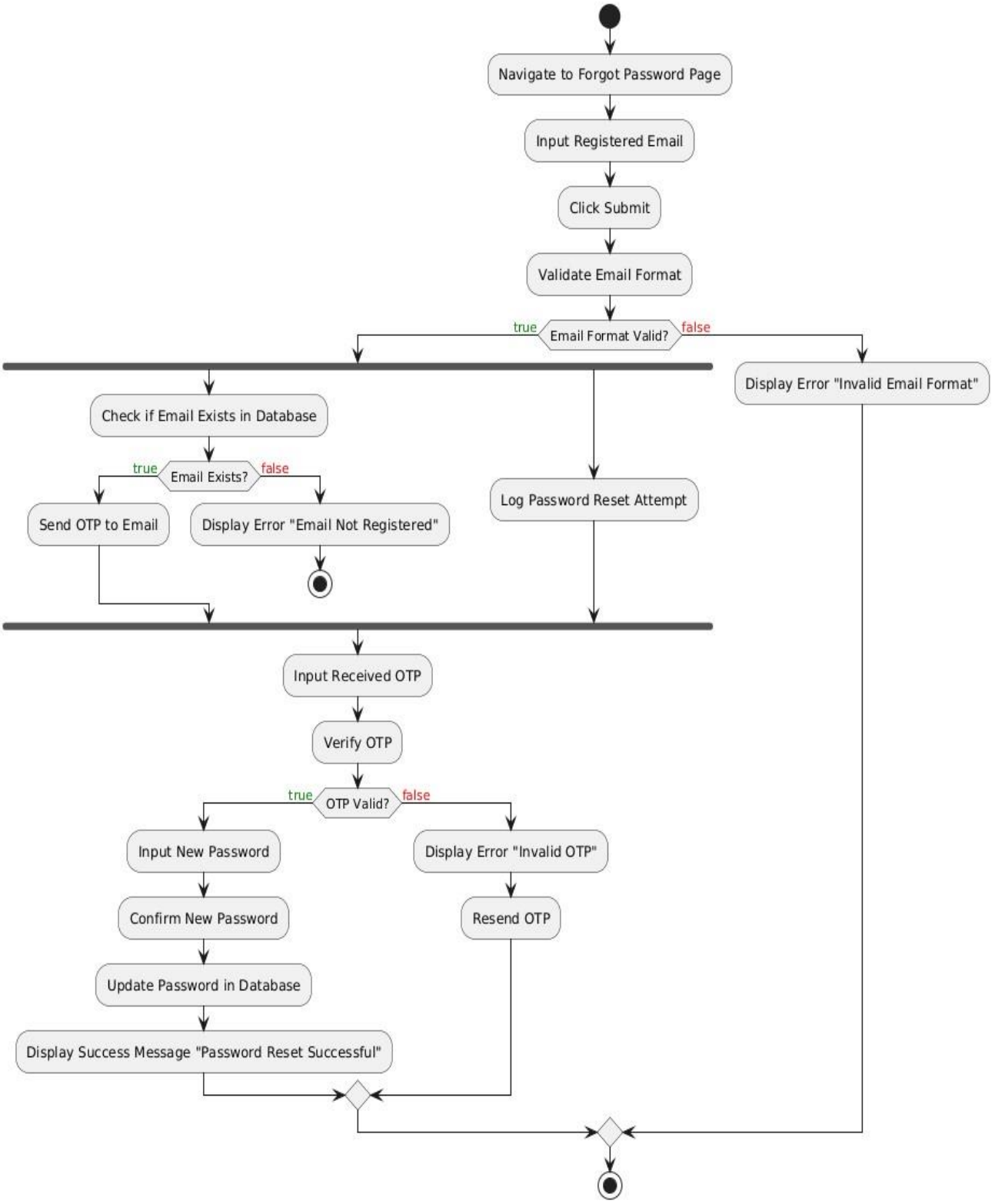


Figure 28 *Forgot Password Activity Diagram*

3.2.4 Change Password

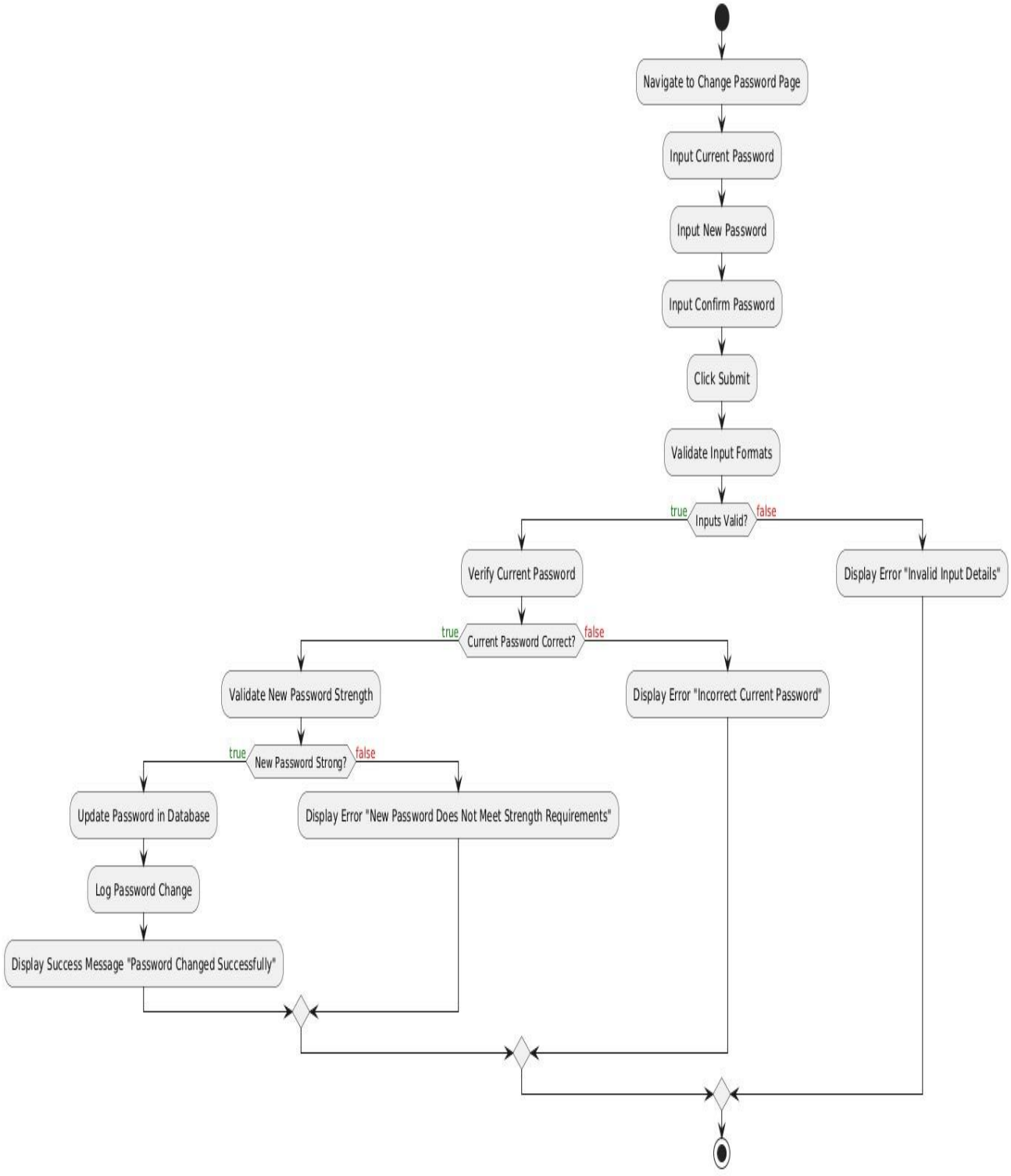


Figure 29 *Change Password Activity Diagram*

3.2.5 Edit Profile

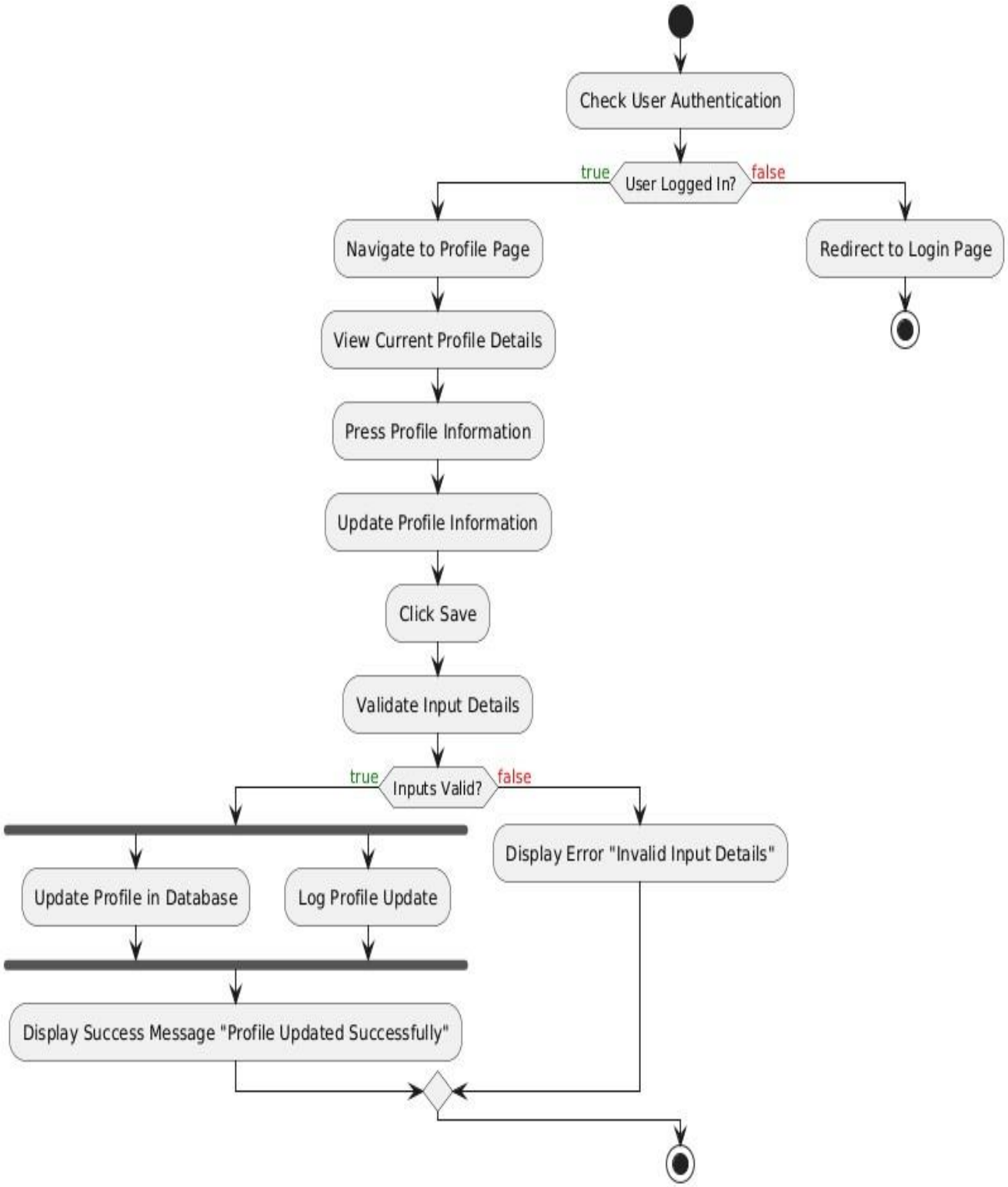


Figure 30 *Edit Profile Activity Diagram*

3.2.6 API services

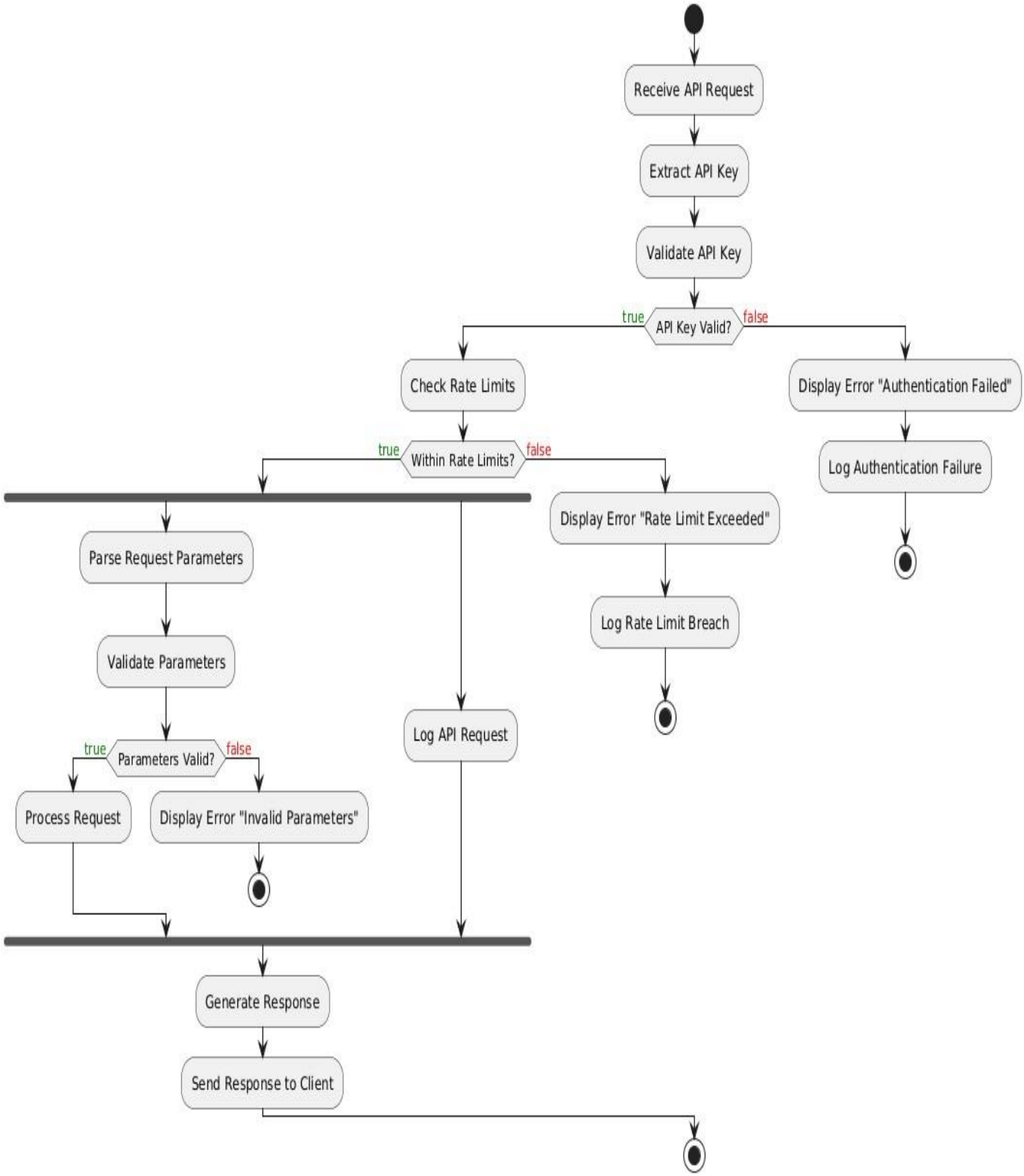


Figure 31 *API Services Activity Diagram*

3.2.7 Rate Limiting for API

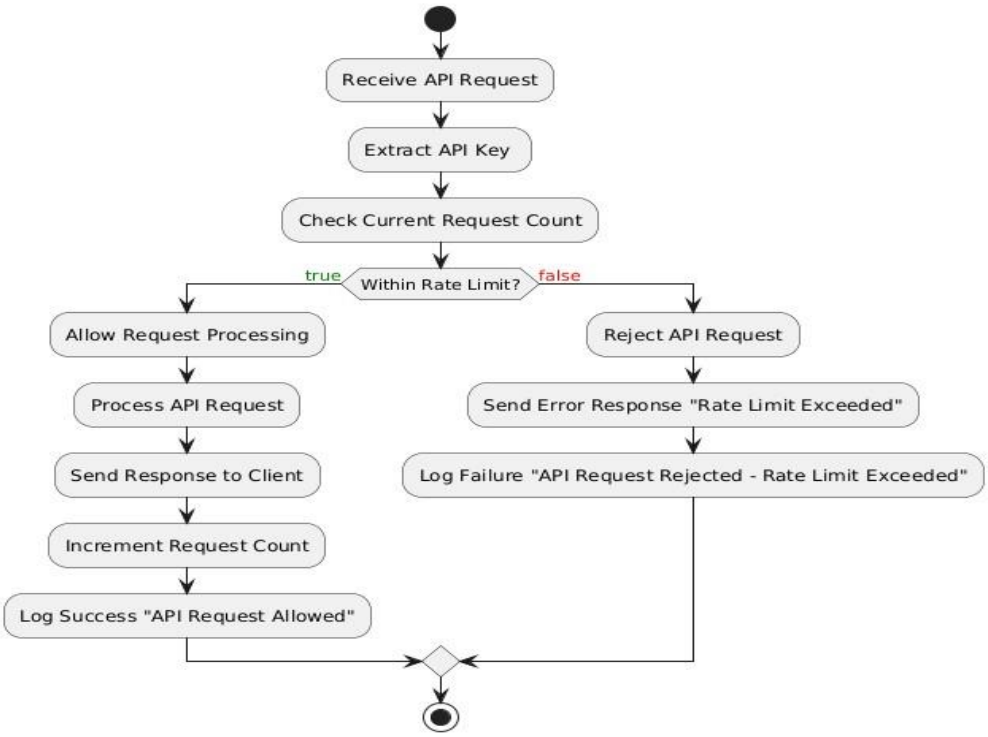


Figure 32 *API Rate Limiting Activity Diagram*

3.2.8 Authorization for API

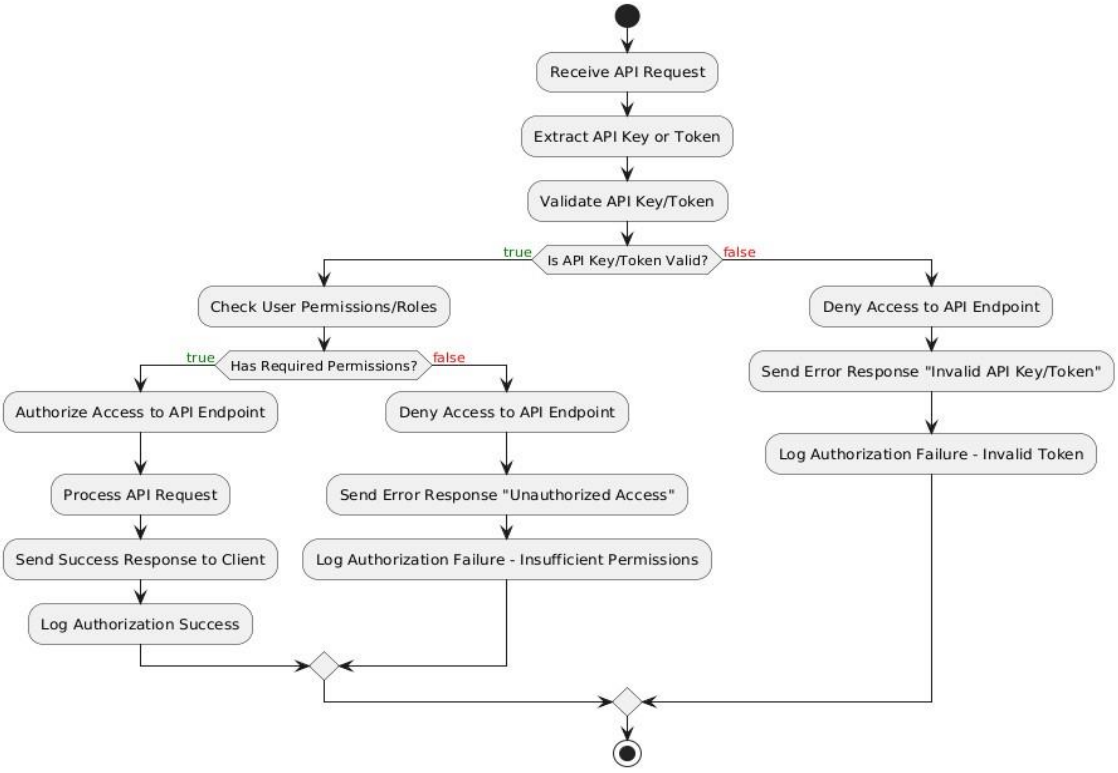


Figure 33 *API Authorization Activity Diagram*

3.2.9 API logging

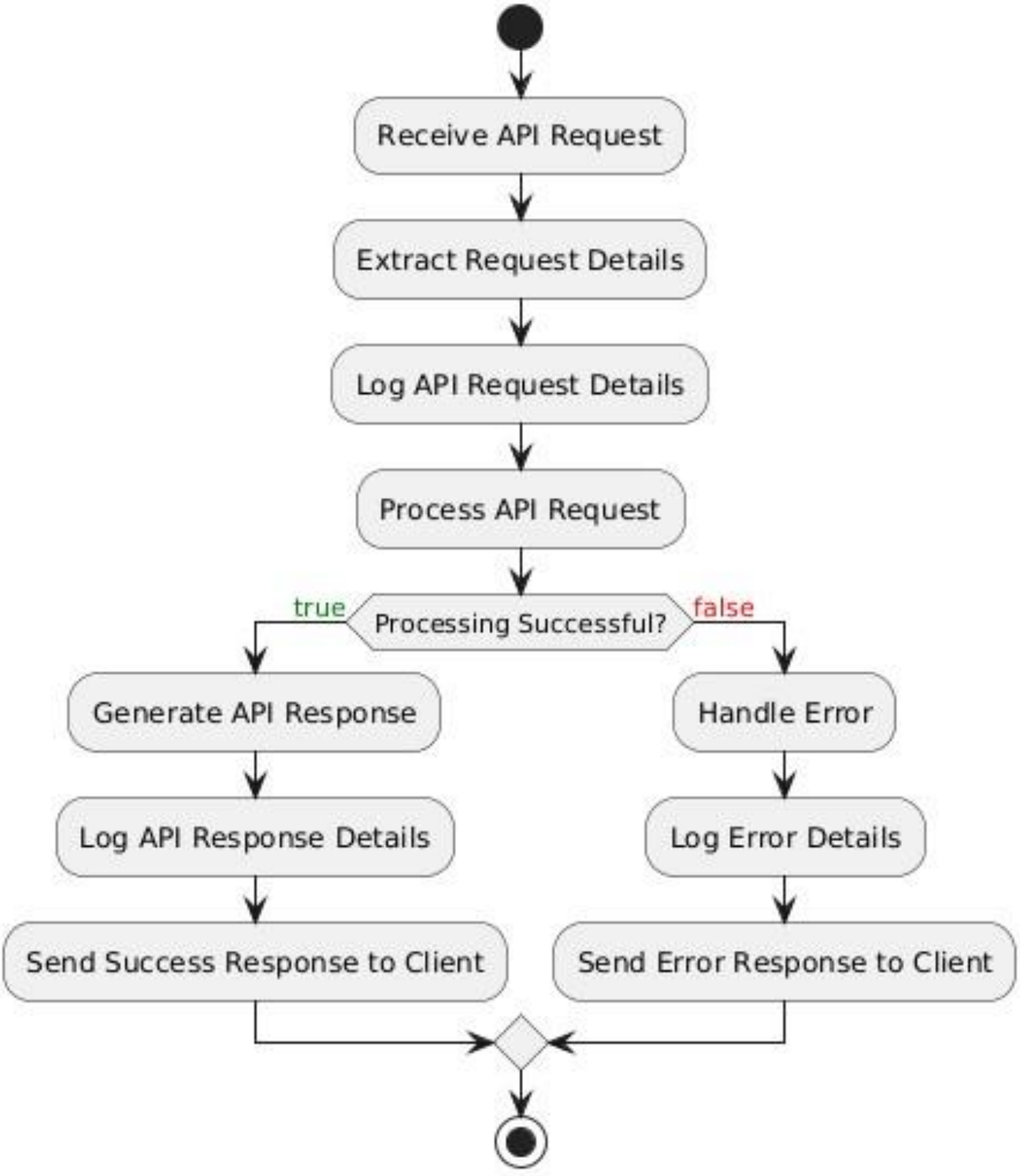


Figure 34 *API logging Activity Diagram*

3.2.10 User History

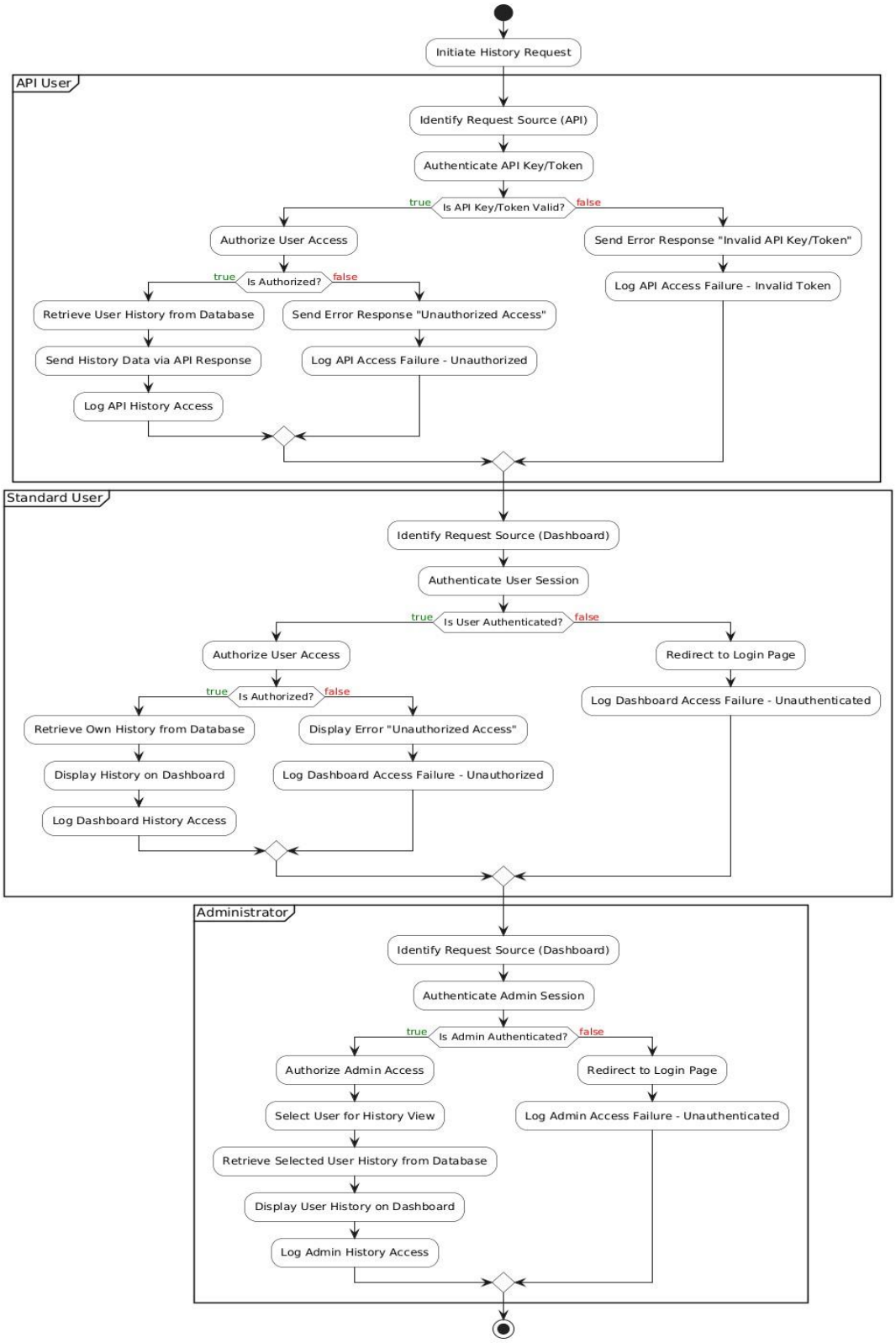


Figure 35 *User history Activity Diagram*

3.2.11 API Key management

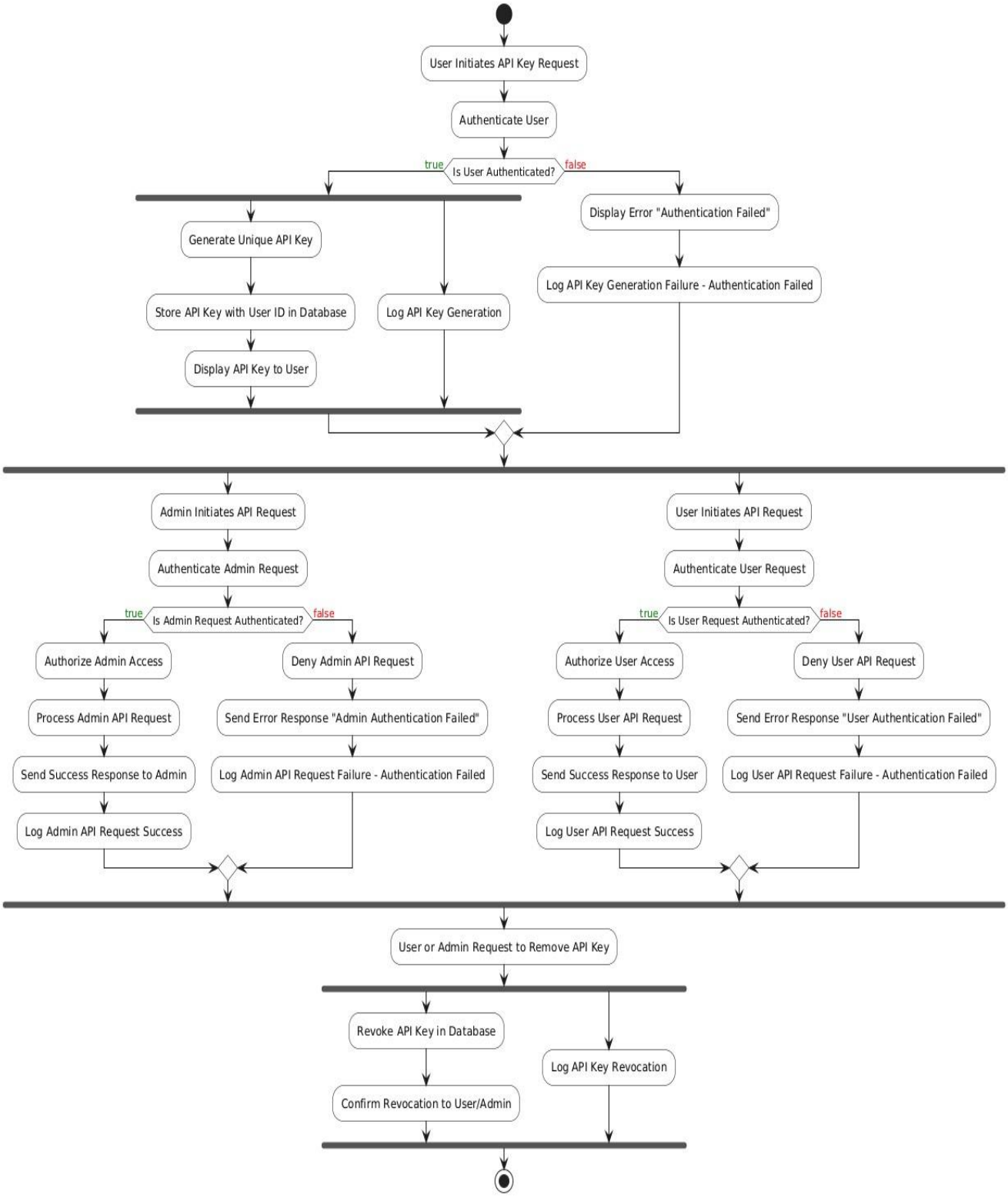


Figure 36 *API key Management Activity Diagram*

3.2.12 API monitoring

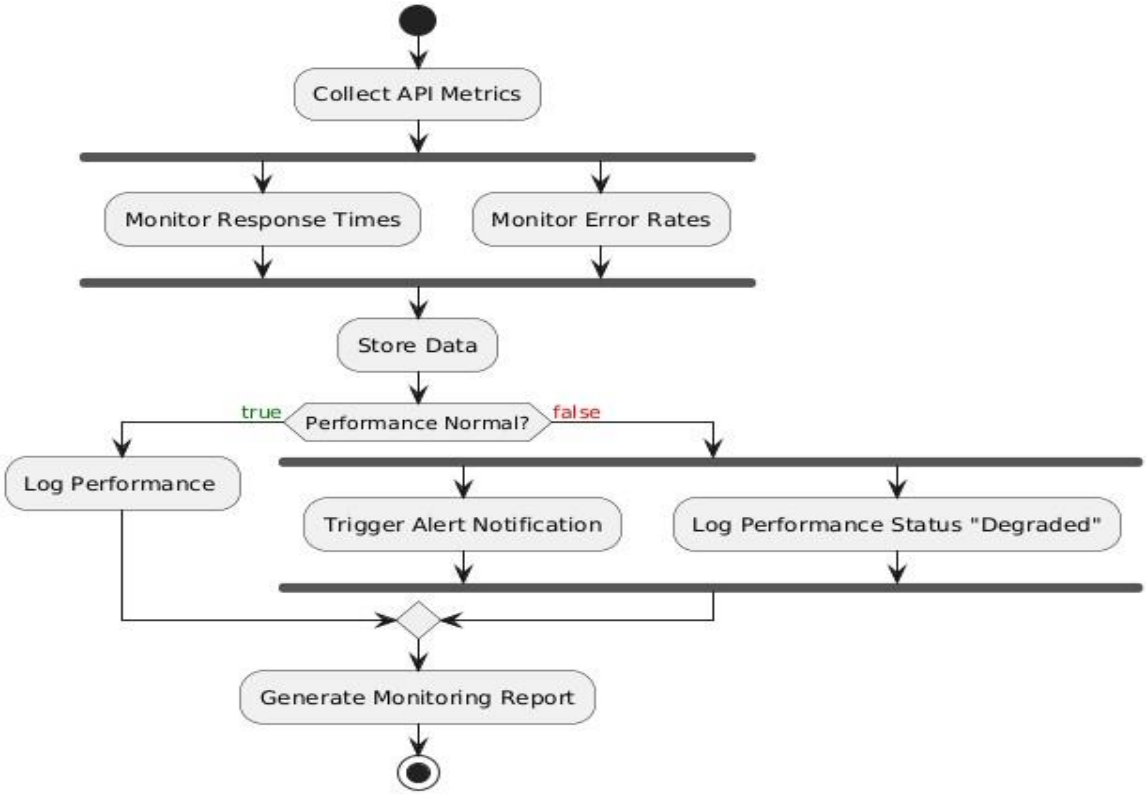


Figure 37 *API monitoring Activity Diagram*

3.2.13 Text Reuse Detection

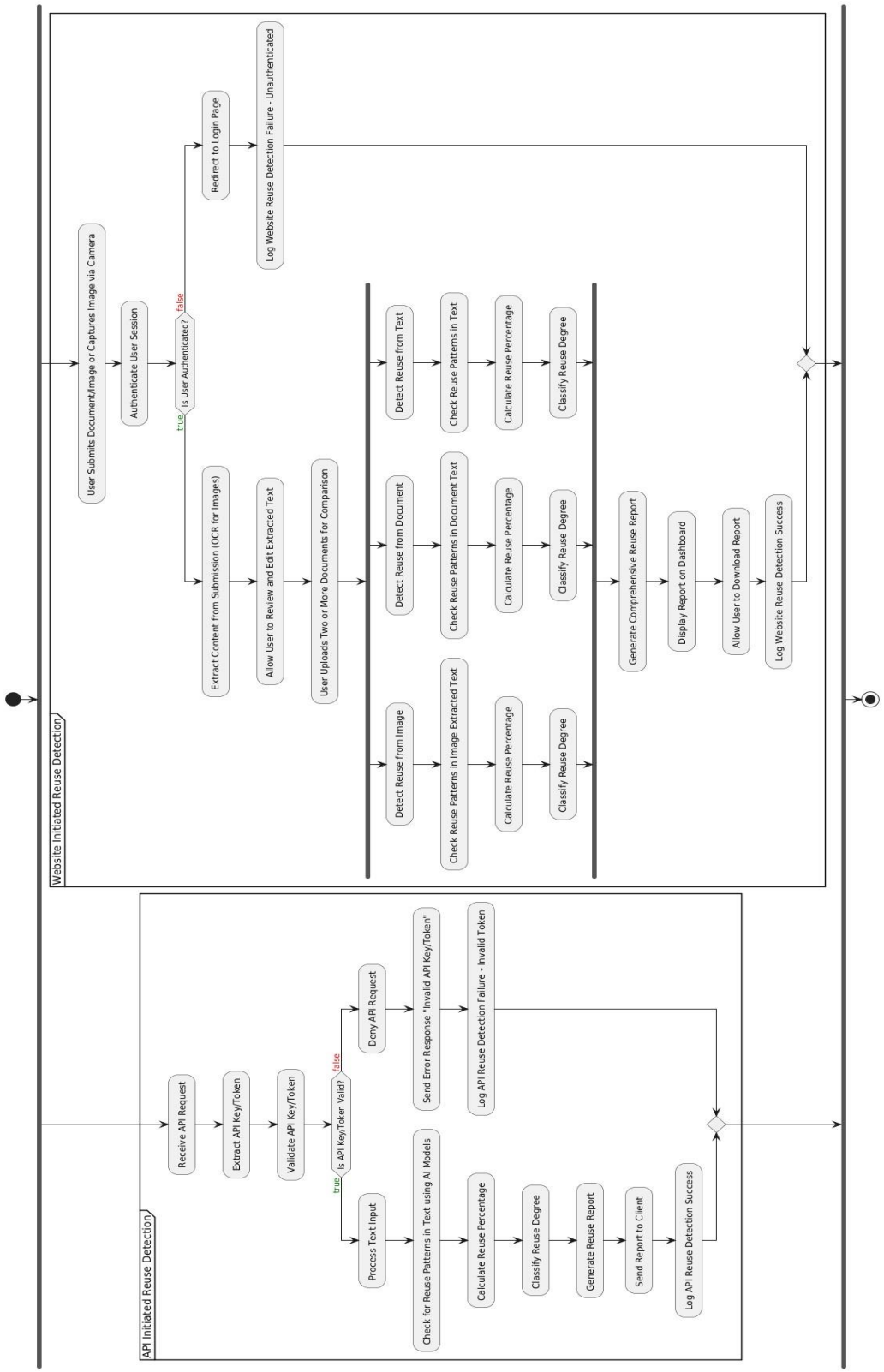


Figure 38 *Text Reuse Detection Activity Diagram*

3.2.14 Plagiarism Detection

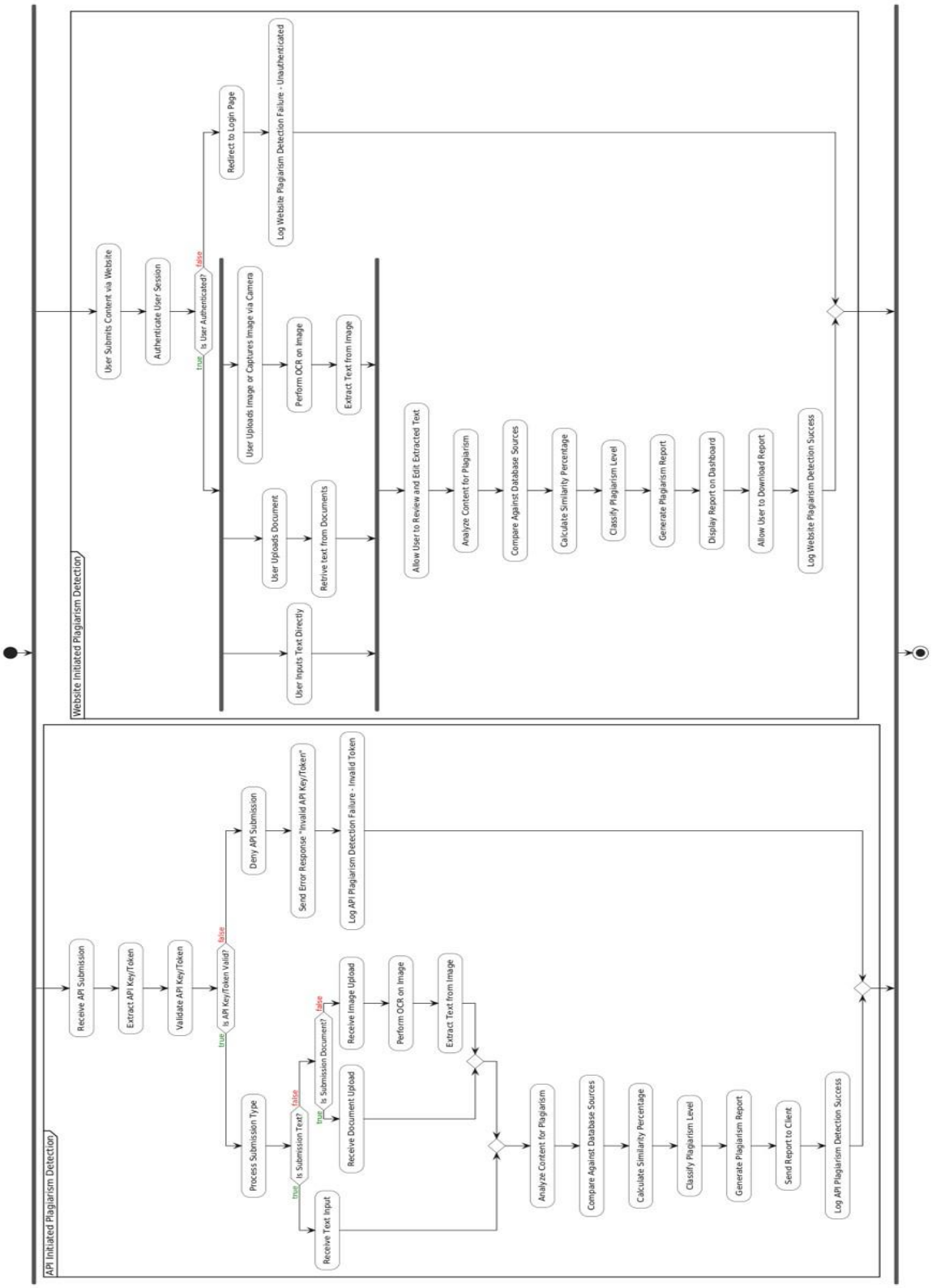


Figure 39 *Plagiarism Detection Activity Diagram*

3.2.15 Contact

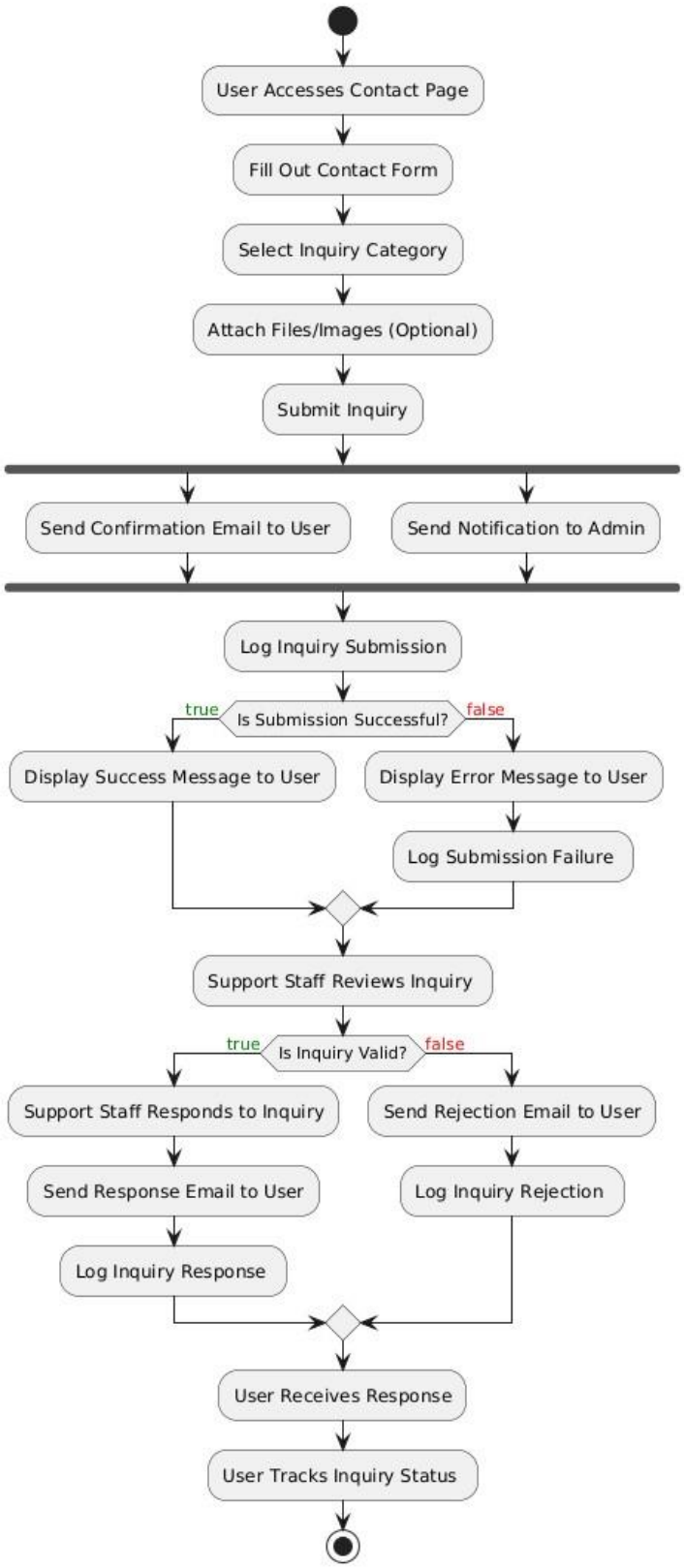


Figure 40 *Contact Activity Diagram*

3.2.16 Notification

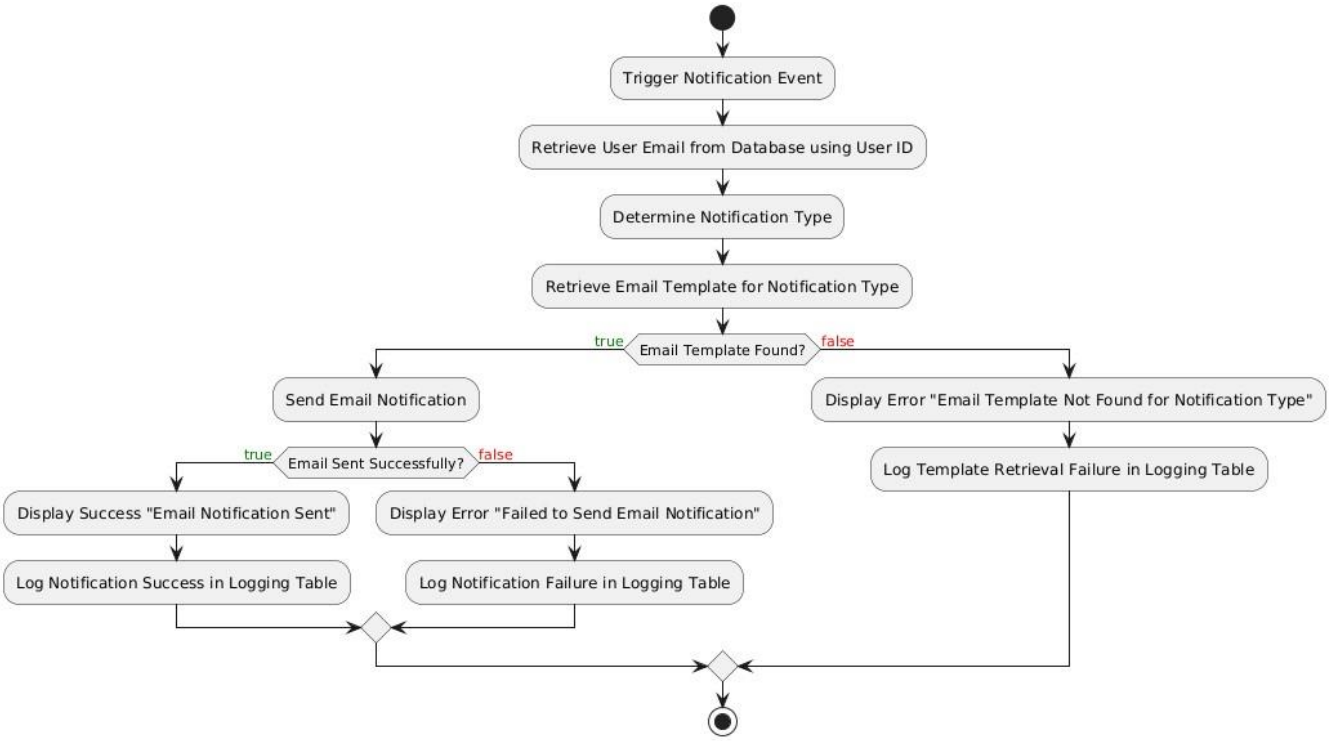


Figure 41 *Notifications Activity Diagram*

3.2.17 Admin Dashboard

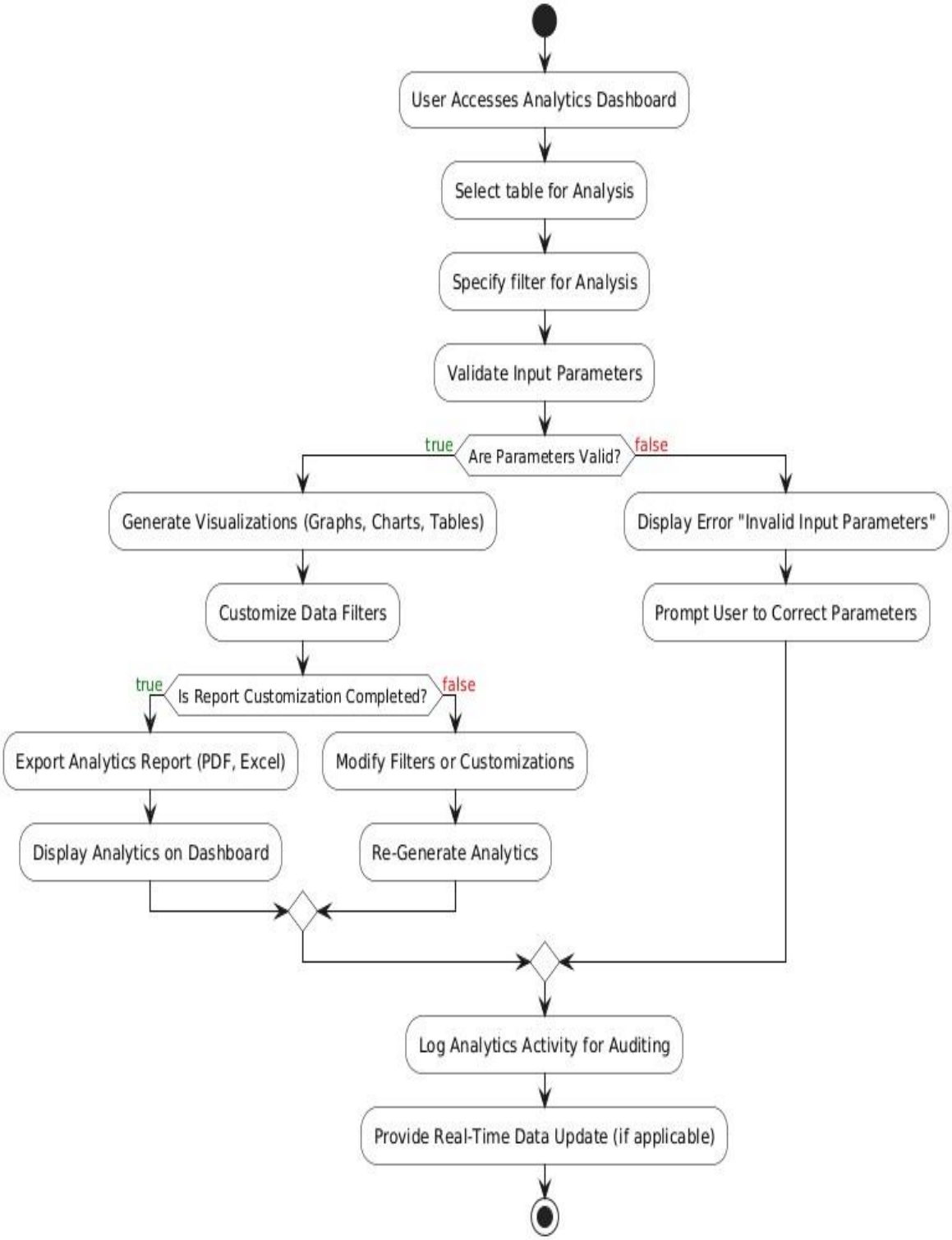


Figure 42 *Admin Dashboard Activity Diagram*

3.2.18 System Management

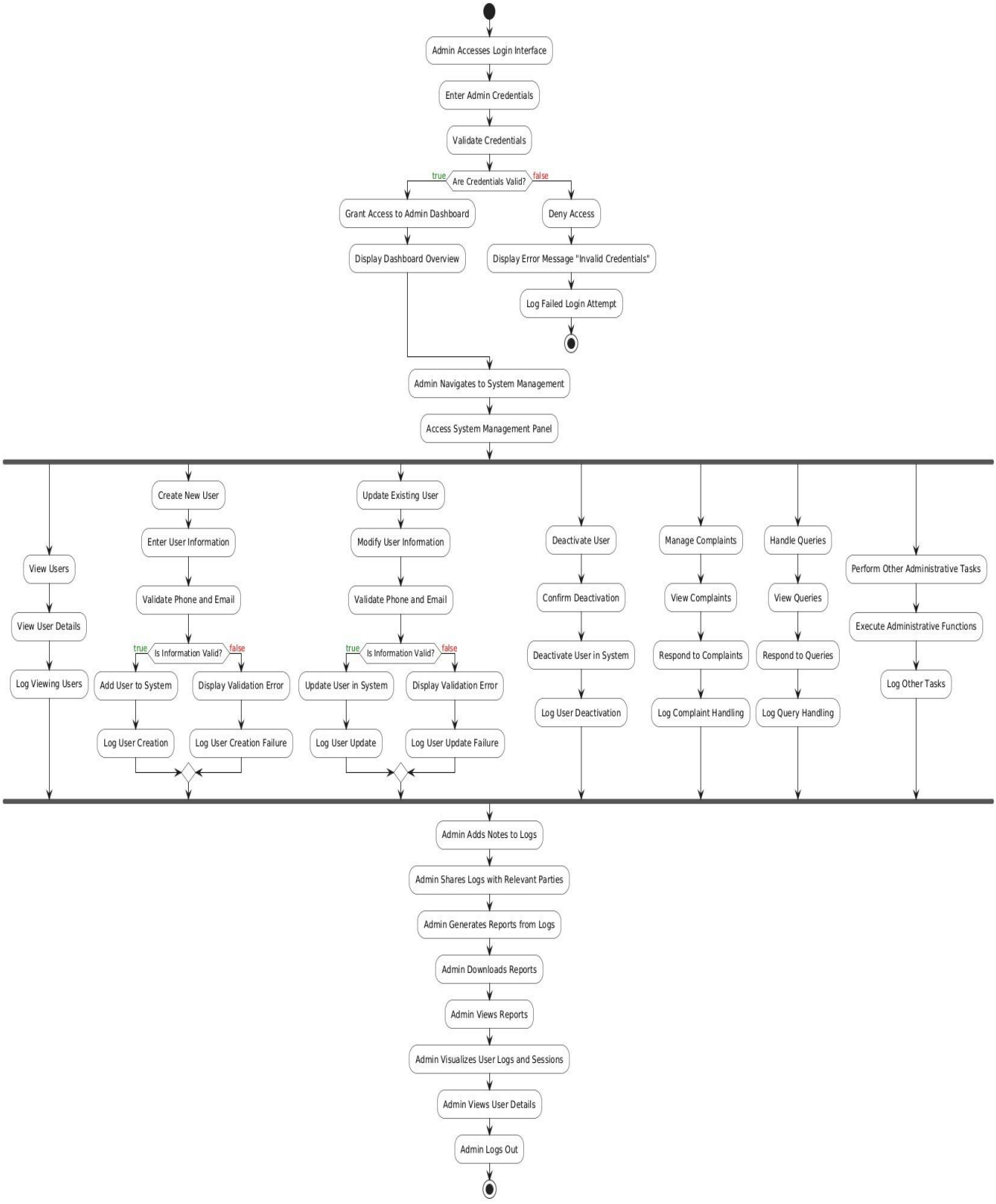


Figure 43 *System Management Activity Diagram*

3.2.19 Report Generation

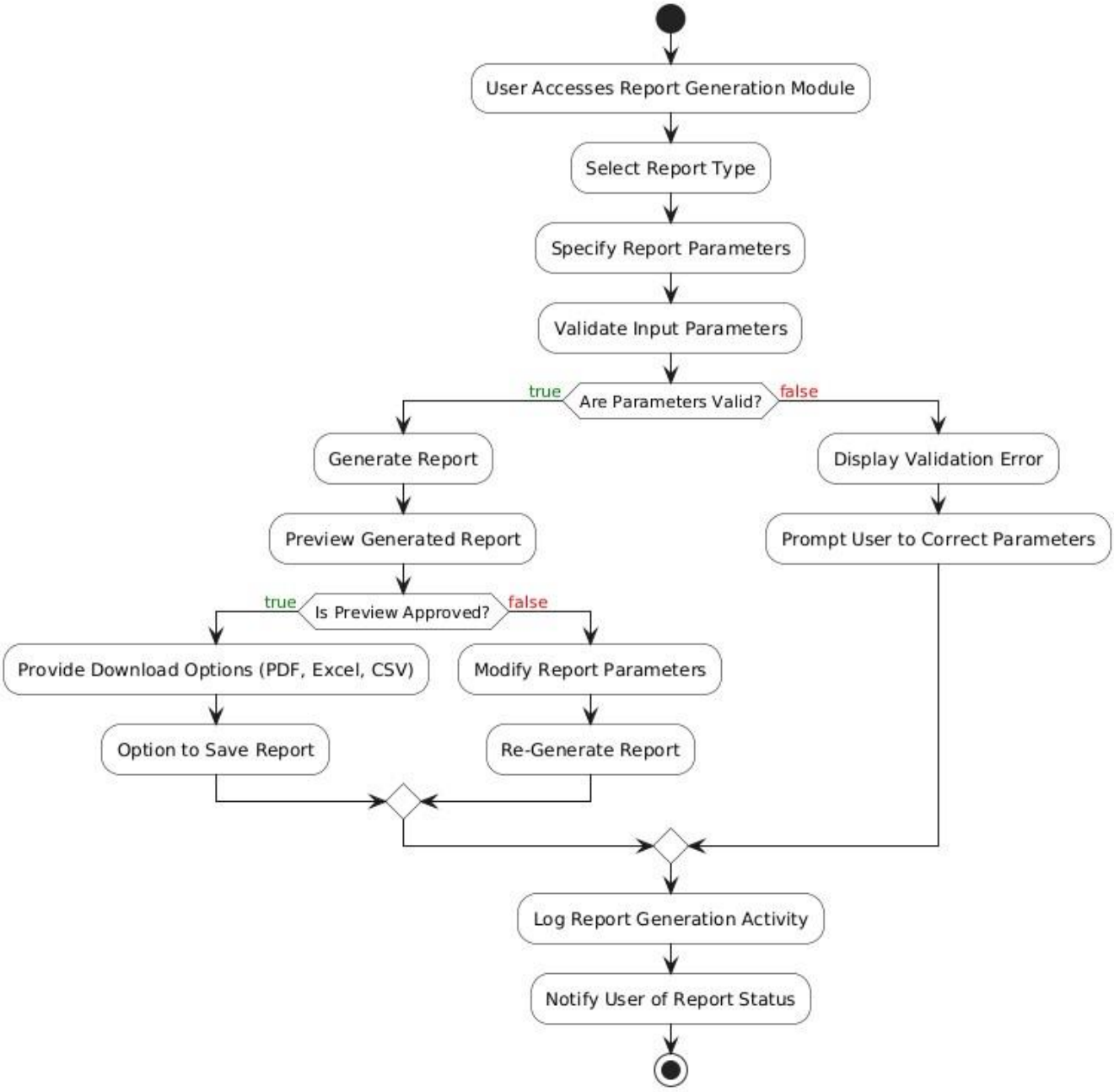


Figure 44 *Report Generation Activity Diagram*

3.2.20 Analytics

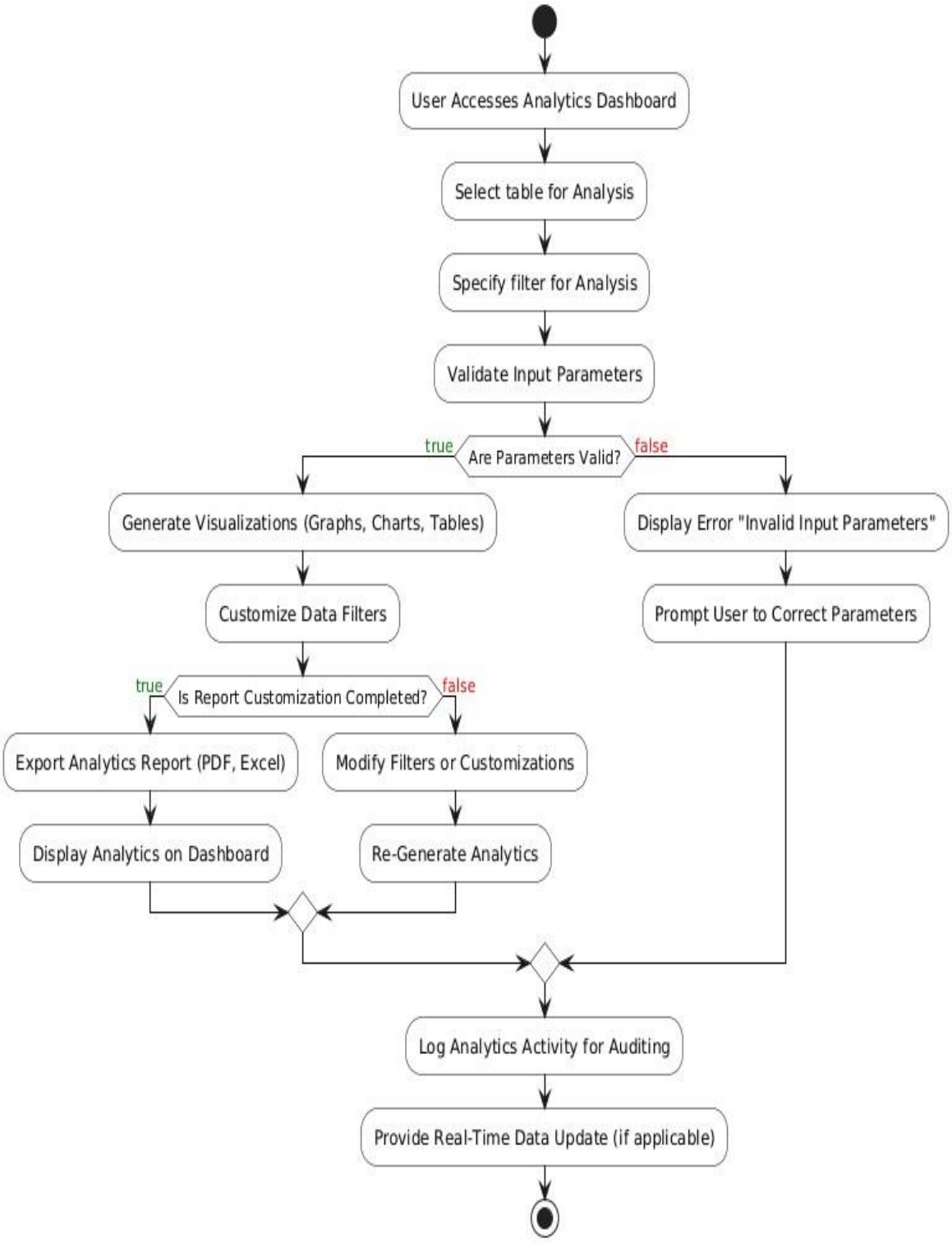


Figure 45 *Analytics Activity Diagram*

3.3 Sequence diagram

3.3.1 Login

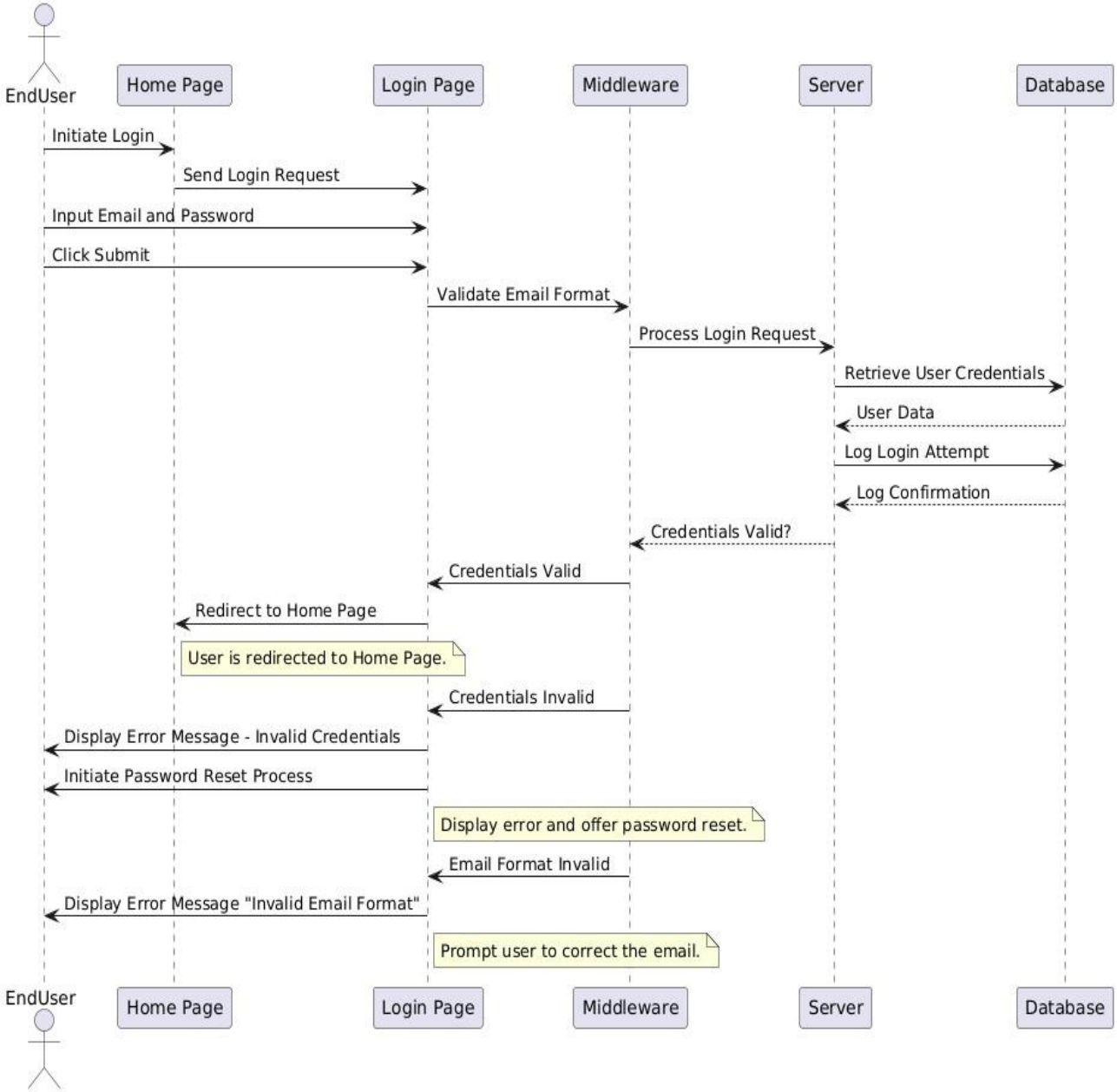


Figure 46 *Login Sequence Diagram*

3.3.2 Create Account

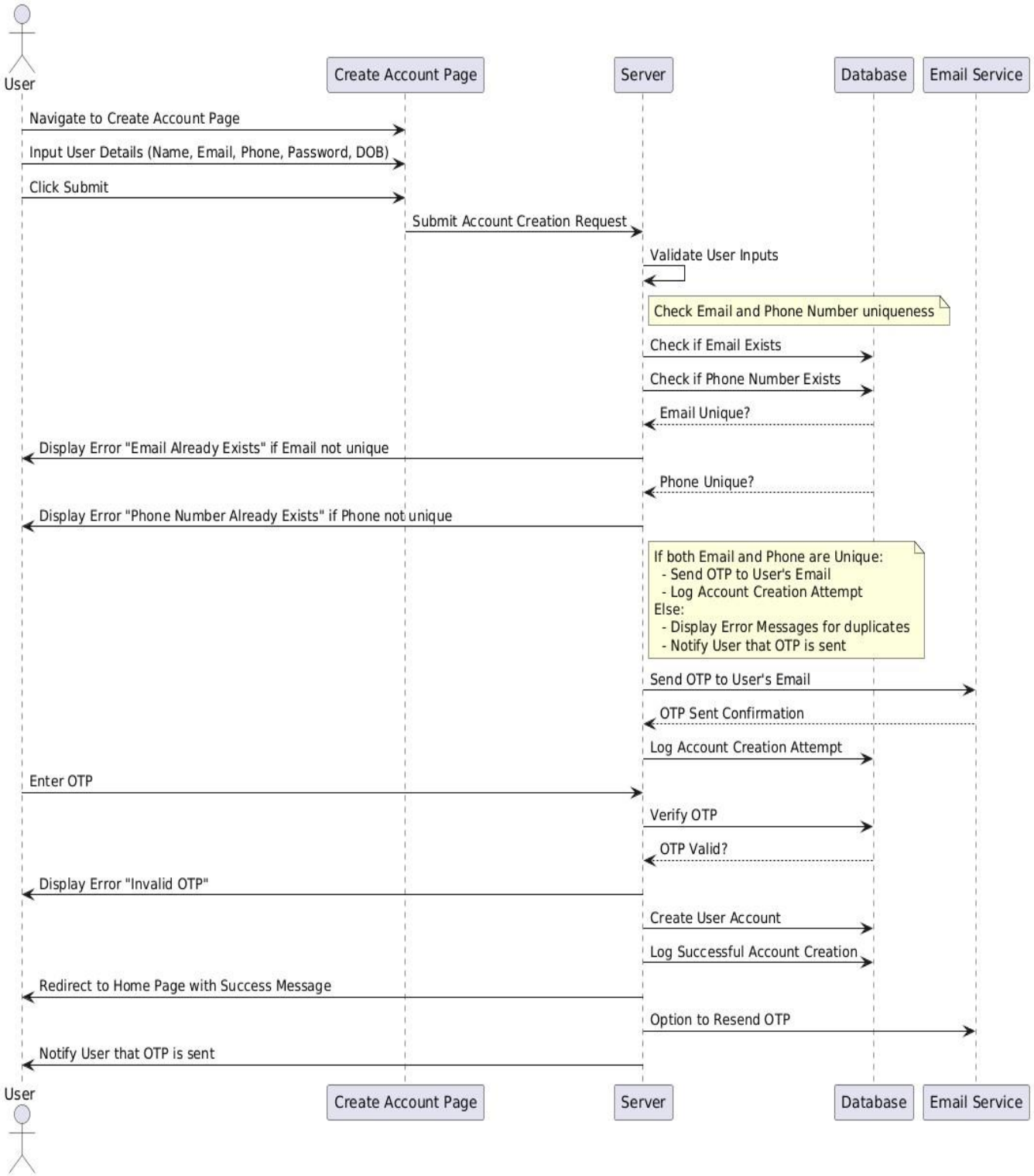


Figure 47 *Create Account Sequence Diagram*

3.3.3 Forgot Password

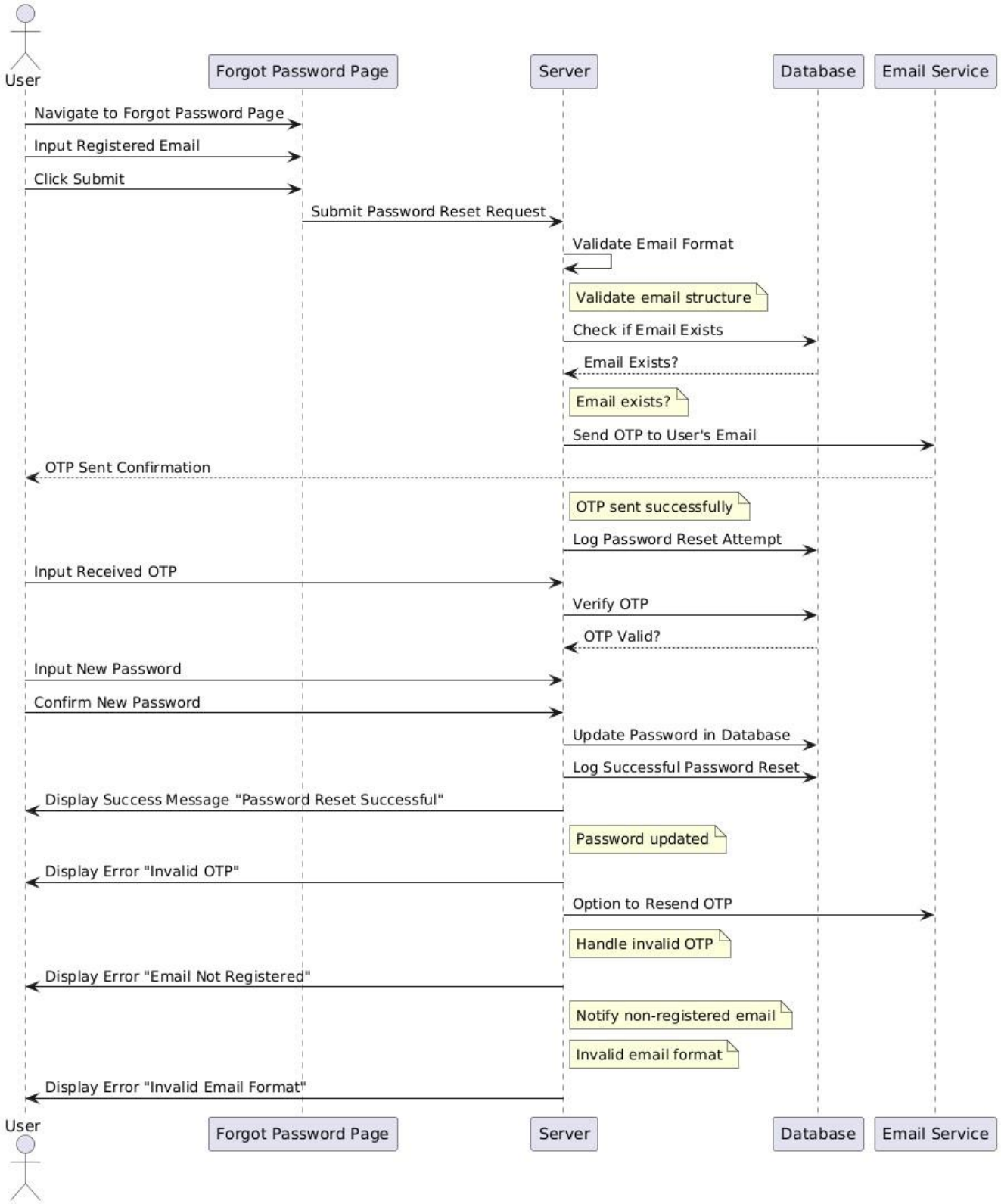


Figure 48 *Forgot Password Sequence Diagram*

3.3.4 Change Password

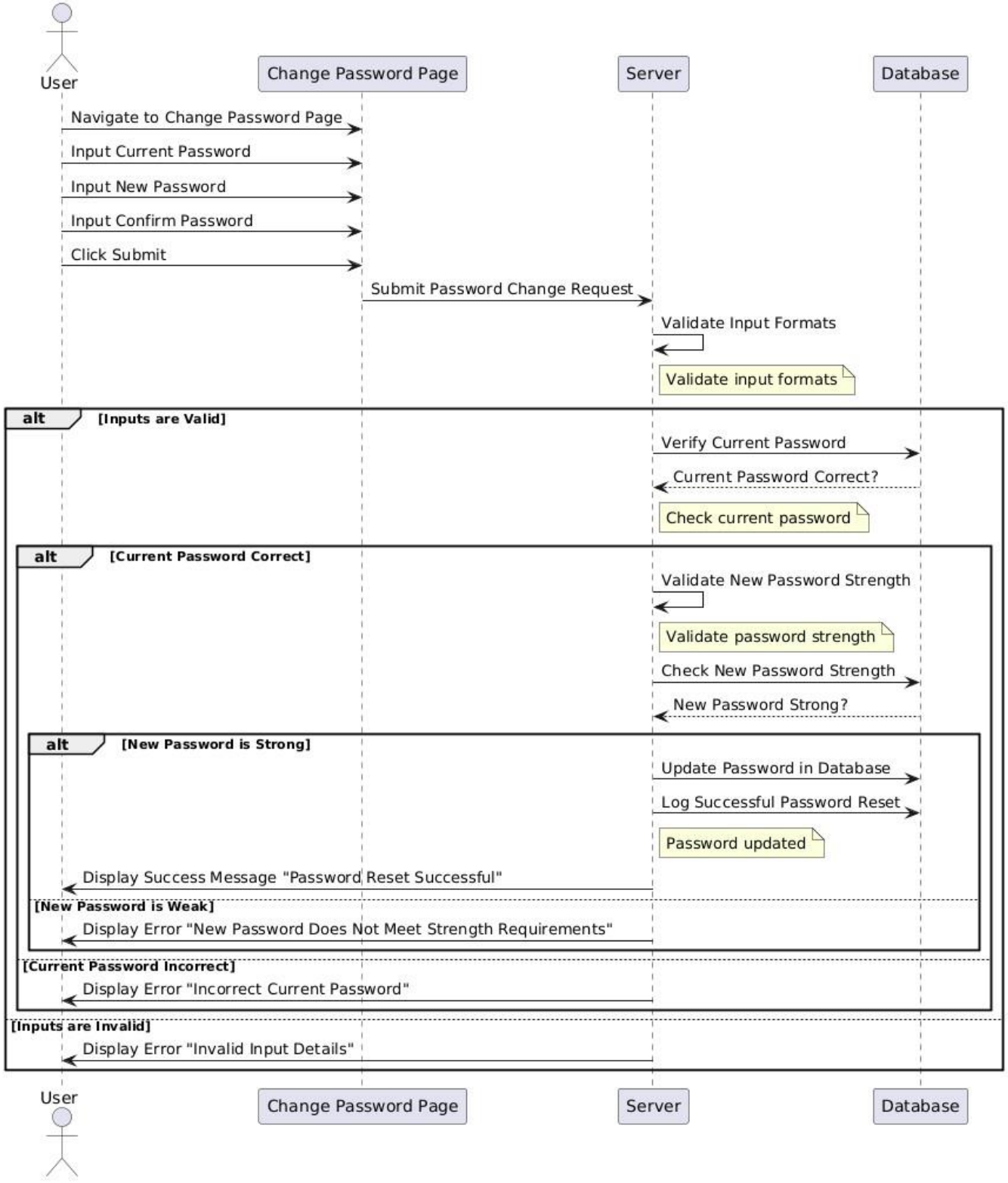


Figure 49 *Change Password Sequence Diagram*

2.1.1. Edit Profile

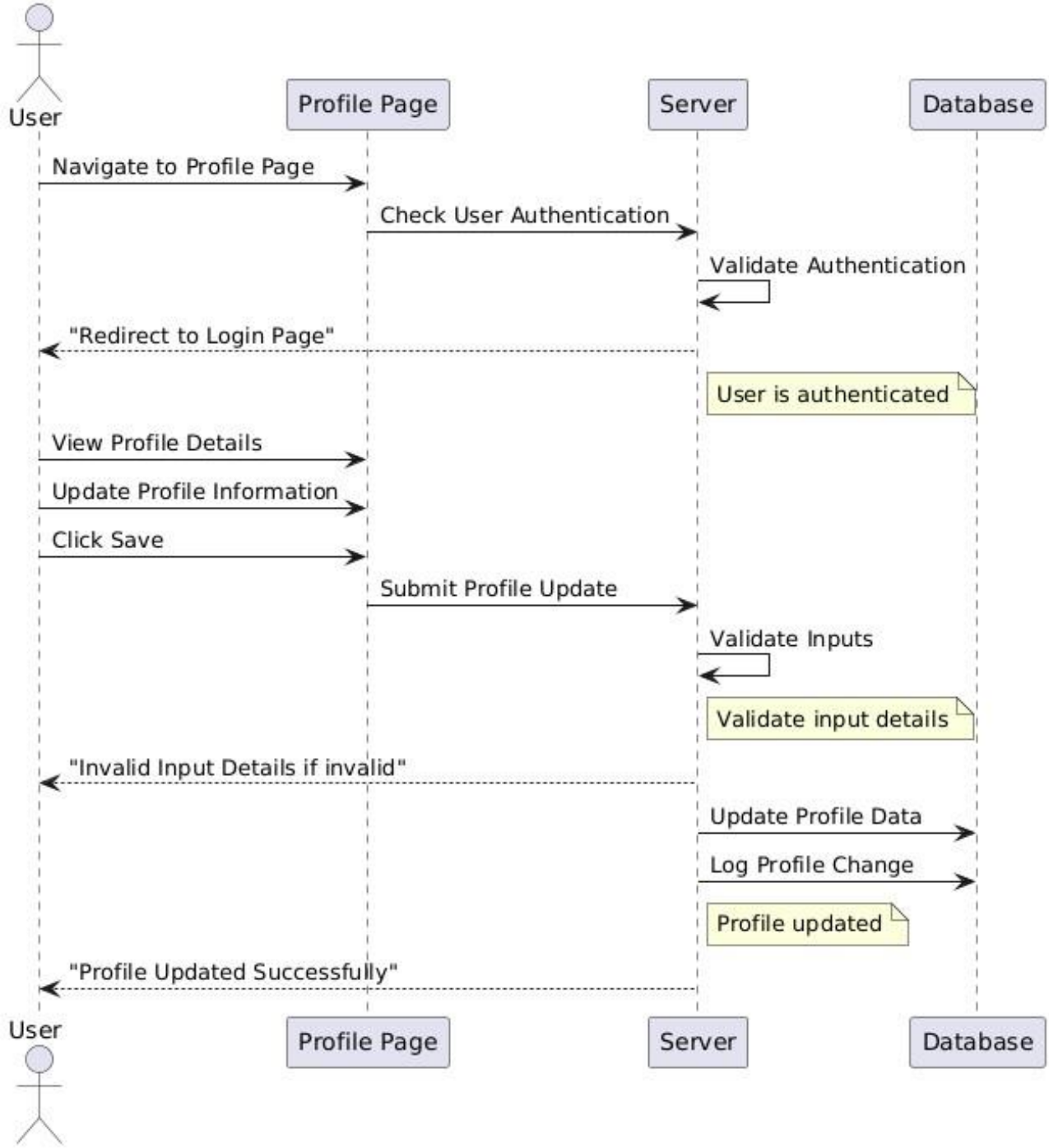


Figure 50 *Update Profile Sequence Diagram*

3.3.5 API services

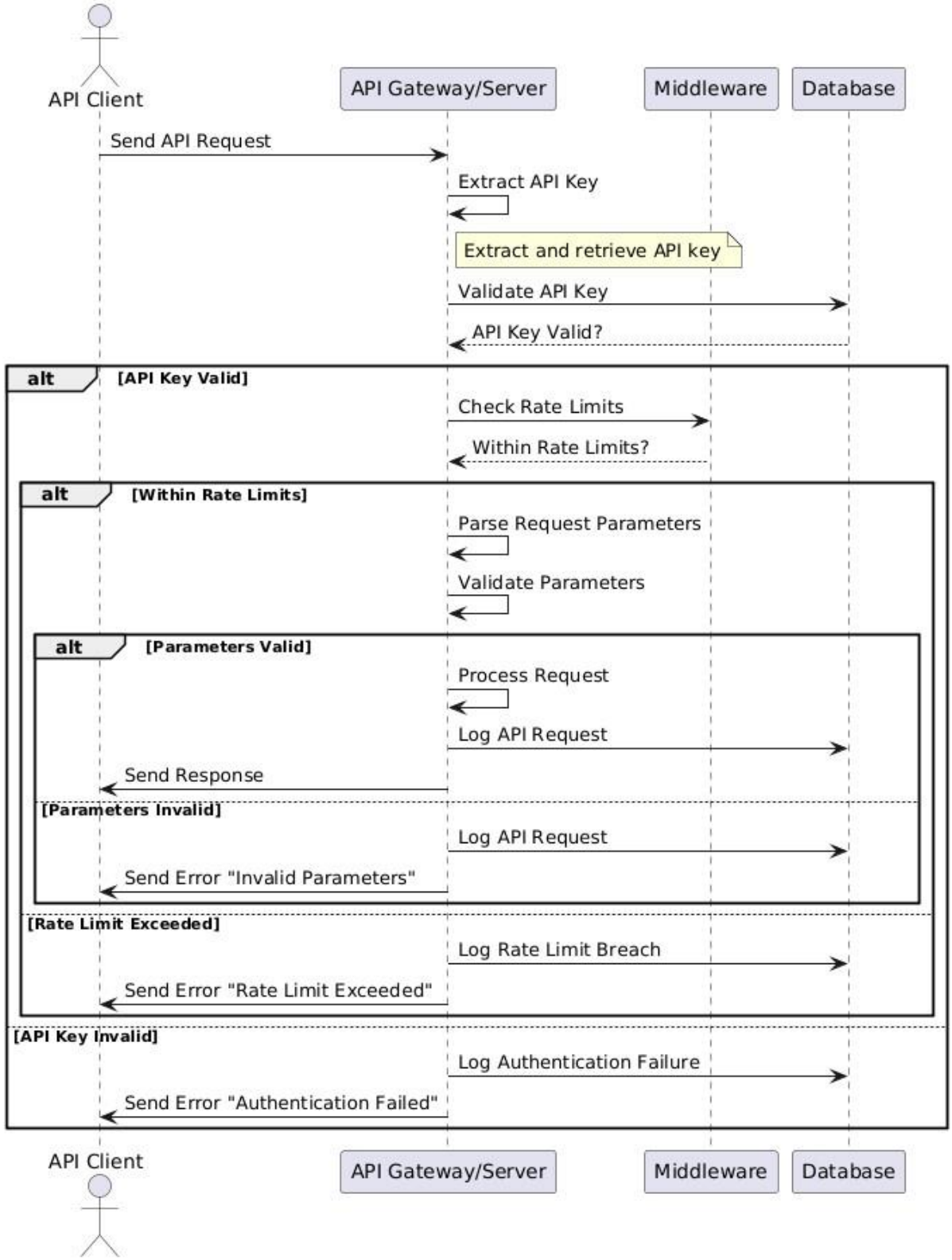


Figure 51 *API Services Sequence Diagram*

3.3.6 Rate Limiting for API

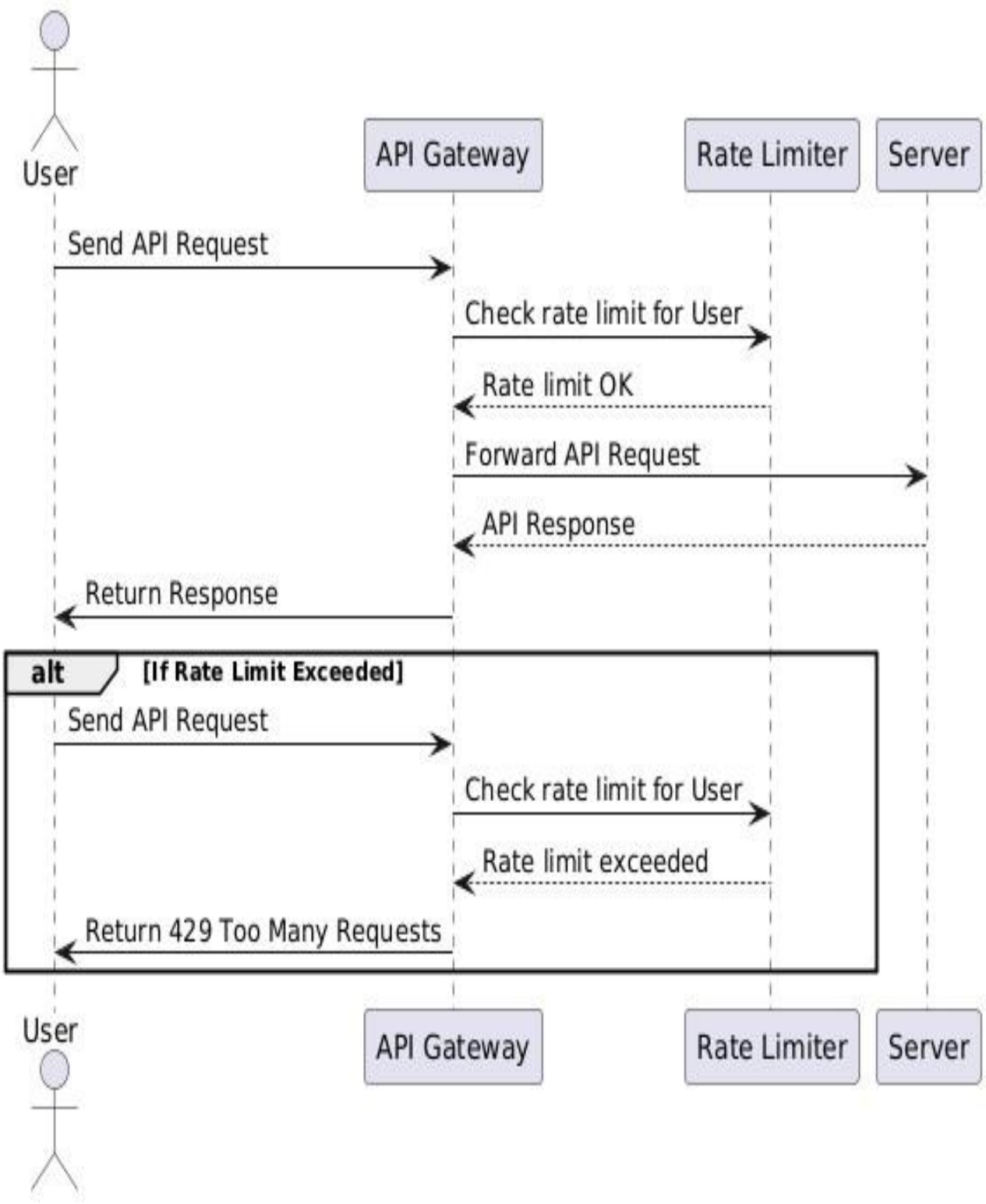


Figure 52 *Rate Limiting Sequence Diagram*

3.3.7 Authorization for API

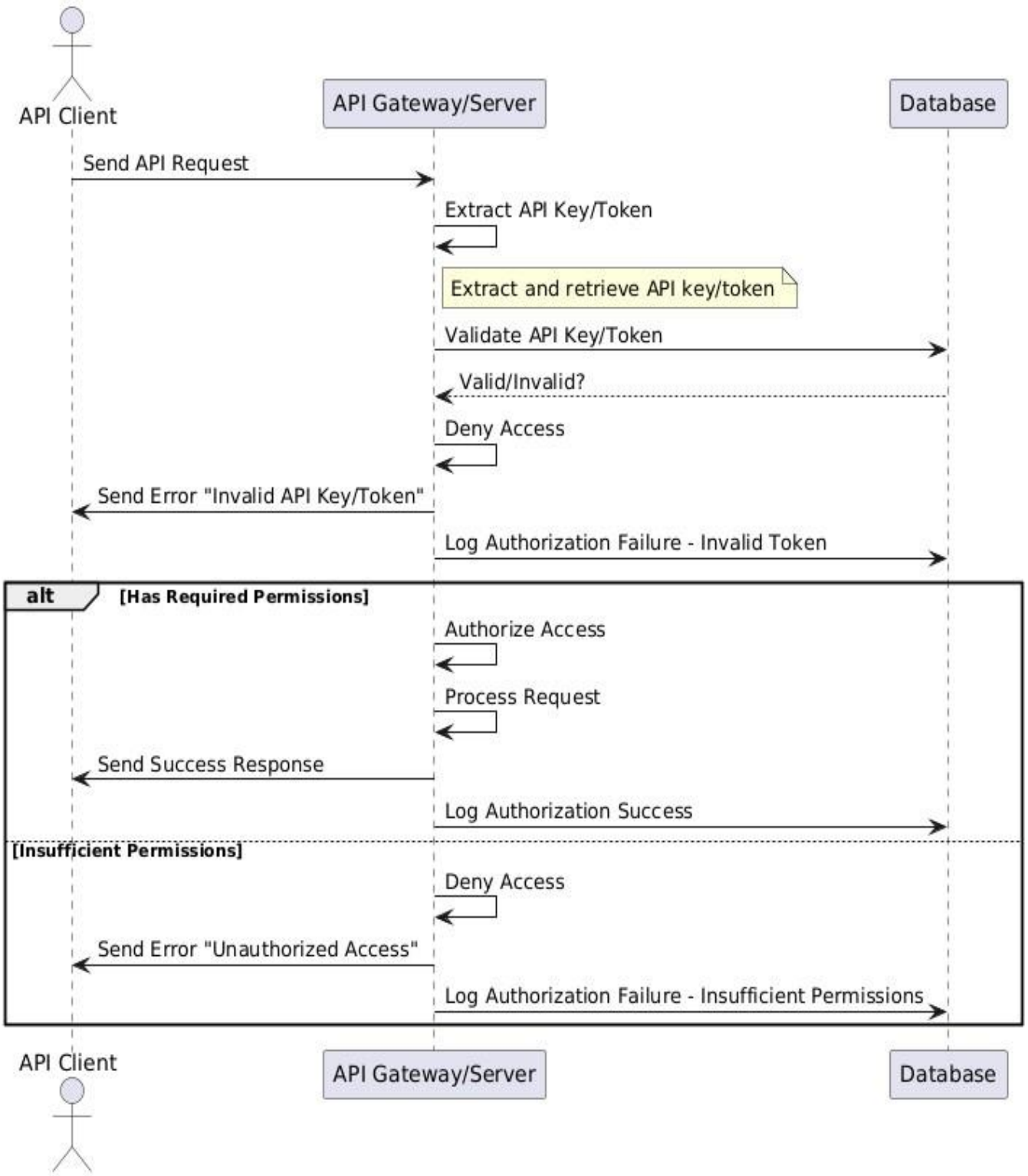


Figure 53 *Authorization Sequence Diagram*

3.3.8 API logging

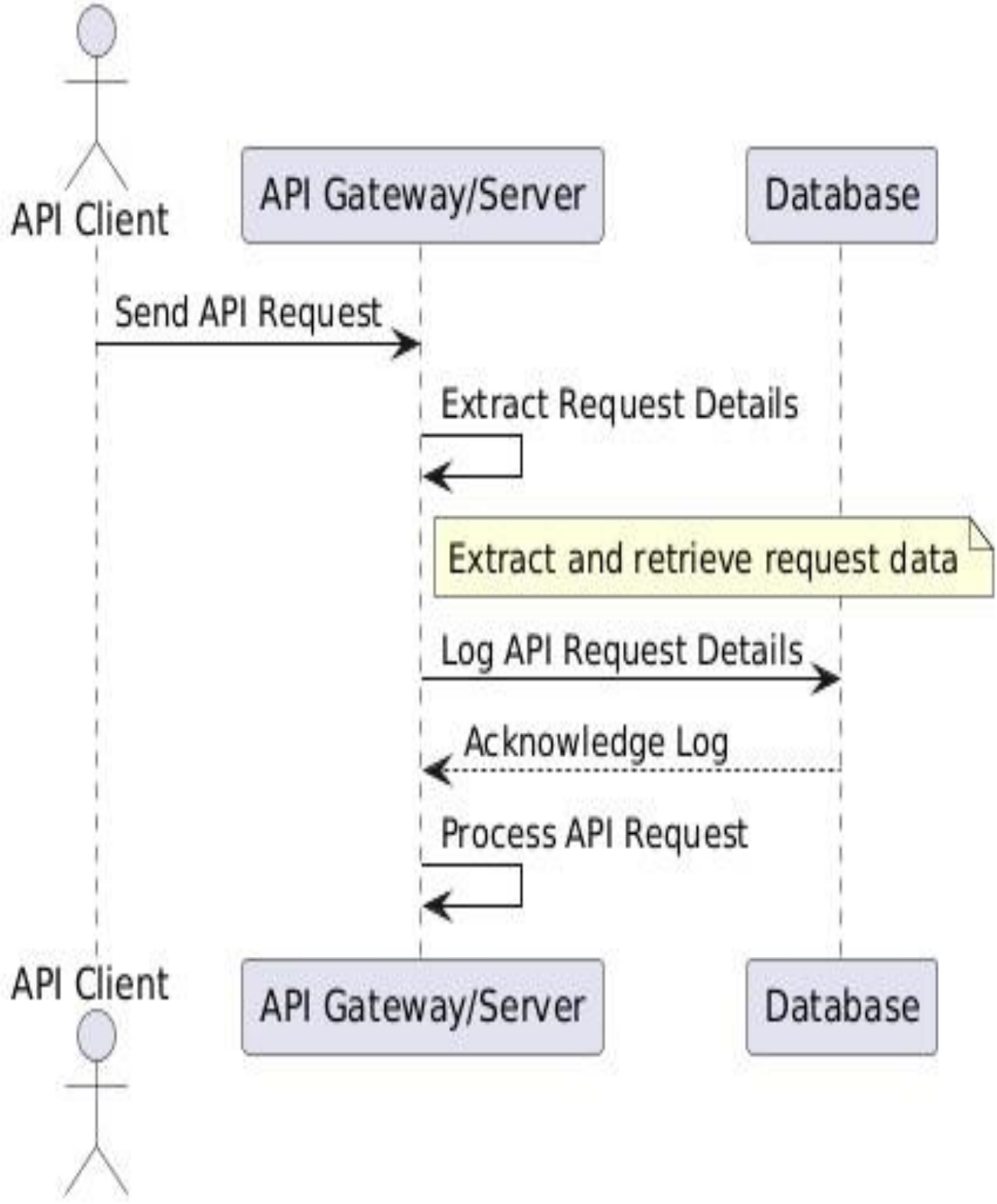


Figure 54 *API logging Sequence Diagram*

3.3.9 User History

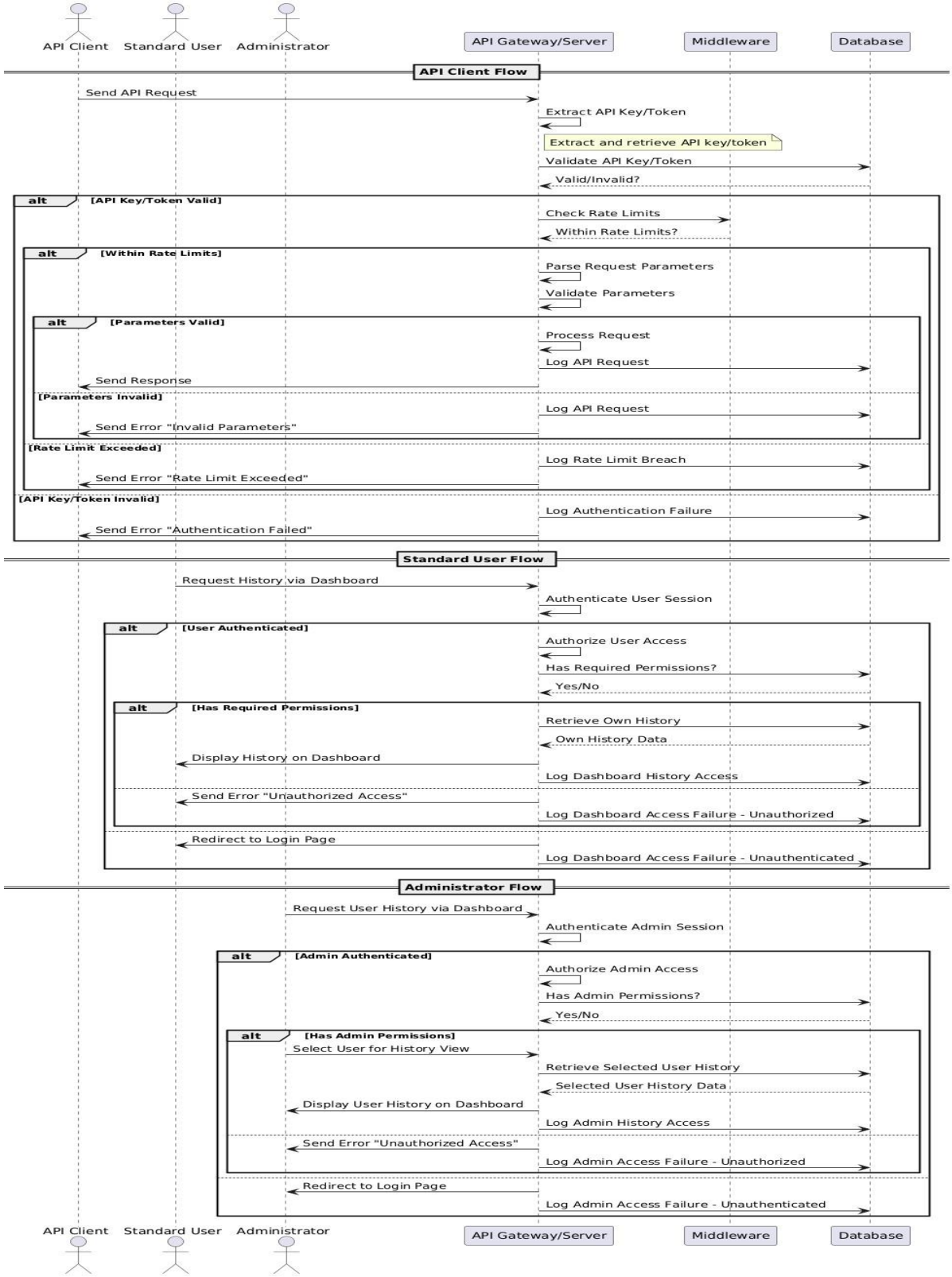


Figure 55 *User History Sequence Diagram*

3.3.10 API Key management

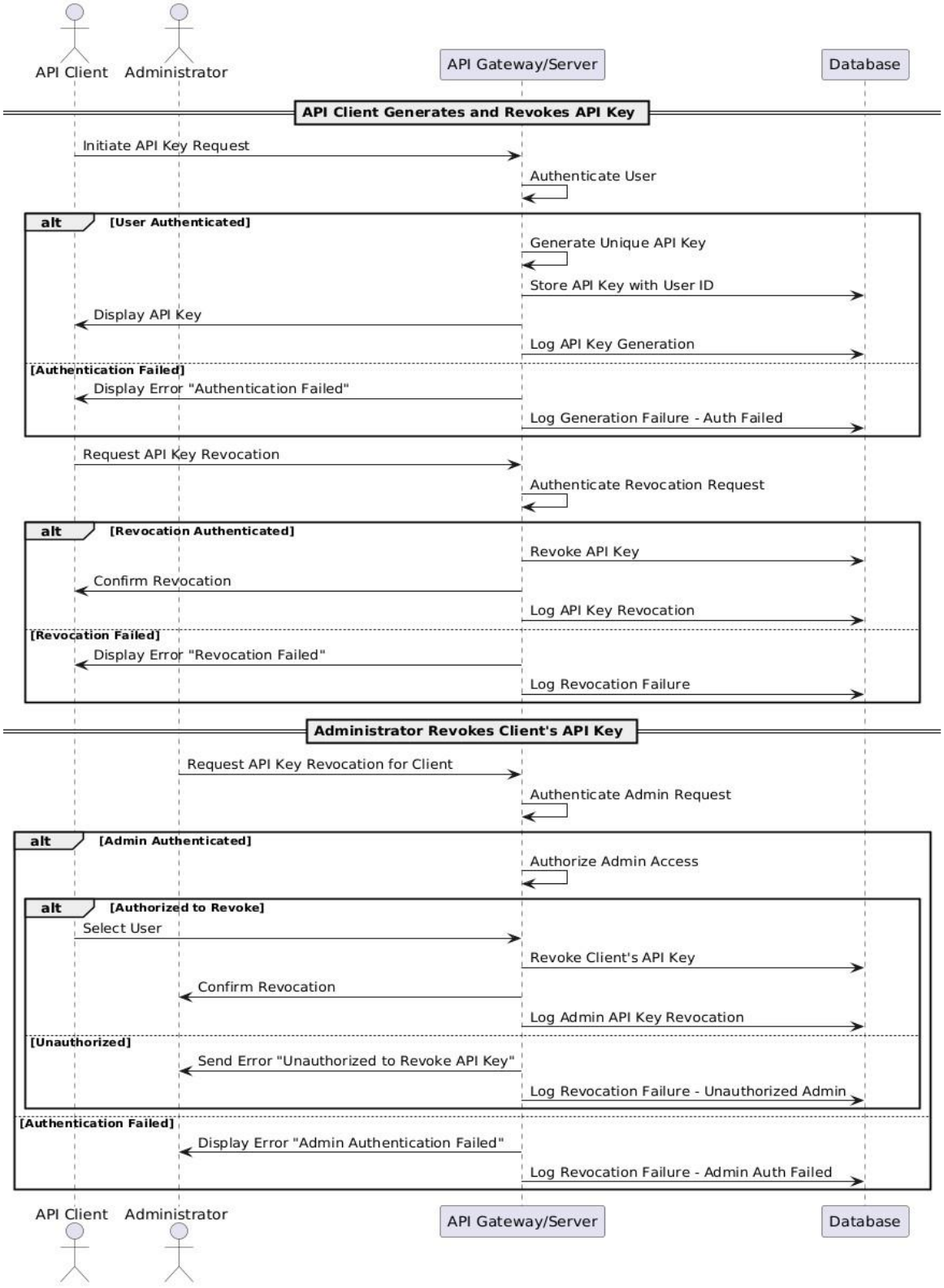


Figure 56 *API Key Management Sequence Diagram*

3.3.11 API monitoring

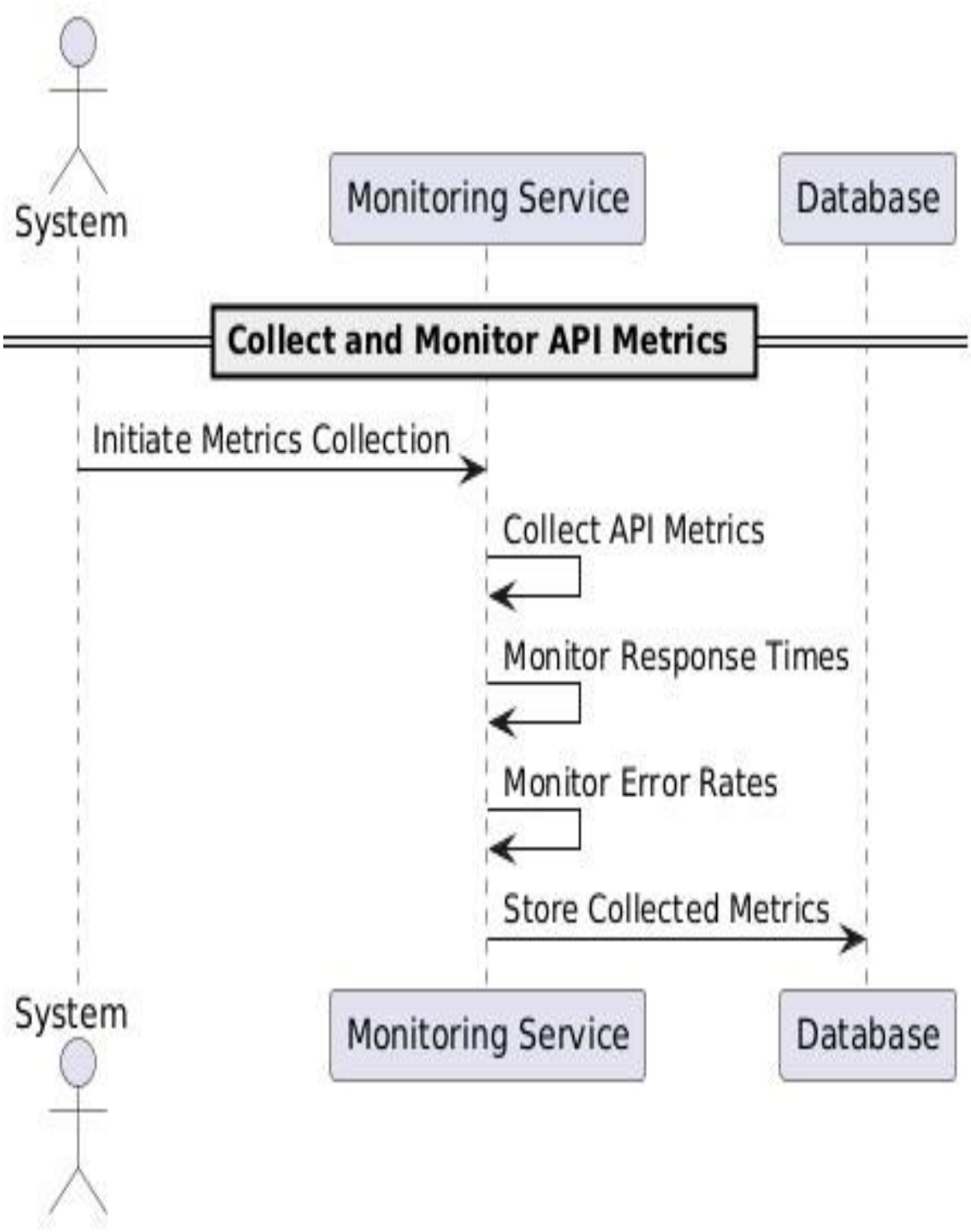


Figure 57 *API Monitoring Sequence Diagram*

3.3.12 Text Reuse Detection

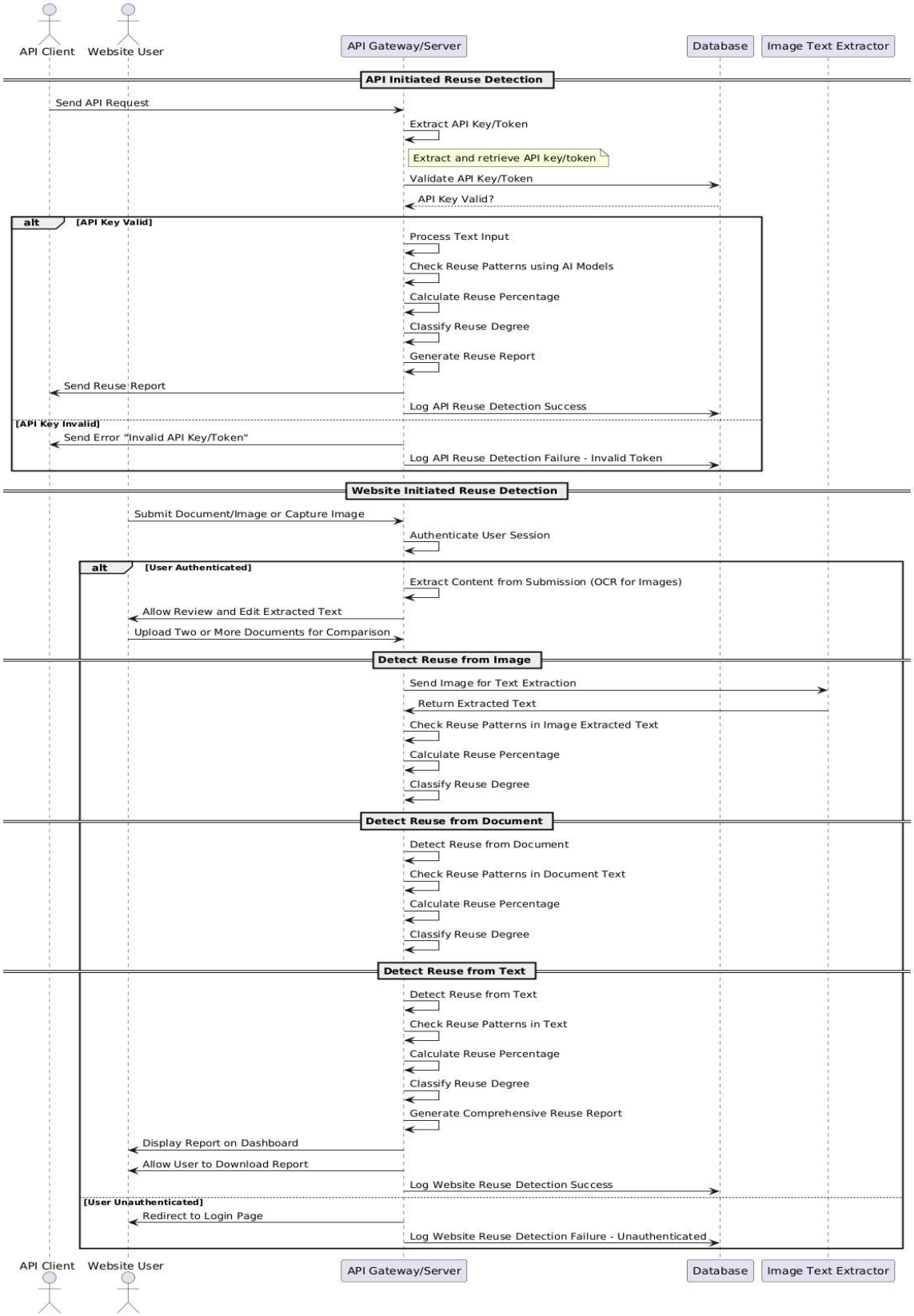


Figure 58 *Text Reuse Detection Sequence Diagram*

2.1.2. Plagiarism Detection

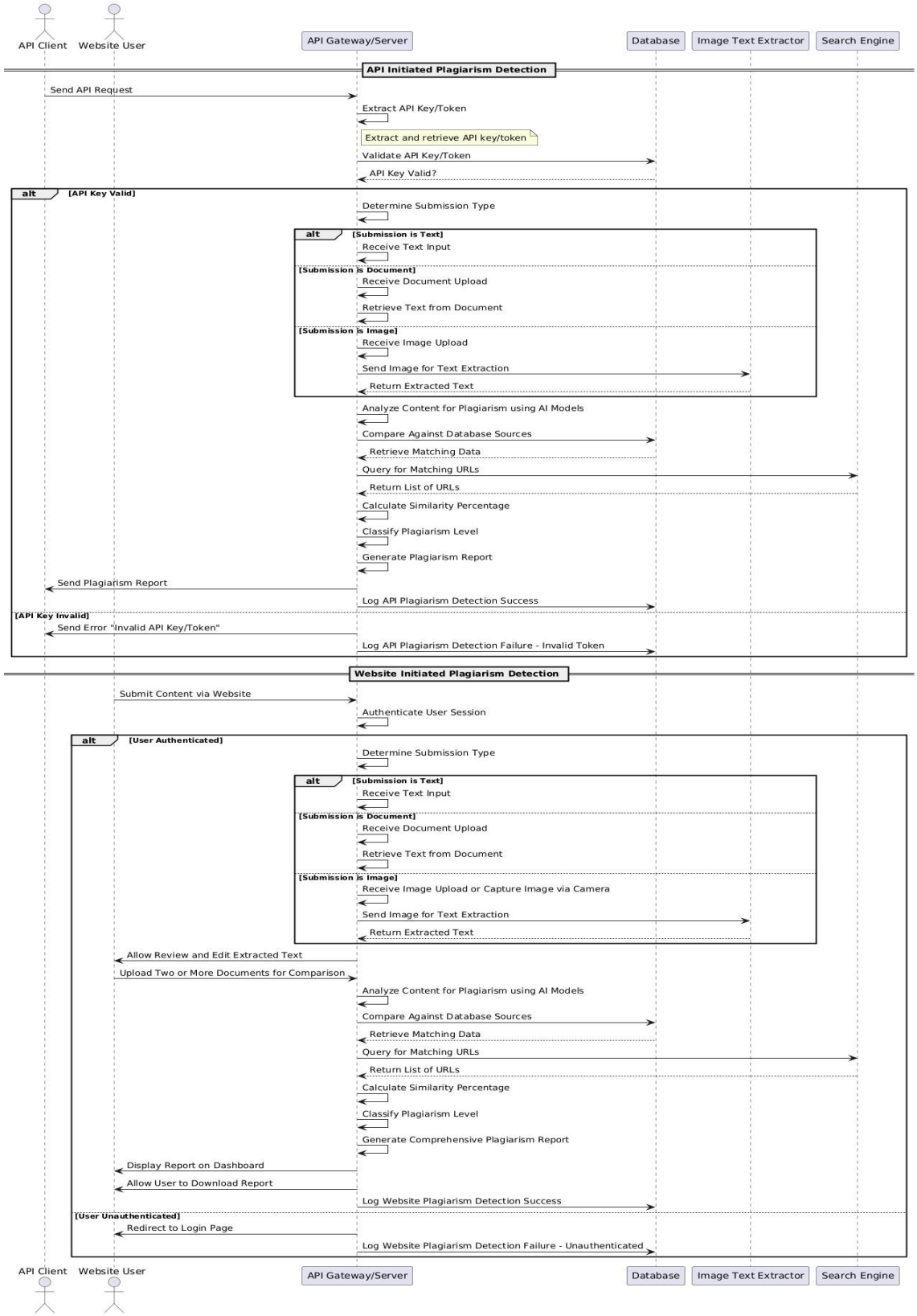


Figure 59 *Plagiarism Detection Sequence Diagram*

3.3.13 Contact

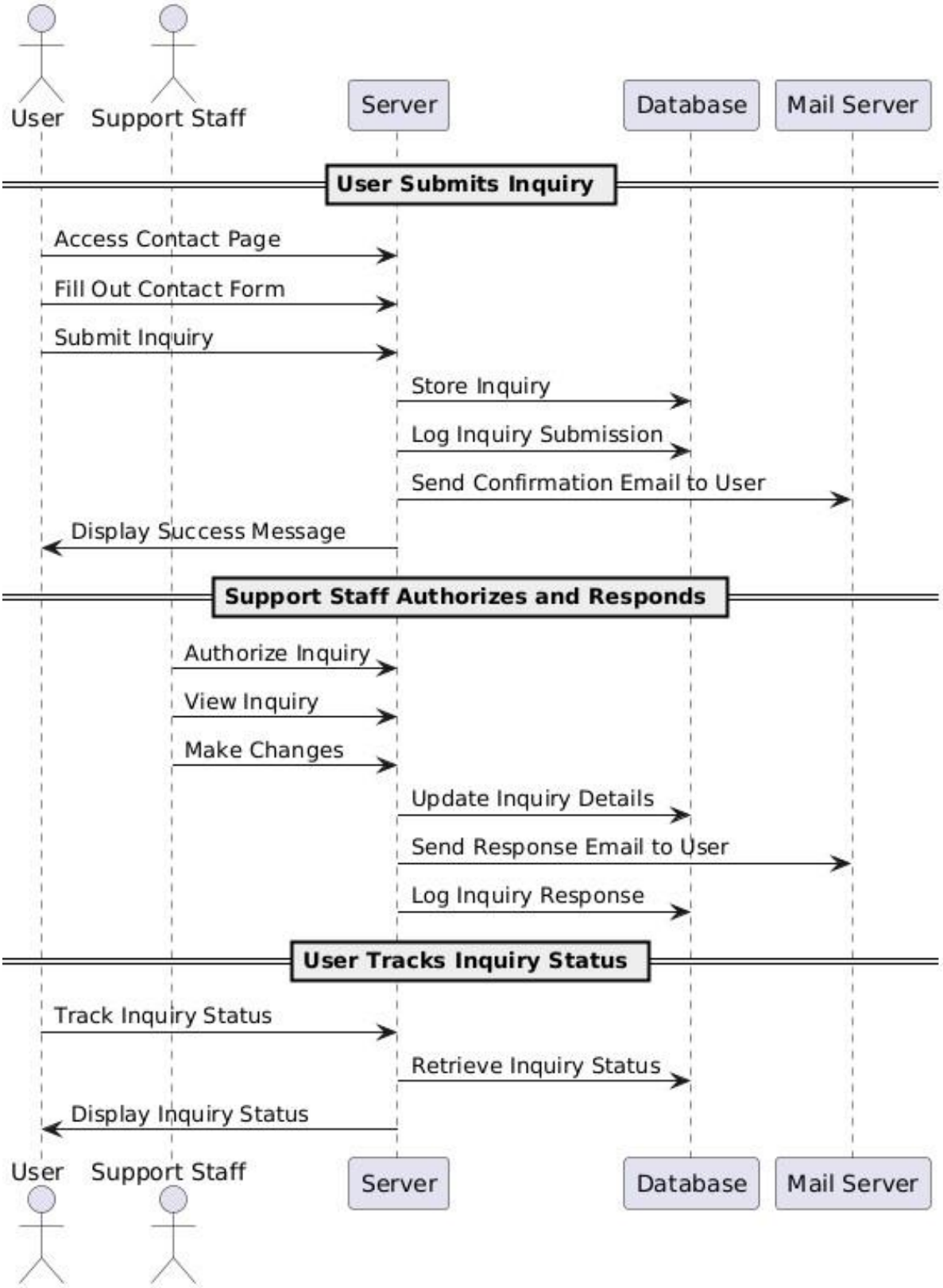


Figure 60 *Contact Sequence Diagram*

3.3.14 Notification

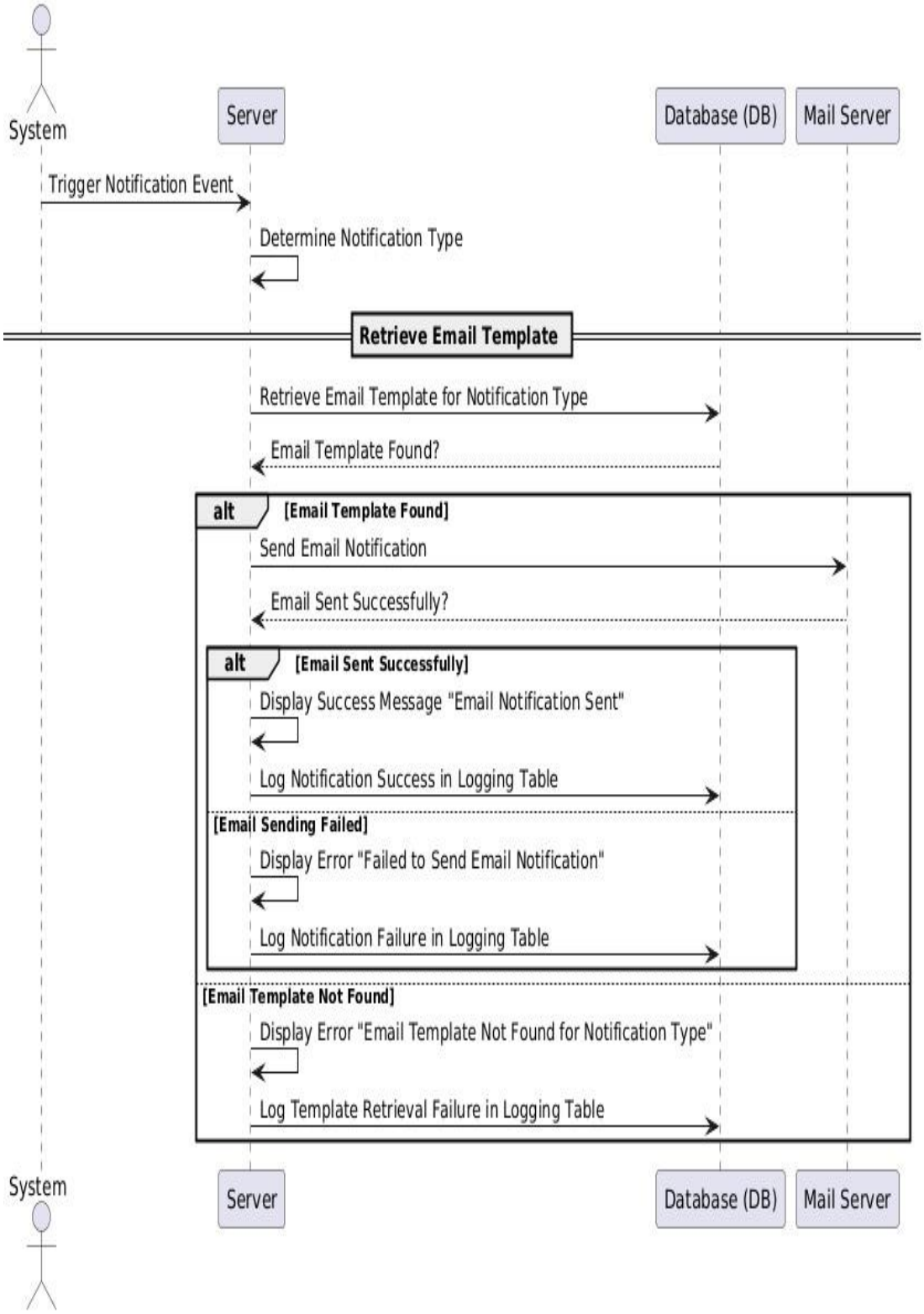


Figure 61 *Notification Sequence Diagram*

3.3.15 Admin Dashboard

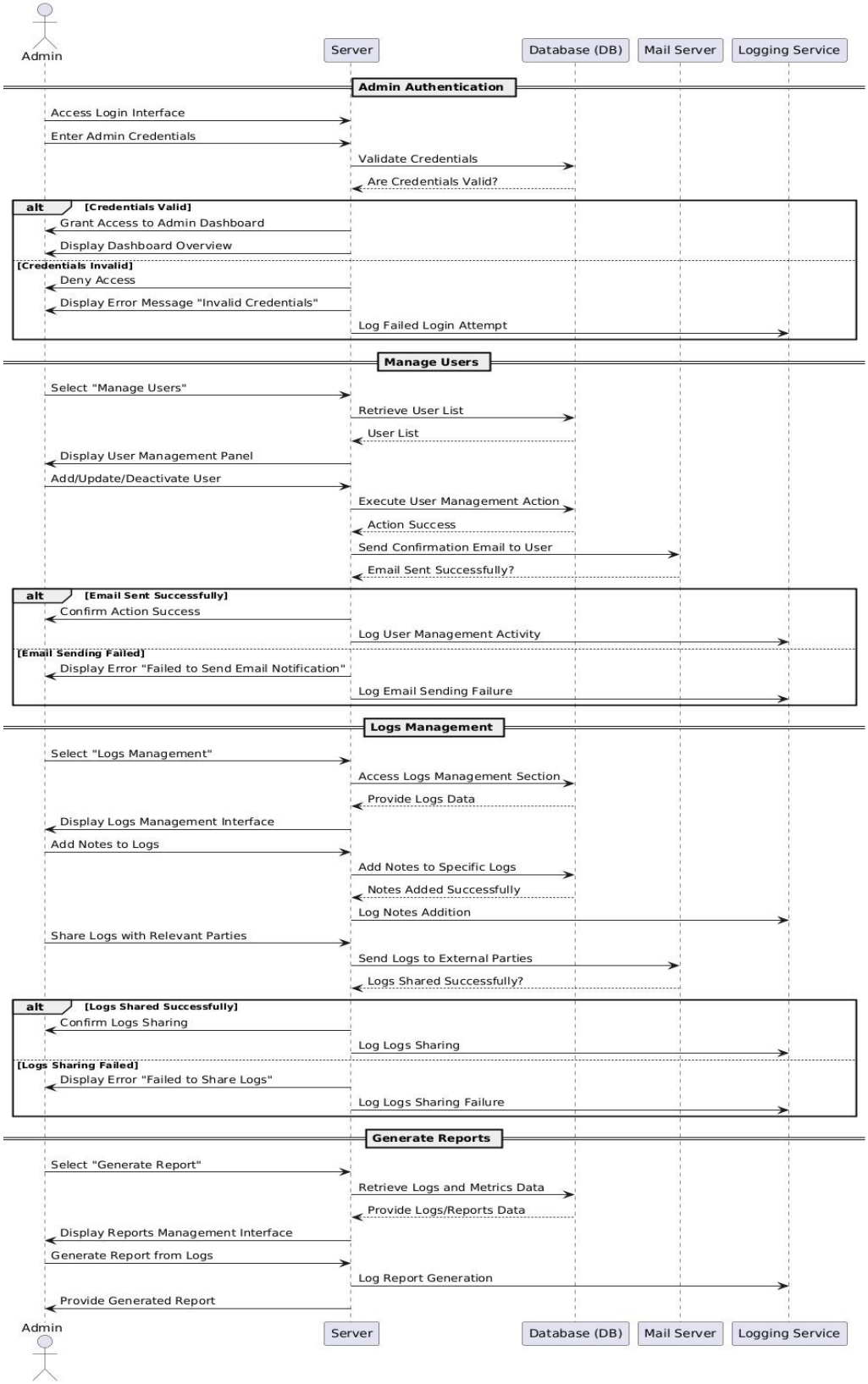


Figure 62 *Admin Dashboard Sequence Diagram*

3.3.16 System Management

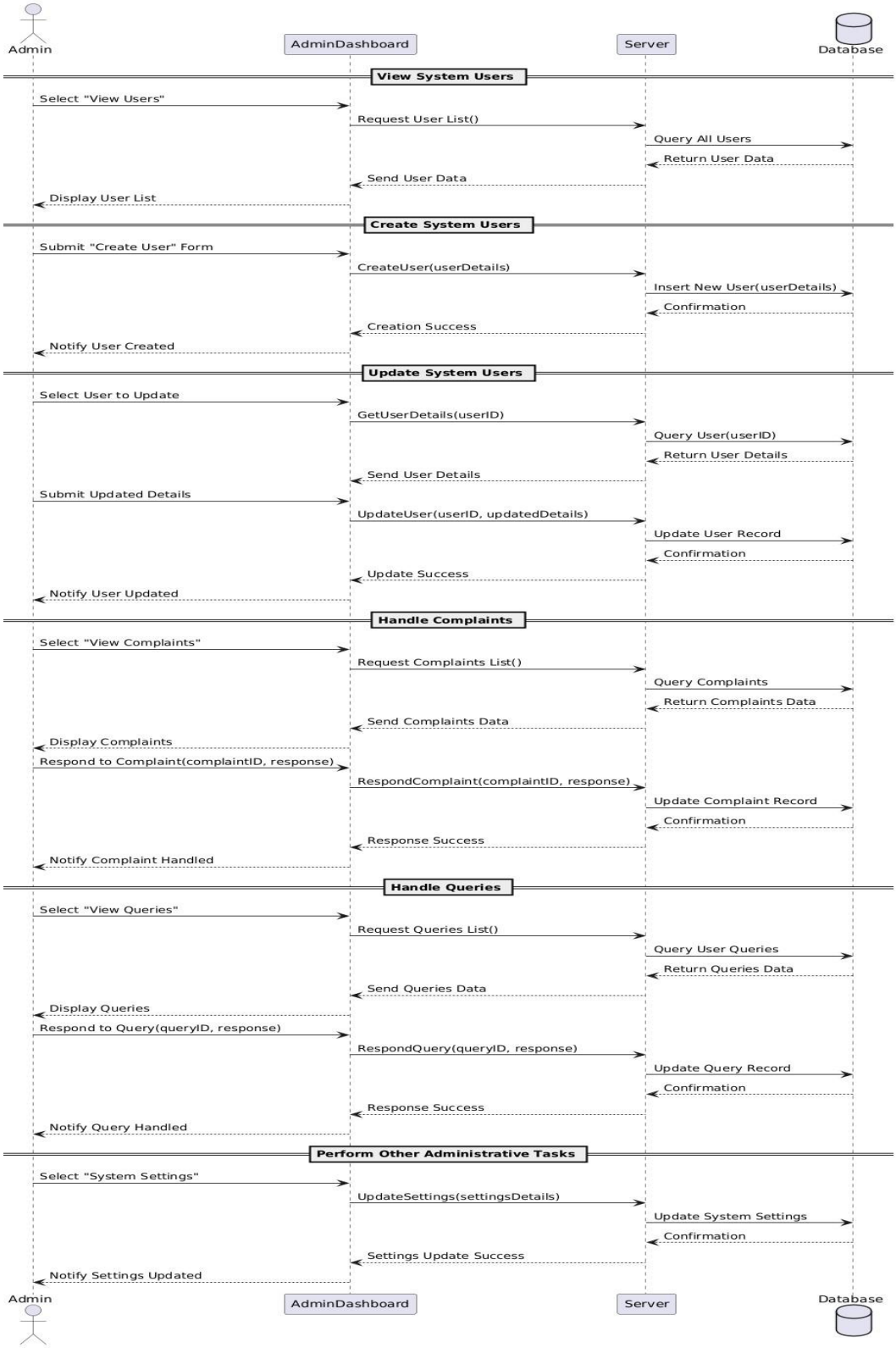


Figure 63 *System Management Sequence Diagram*

3.3.17 Report Generation

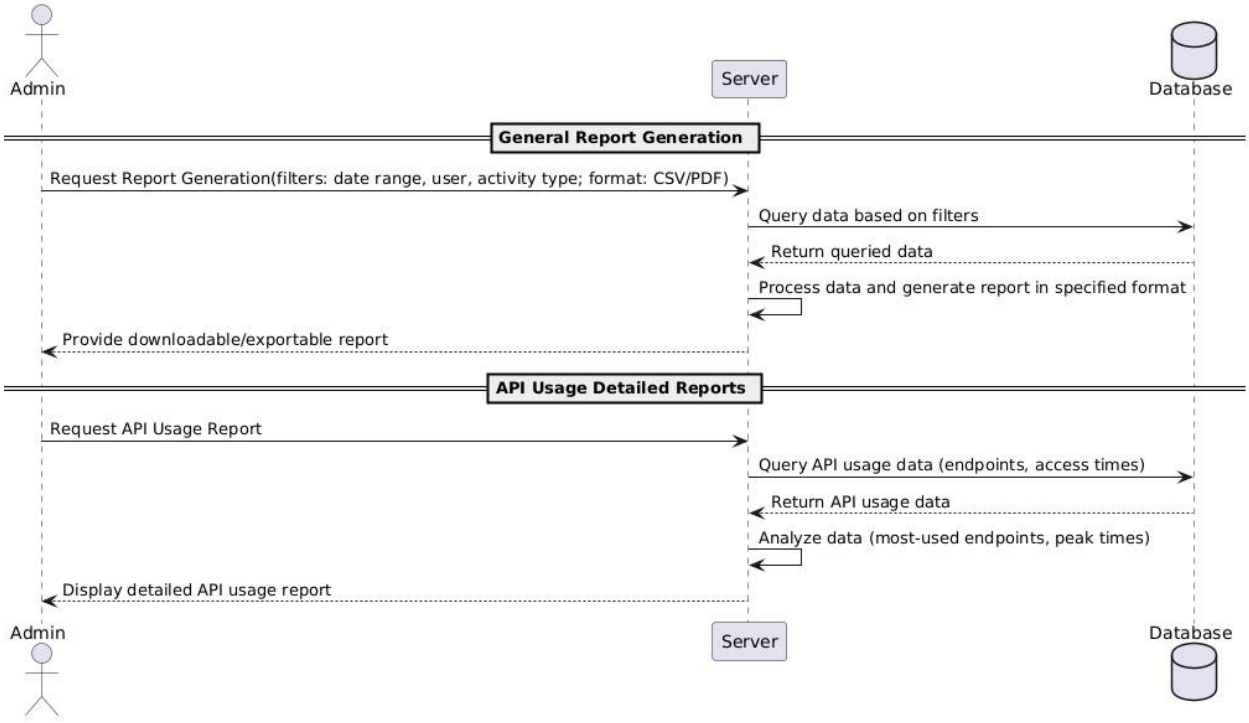


Figure 64 *Report Generation Sequence Diagram*

3.3.18 Analytics

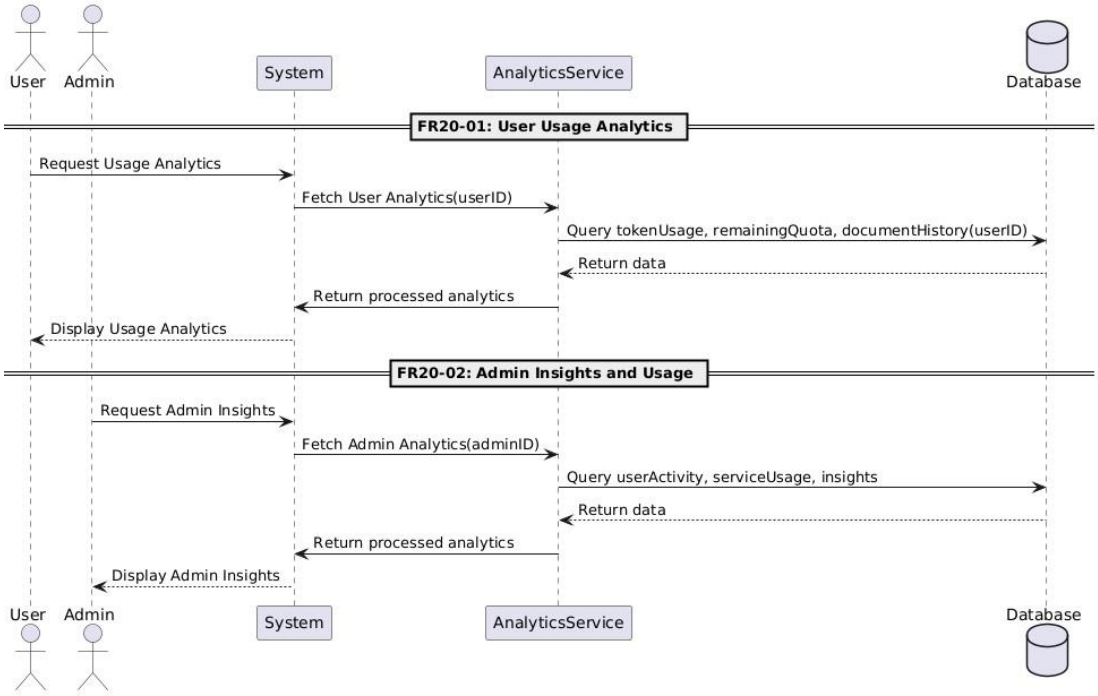


Figure 65 *Analytics Sequence Diagram*

3.4 Software architecture

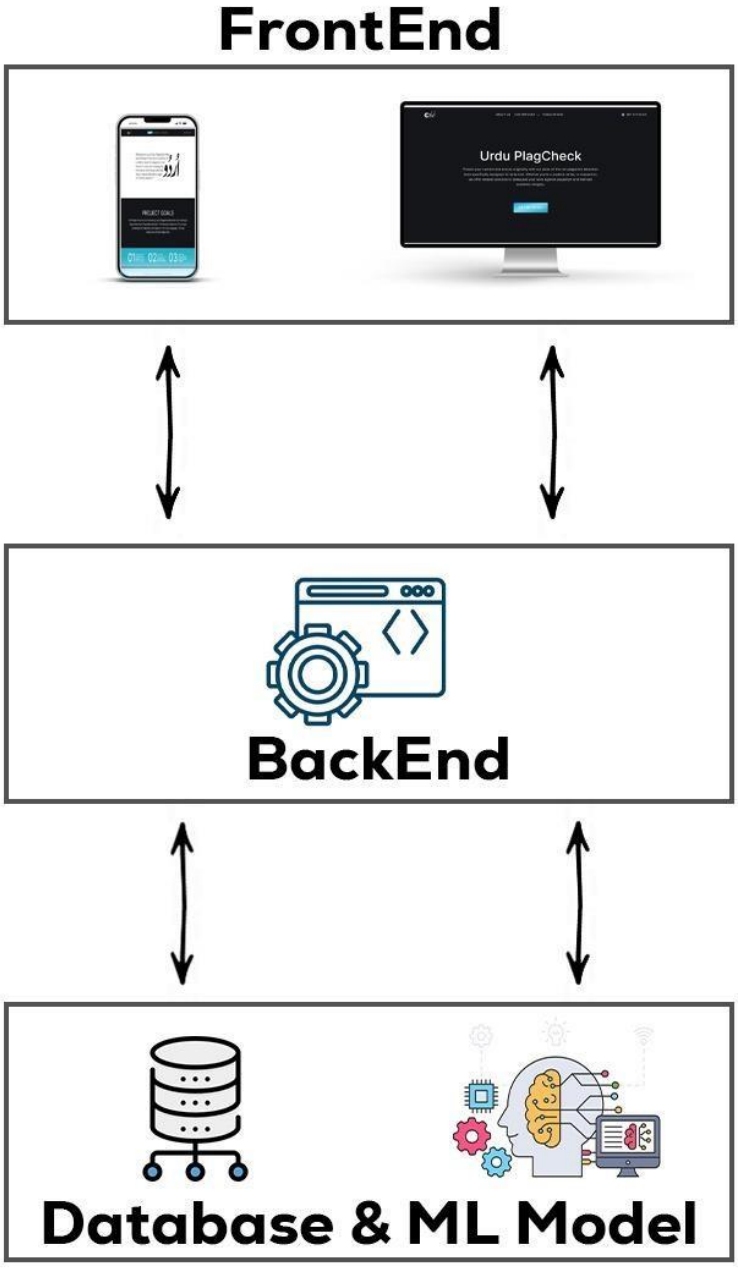


Figure 66 *Software Architecture Diagram*

3.5 Model

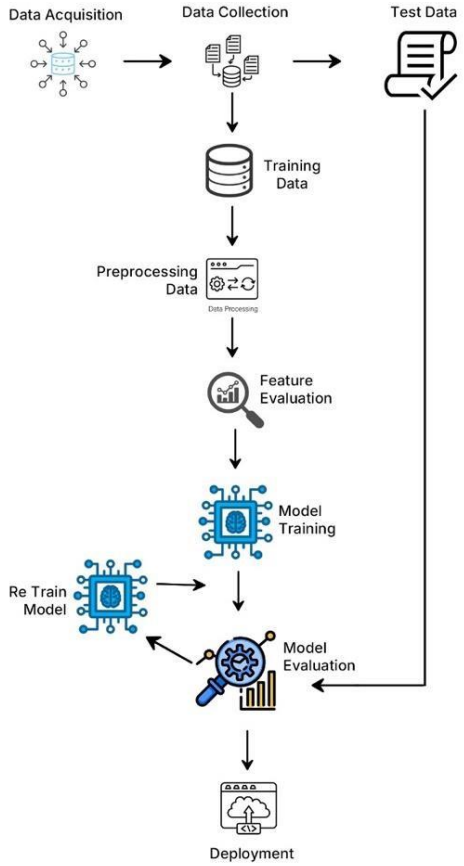


Figure 67 *Machine Learning Model Process*

3.6 Class diagram

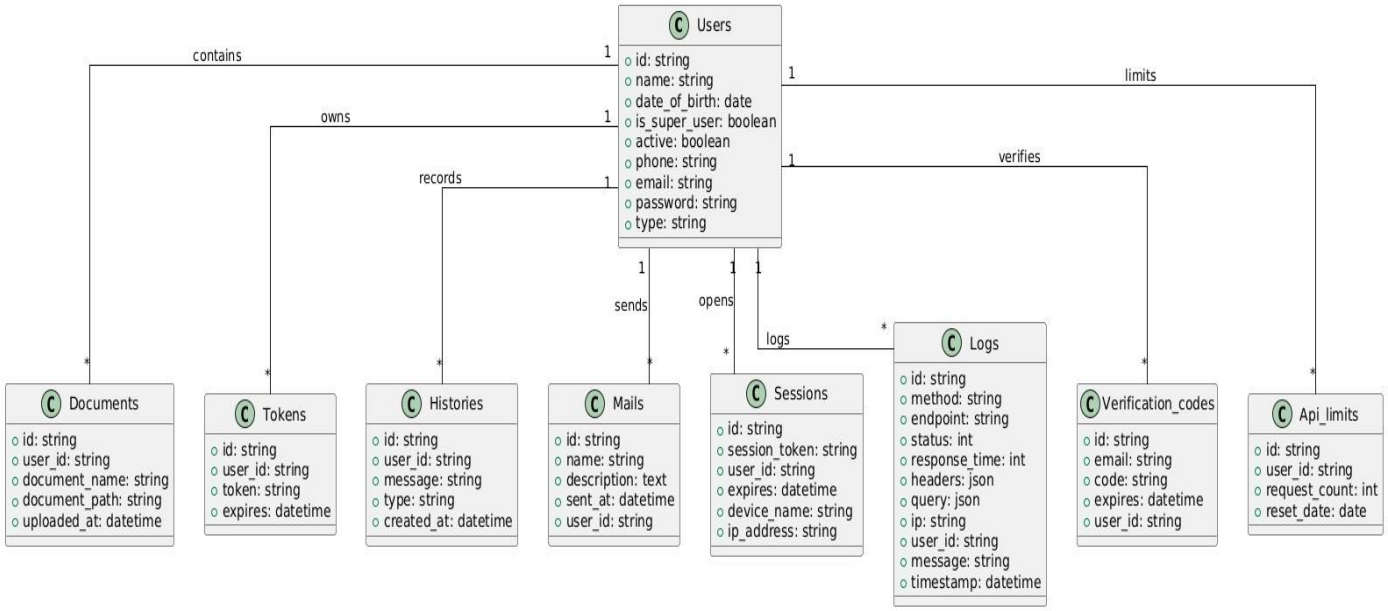


Figure 68 Class Diagram

3.7 Database diagram

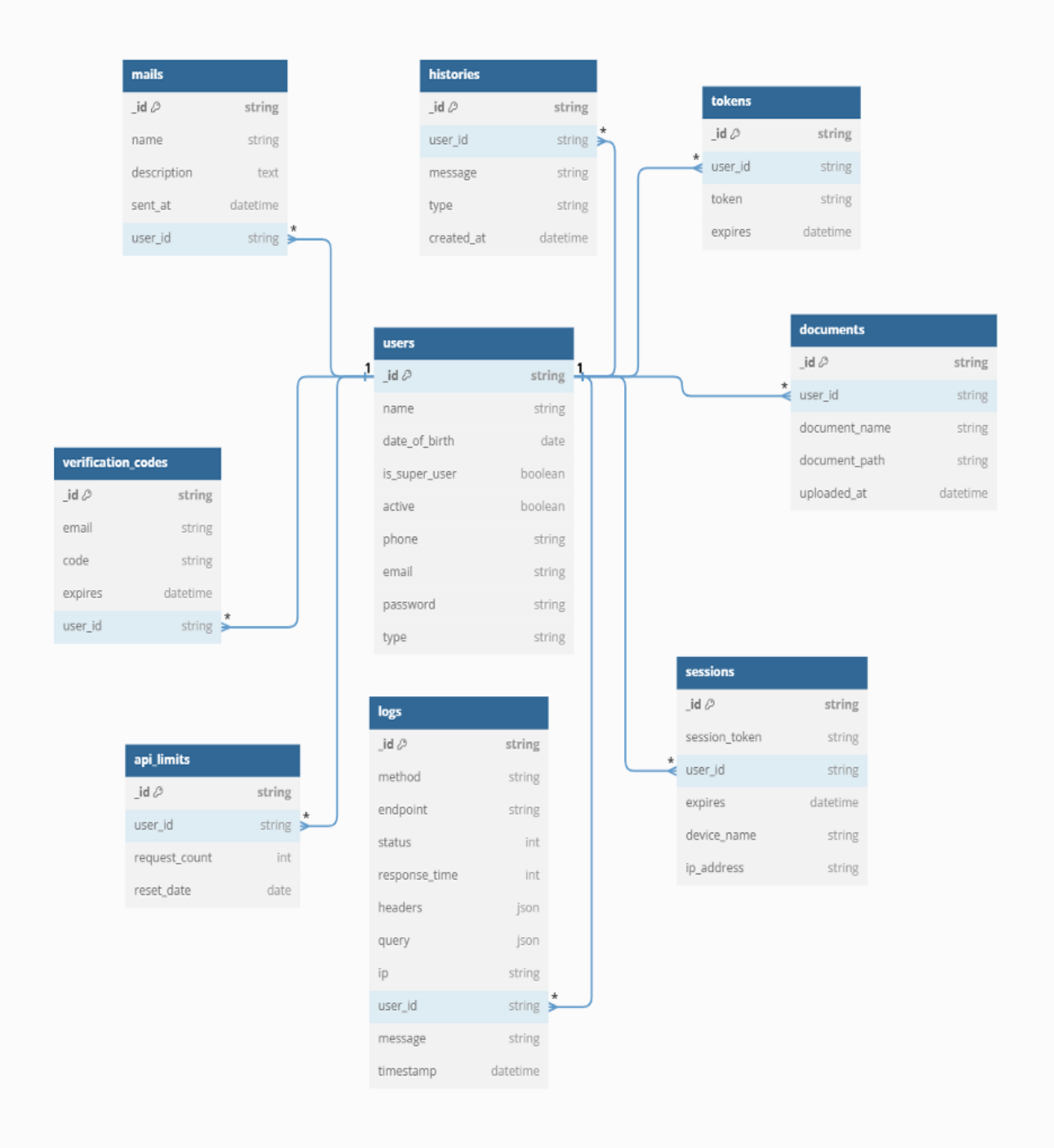
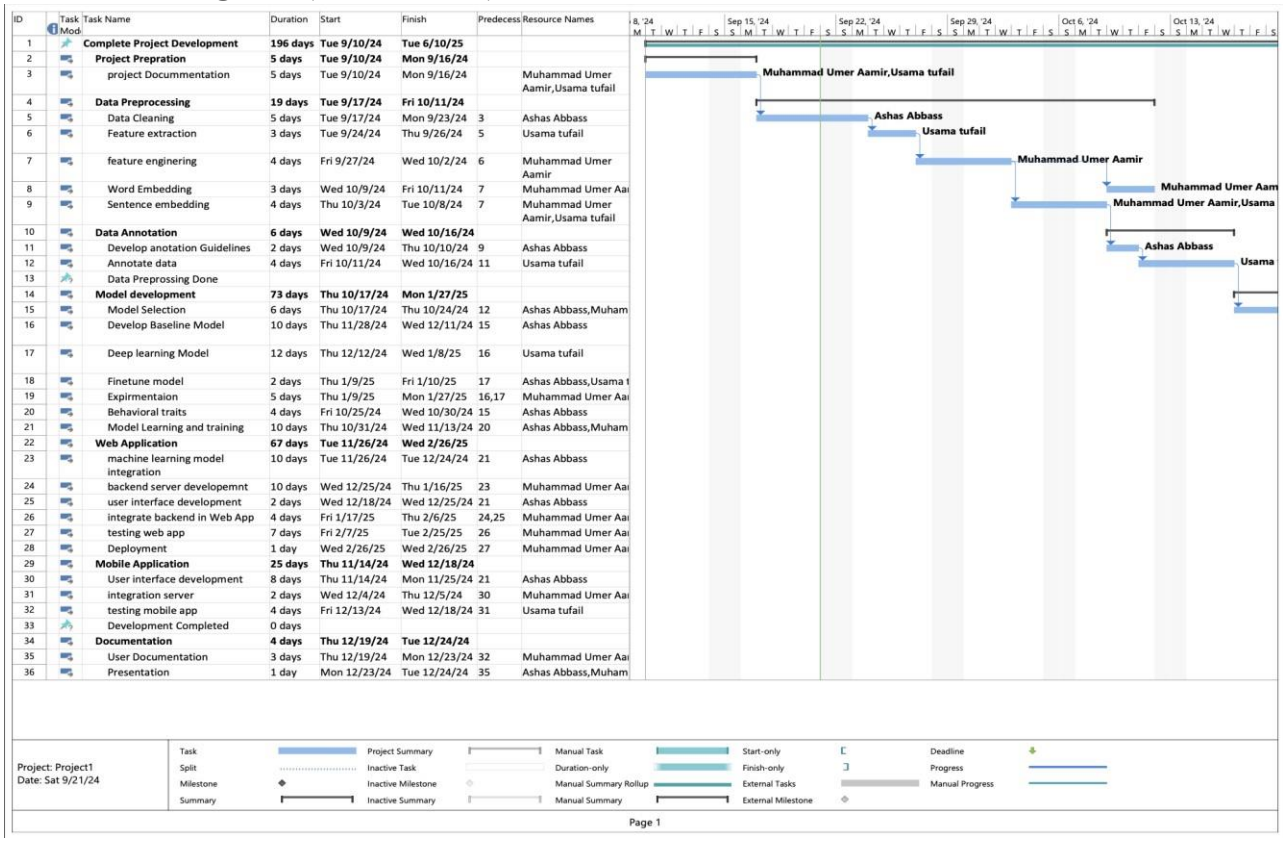


Figure 69 *Database Diagram*

3.8 Network diagram (Gantt chart)



ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		**Complete Project Development**	**196 days**	**Tue 9/10/24**	**Tue 6/10/25**		
2		**Project Prepration**	**5 days**	**Tue 9/10/24**	**Mon 9/16/24**		
3		project Documentation	5 days	Tue 9/10/24	Mon 9/16/24		Muhammad Umer Aamir,Usama tufail
4		**Data Preprocessing**	**19 days**	**Tue 9/17/24**	**Fri 10/11/24**		
5		Data Cleaning	5 days	Tue 9/17/24	Mon 9/23/24	3	Ashas Abbass
6		Feature extraction	3 days	Tue 9/24/24	Thu 9/26/24	5	Usama tufail
7		feature enginering	4 days	Fri 9/27/24	Wed 10/2/24	6	Muhammad Umer Aamir
8		Word Embedding	3 days	Wed 10/9/24	Fri 10/11/24	7	Muhammad Umer Aa
9		Sentence embedding	4 days	Thu 10/3/24	Tue 10/8/24	7	Muhammad Umer Aamir,Usama tufail
10		**Data Annotation**	**6 days**	**Wed 10/9/24**	**Wed 10/16/24**		
11		Develop anotation Guidelines	2 days	Wed 10/9/24	Thu 10/10/24	9	Ashas Abbass
12		Annotate data	4 days	Fri 10/11/24	Wed 10/16/24	11	Usama tufail
13		Data Preprossing Done					
14		**Model development**	**73 days**	**Thu 10/17/24**	**Mon 1/27/25**		
15		Model Selection	6 days	Thu 10/17/24	Thu 10/24/24	12	Ashas Abbass,Muham
16		Develop Baseline Model	10 days	Thu 11/28/24	Wed 12/11/24	15	Ashas Abbass
17		Deep learning Model	12 days	Thu 12/12/24	Wed 1/8/25	16	Usama tufail
18		Finetune model	2 days	Thu 1/9/25	Fri 1/10/25	17	Ashas Abbass,Usama t
19		Expirmentaion	5 days	Thu 1/9/25	Mon 1/27/25	16,17	Muhammad Umer Aa
20		Behavioral traits	4 days	Fri 10/25/24	Wed 10/30/24	15	Ashas Abbass
21		Model Learning and training	10 days	Thu 10/31/24	Wed 11/13/24	20	Ashas Abbass,Muham
22		**Web Application**	**67 days**	**Tue 11/26/24**	**Wed 2/26/25**		
23		machine learning model integration	10 days	Tue 11/26/24	Tue 12/24/24	21	Ashas Abbass
24		backend server developemnt	10 days	Wed 12/25/24	Thu 1/16/25	23	Muhammad Umer Aa
25		user interface development	2 days	Wed 12/18/24	Wed 12/25/24	21	Ashas Abbass
26		integrate backend in Web App	4 days	Fri 1/17/25	Thu 2/6/25	24,25	Muhammad Umer Aa
27		testing web app	7 days	Fri 2/7/25	Tue 2/25/25	26	Muhammad Umer Aa
28		Deployment	1 day	Wed 2/26/25	Wed 2/26/25	27	Muhammad Umer Aa
29		**Mobile Application**	**25 days**	**Thu 11/14/24**	**Wed 12/18/24**		
30		User interface development	8 days	Thu 11/14/24	Mon 11/25/24	21	Ashas Abbass
31		integration server	2 days	Wed 12/4/24	Thu 12/5/24	30	Muhammad Umer Aa
32		testing mobile app	4 days	Fri 12/13/24	Wed 12/18/24	31	Usama tufail
33		Development Completed	0 days				
34		**Documentation**	**4 days**	**Thu 12/19/24**	**Tue 12/24/24**		
35		User Documentation	3 days	Thu 12/19/24	Mon 12/23/24	32	Muhammad Umer Aa
36		Presentation	1 day	Mon 12/23/24	Tue 12/24/24	35	Ashas Abbass,Muham

Figure 70 *Network diagram (Gantt chart)*

3.9 Collaboration Diagram

3.9.1 Login

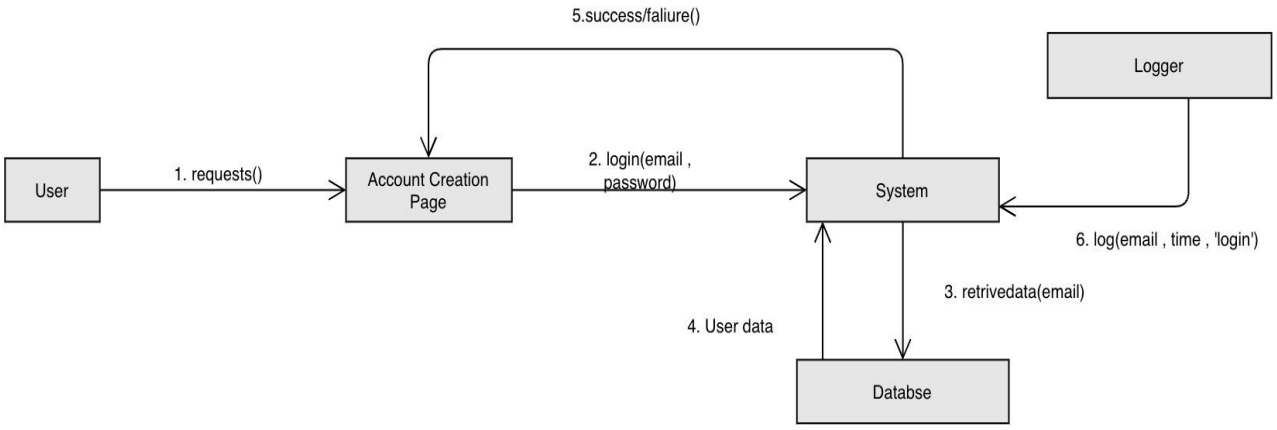


Figure 71 *Login Collaboration Diagram*

3.9.2 Create Account

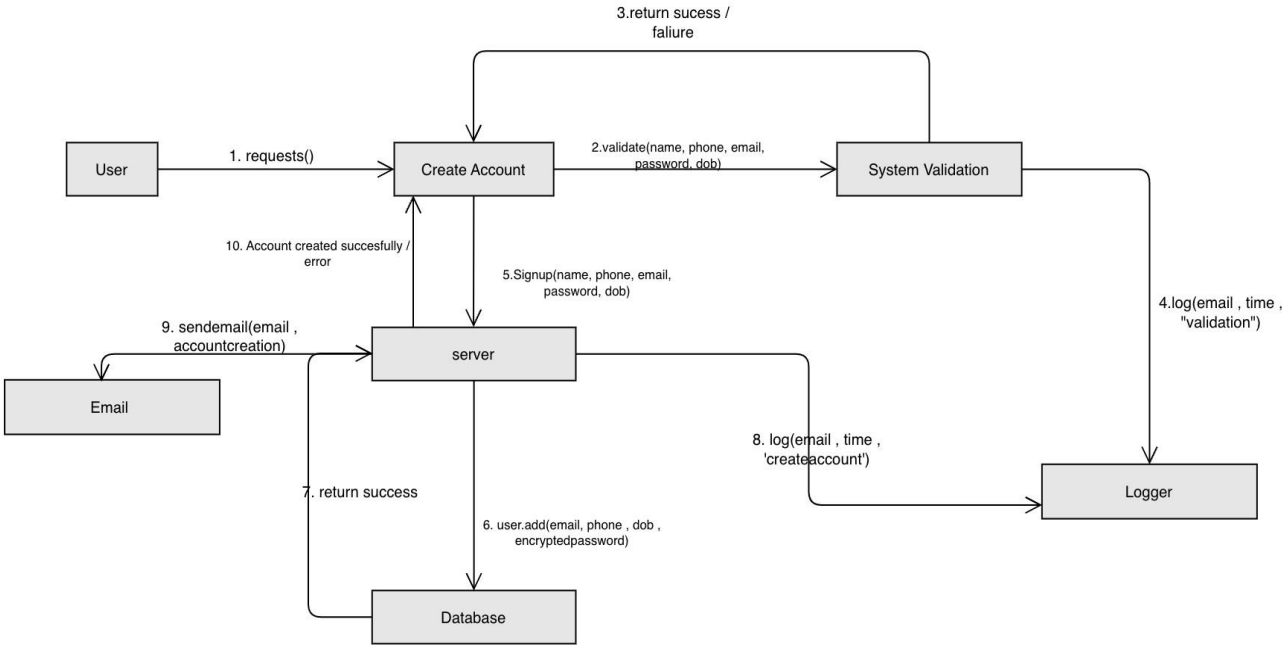


Figure 72 *Create Account Collaboration Diagram*

3.9.3 Forgot Password

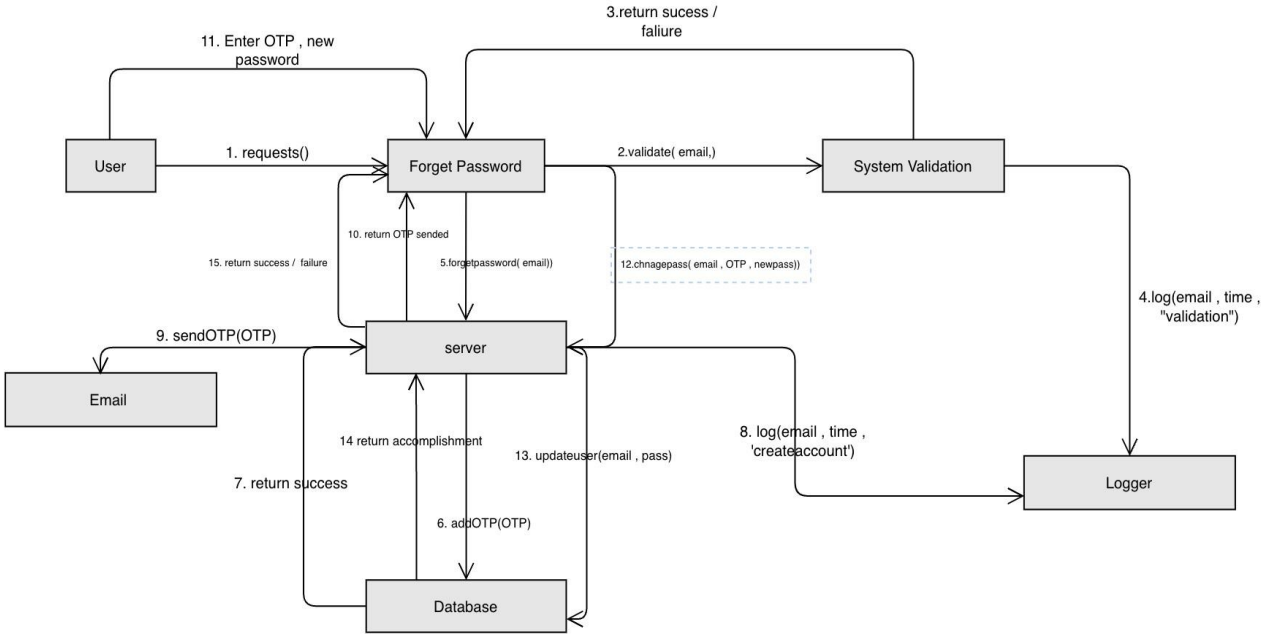


Figure 73 *Forgot Password Collaboration Diagram*

3.9.4 Change Password

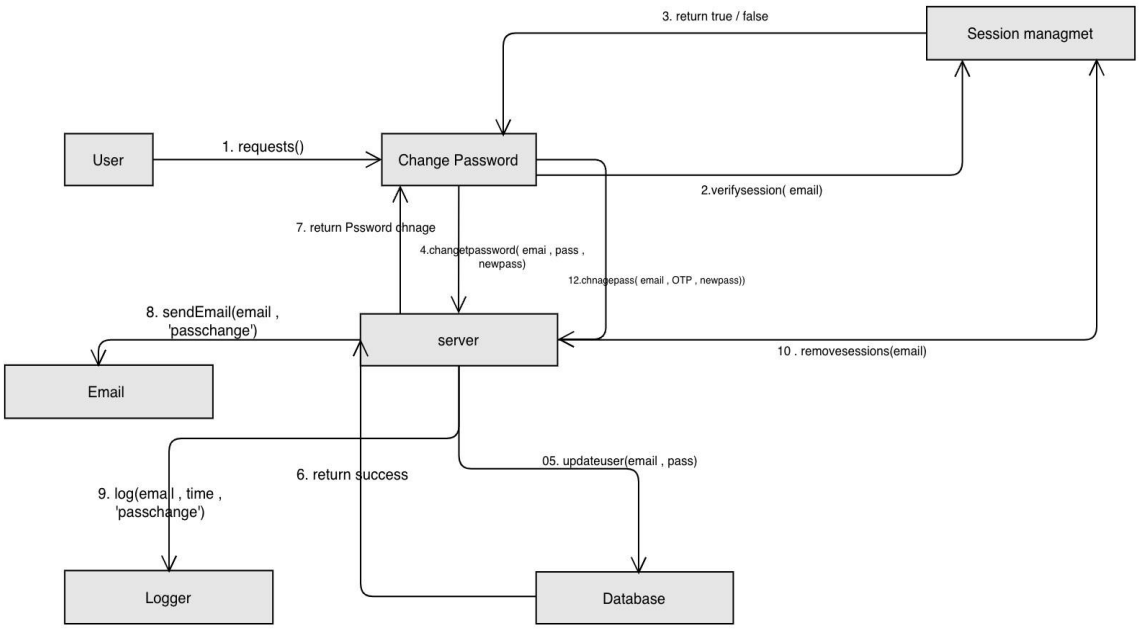


Figure 74 *Change Password Collaboration Diagram*

3.9.5 Edit Profile

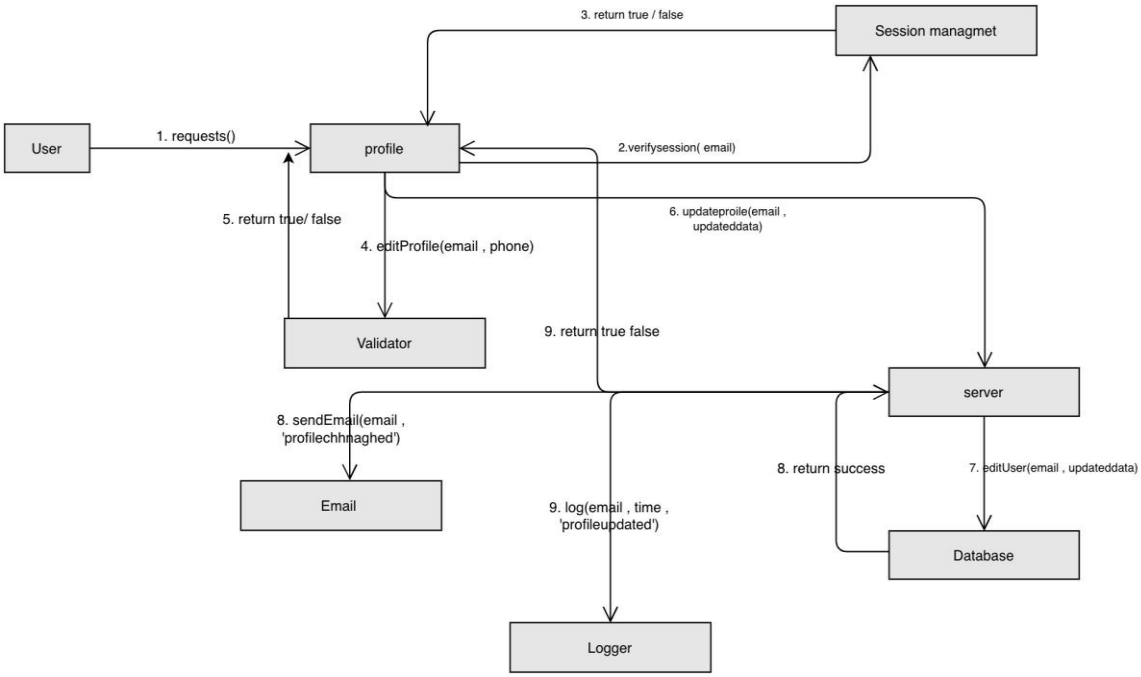


Figure 75 *Edit Profile Collaboration Diagram*

3.9.6 API services

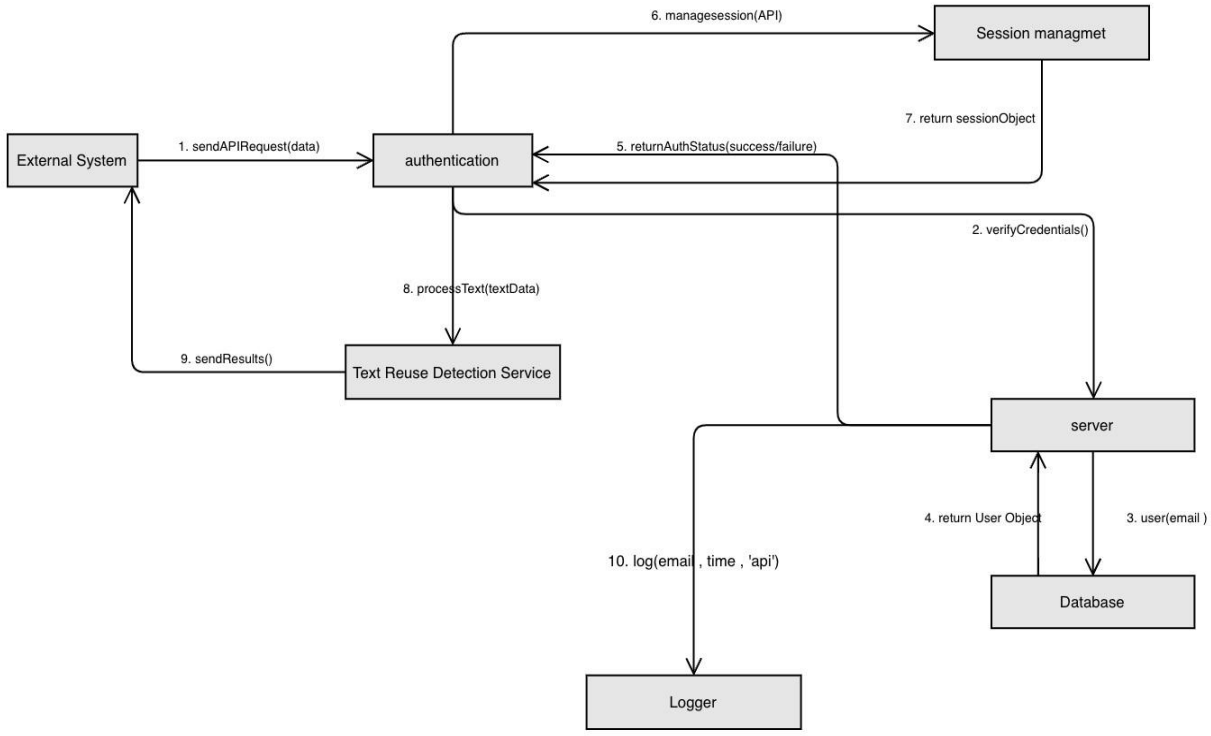


Figure 76 *API services Collaboration Diagram*

3.9.7 Rate Limiting for API

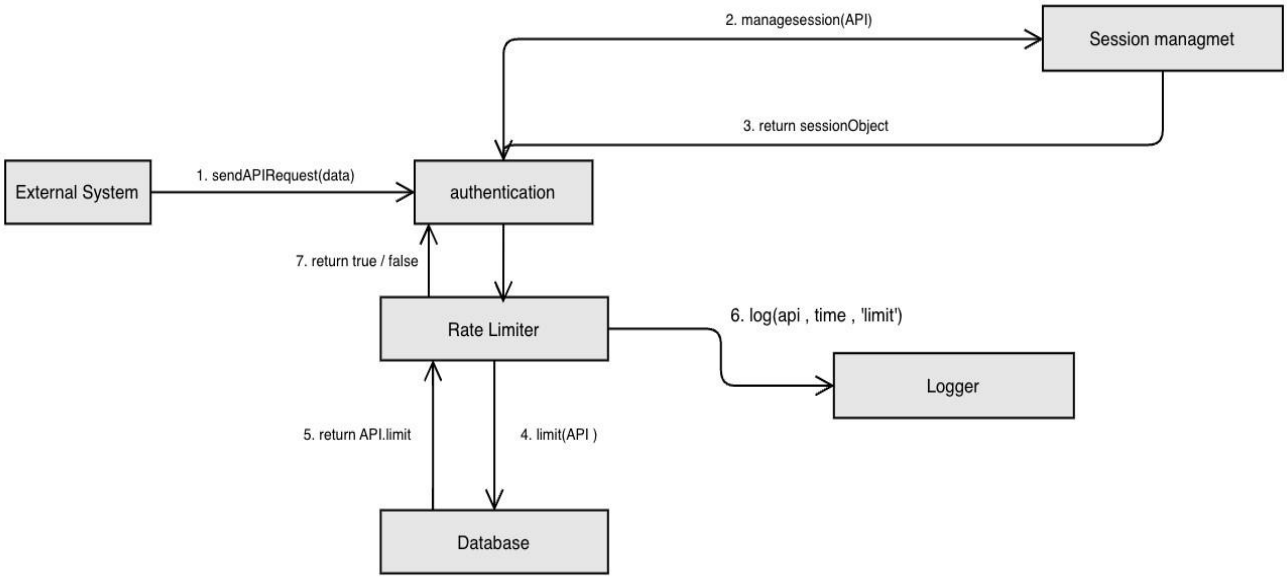


Figure 77 *Rate Limiting for API Collaboration Diagram*

3.9.8 Authorization for API

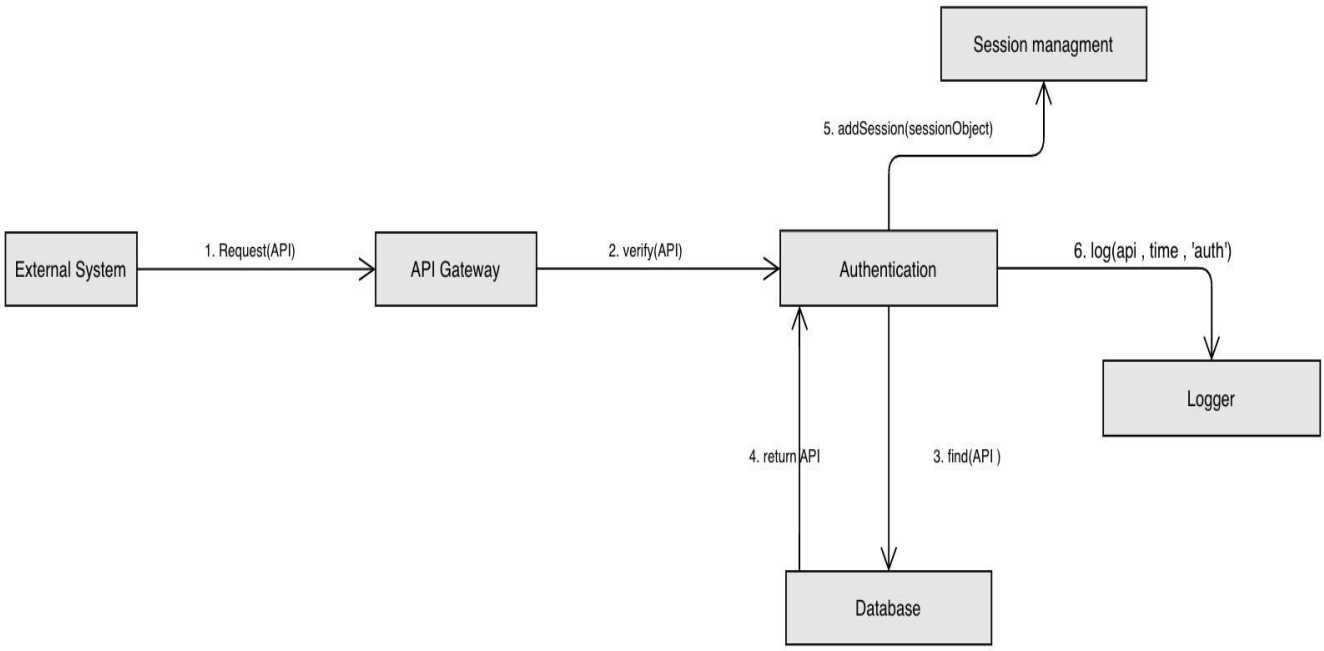


Figure 78 *Authorization for API Collaboration Diagram*

3.9.9 API logging

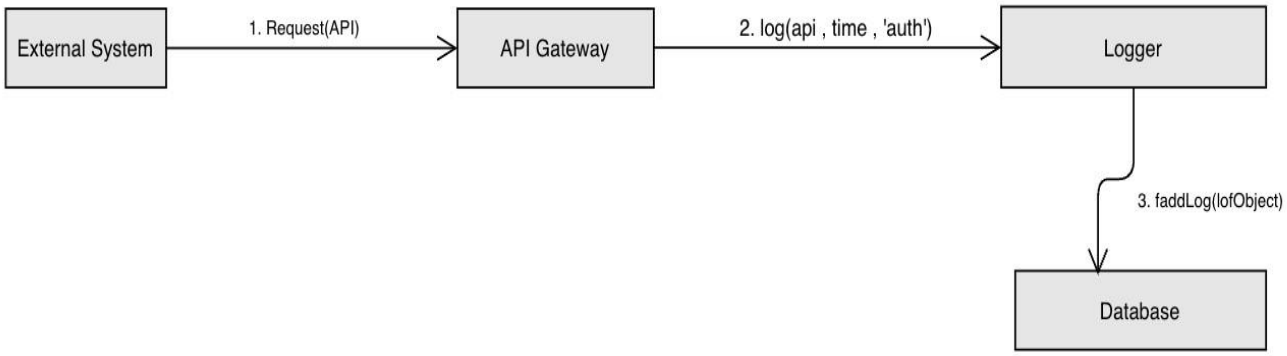


Figure 79 *API logging Collaboration Diagram*

3.9.10 User History

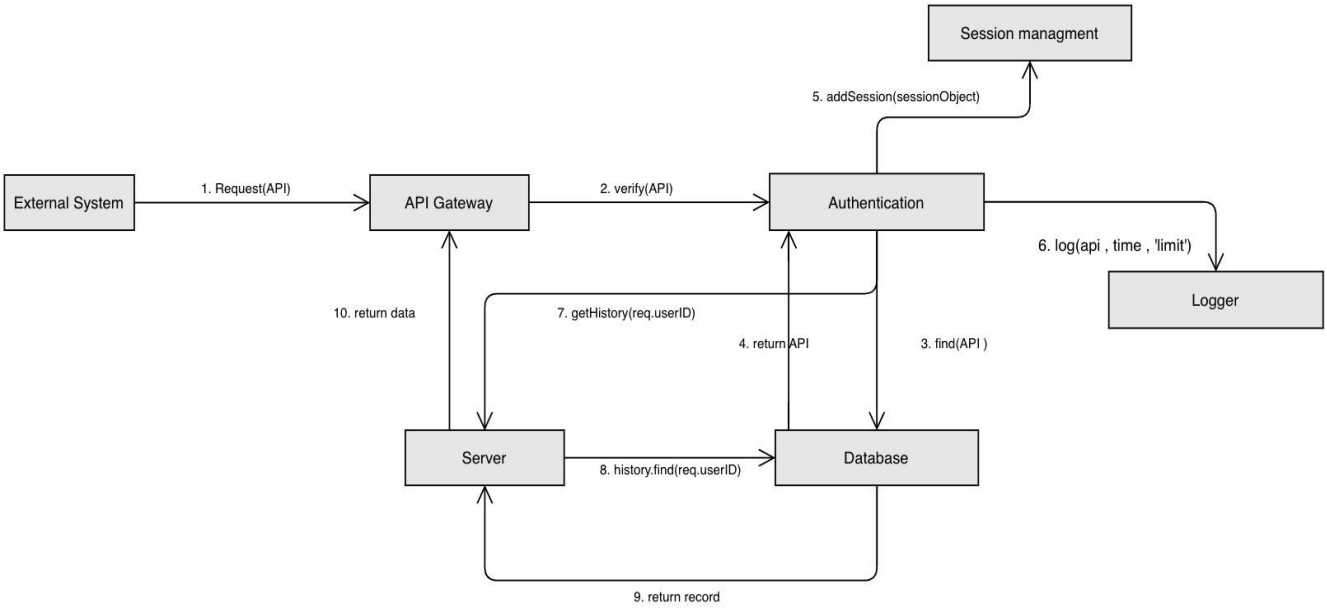


Figure 80 *User History Collaboration Diagram*

3.9.11 API Key management

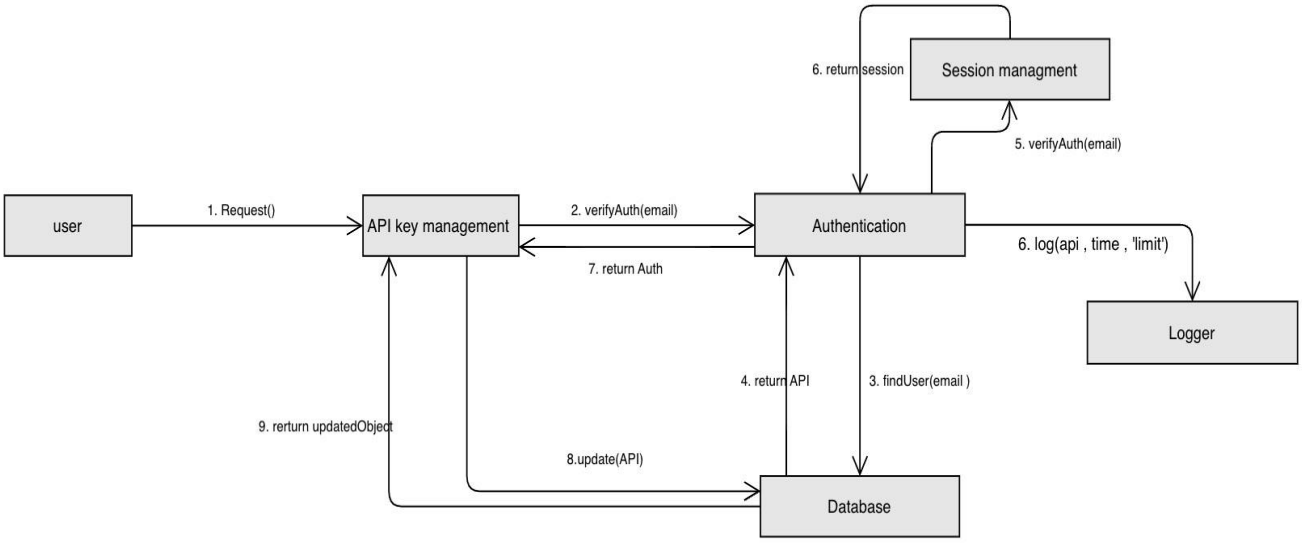


Figure 81 *API Key management Collaboration Diagram*

3.9.12 API monitoring

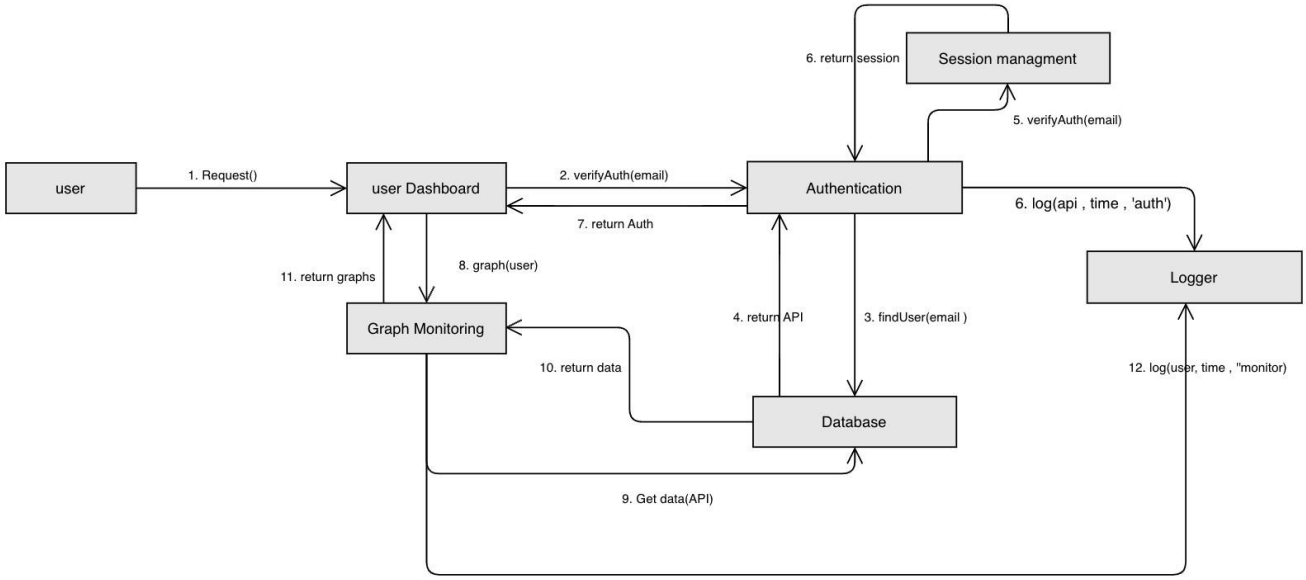


Figure 82 *API monitoring Collaboration Diagram*

3.9.13 Text Reuse Detection

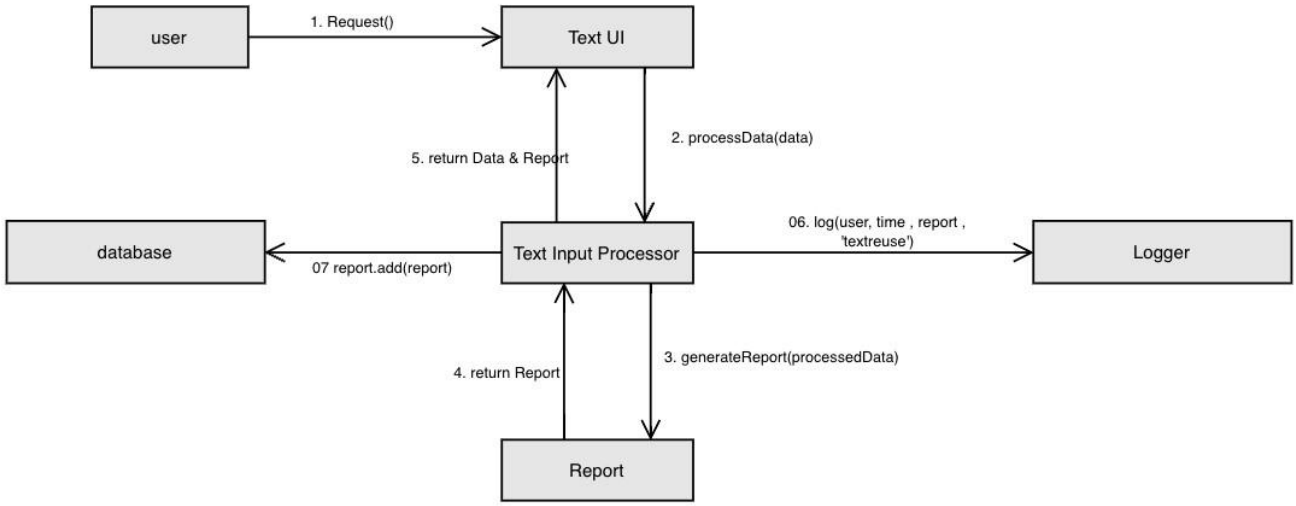


Figure 83 *Text Reuse Detection Collaboration Diagram*

3.9.14 Plagiarism Detection

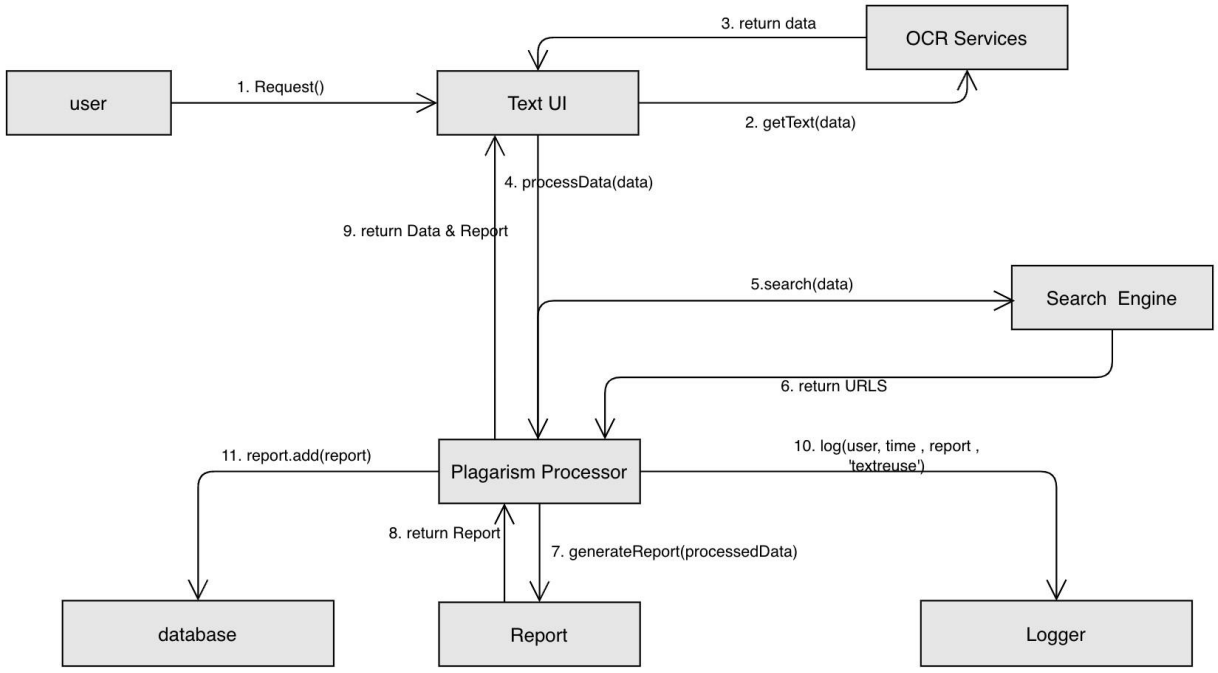


Figure 84 *Plagiarism Detection Collaboration Diagram*

3.9.15 Contact

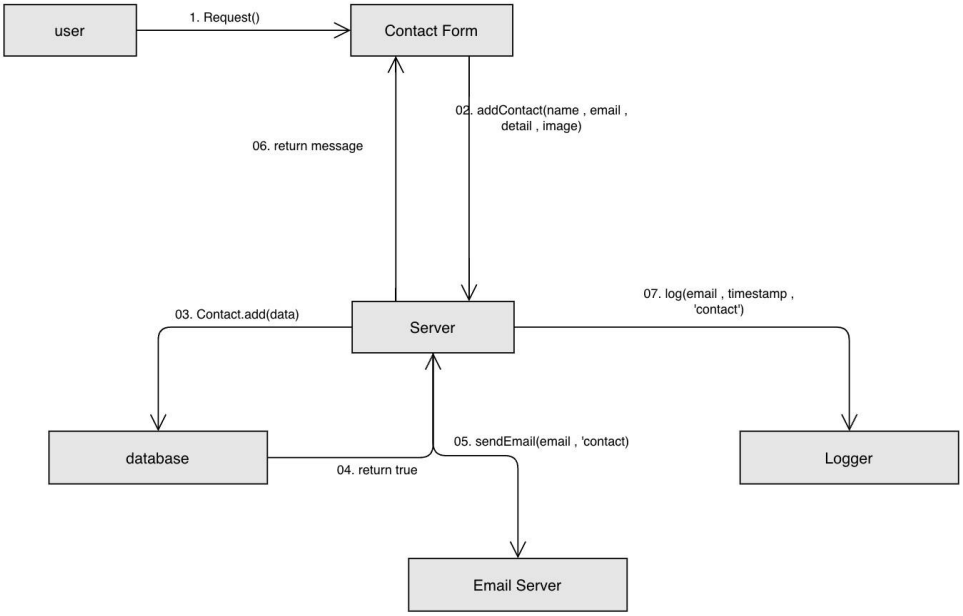


Figure 85 *Contact Collaboration Diagram*

3.9.16 Notification

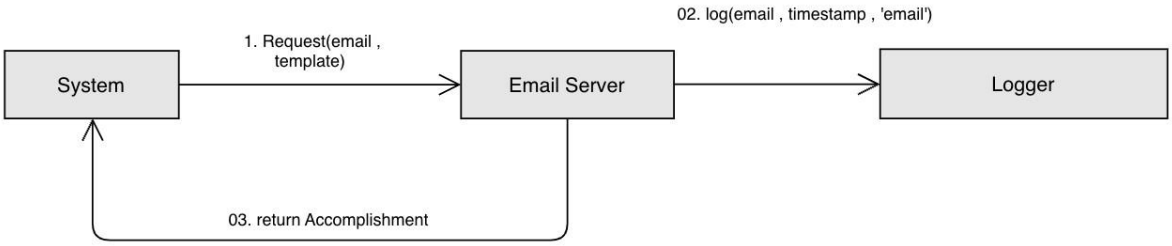


Figure 86 *Notification Collaboration Diagram*

3.9.17 Admin Dashboard

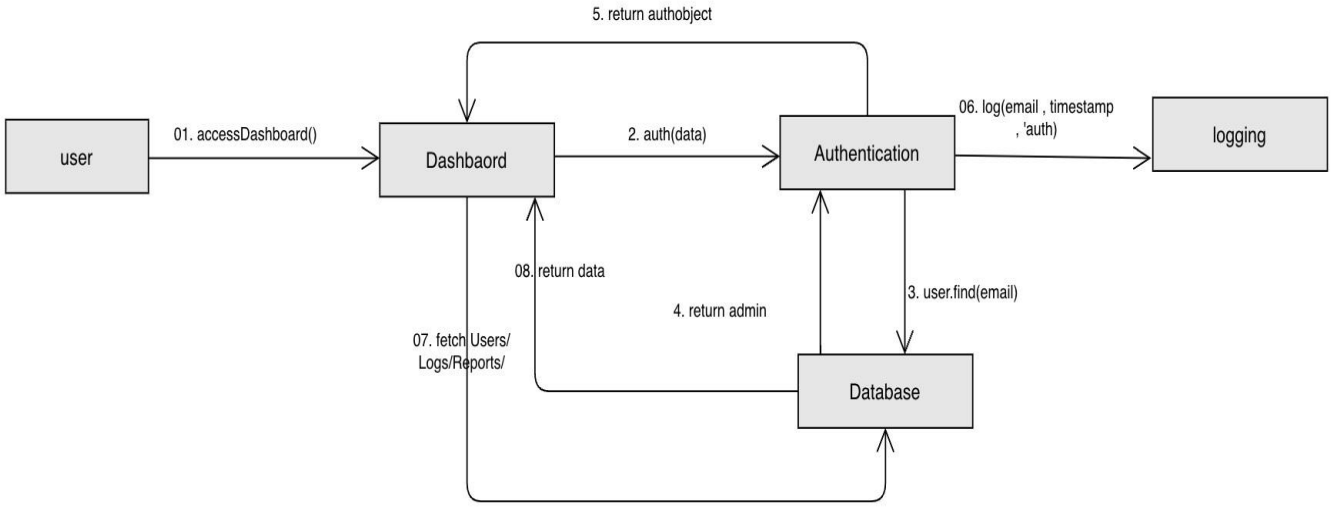


Figure 87 *Admin Dashboard Collaboration Diagram*

3.9.18 System Management

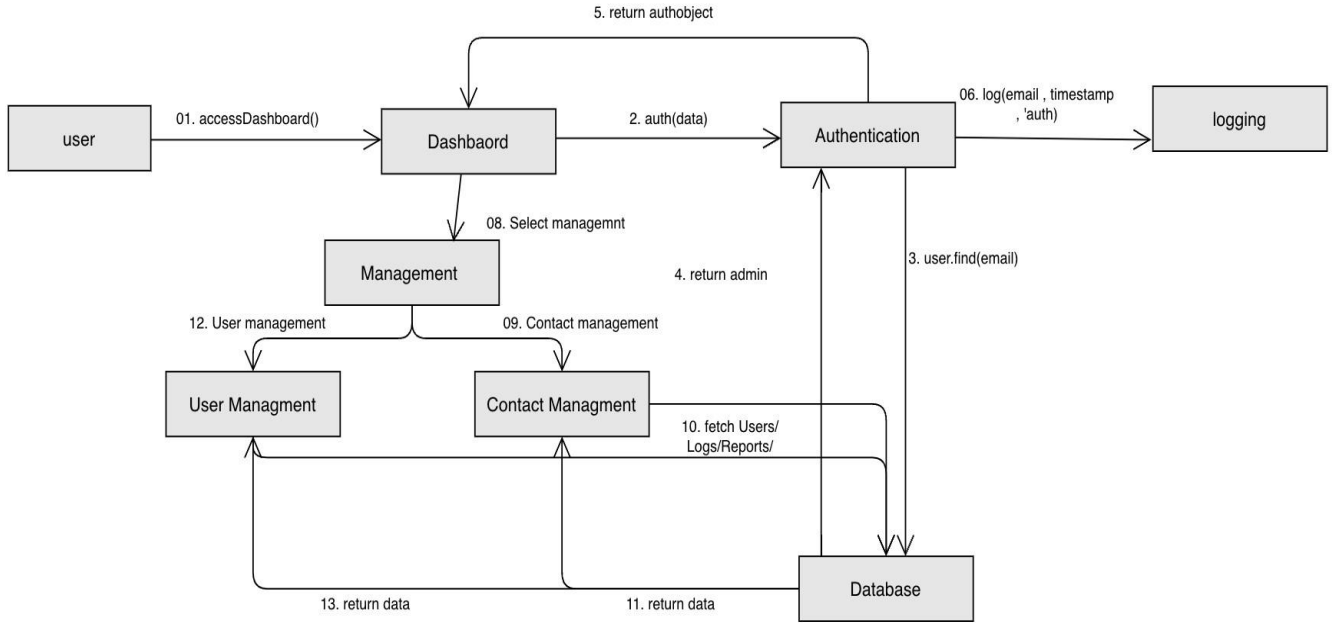


Figure 88 *System Management Collaboration Diagram*

3.9.19 Report Generation

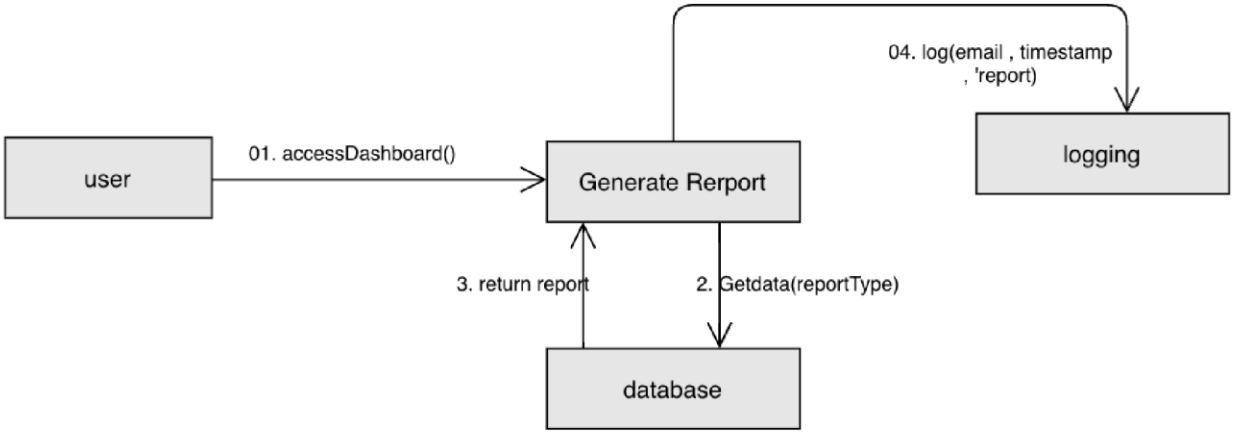


Figure 89 *Report Generation Collaboration Diagram*

3.9.20 Analytics

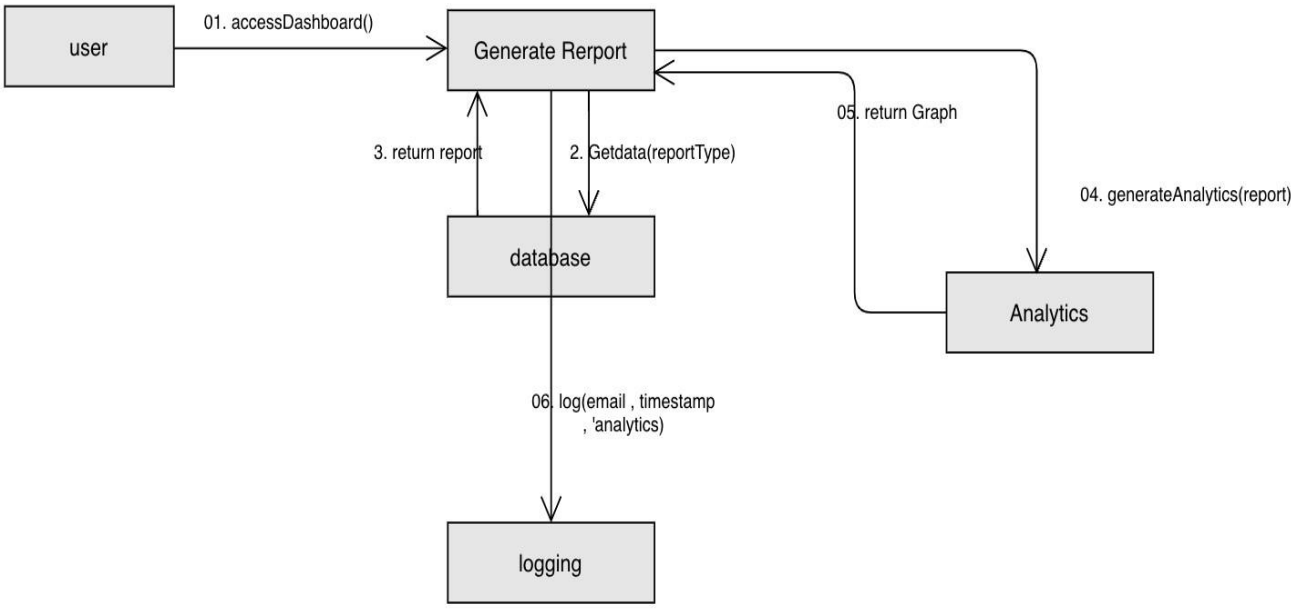


Figure 90 *Analytic Collaboration Diagram*

Chapter # 04

System Testing

4 System Testing

4.1 Test cases

The following are the test cases both valid and invalid for our final year project:

4.1.1 Login

Table 53 *Test Case 01 Login*

Field	**TC01-01 (Valid Login Test)**	**TC01-02 (Invalid Login Test)**
Test Case ID	TC01-01	TC01-02
Test Case Name	Valid Login Test	Invalid Login Test
Description	Test the functionality of logging in with valid credentials.	Test the system behaviour when logging in with invalid credentials.
Pre-Conditions	1. The user have a valid registered account with a valid email and password. 2. No user is currently logged into the system.	1. The user does not enter valid email and/or password. 2. No user is currently logged into the system.
Steps	1. Navigate to the login page. 2. Enter a valid email (e.g., user@example.com) and valid password. 3. Press the submit button.	1. Go to the login page. 2. Enter an wrong email format (for example rand#example.in) or incorrect password. 3. Press the submit button.
Expected Results	1. System validates the email format and credentials. 2. Displays a success notification. 3. Logs the attempt (timestamp, IP). 4. Redirects the user to the system home page.	1. The system validates the email format and flags errors (e.g., "Invalid email format"). 2. Displays an error message. 3. Highlights invalid fields. 4. User remains on the same login page.
Post-Conditions	The user is sign in into the system.	The user is not sign in into the system. If multiple failures occur, the system suggests using the "Forgot Password" feature.

4.1.2 Create Account

Table 54 *Test Case 02 Create Account*

Field	TC02-01 (Valid Account Creation Test)	TC02-02 (Invalid Account Creation Test)
Test Case ID	TC02-01	TC02-02
Test Case Name	Valid Account Creation Test	Invalid Account Creation Test
Description	Test the functionality of creating an account with valid inputs and a successfully verified OTP.	Test the system behaviour when creating an account with invalid inputs (e.g., duplicate email) or incorrect OTP.
Pre-Conditions	1. The user is not sign in into the system. 2. The user does not have an existing account with the email or phone number provided. 3. Email and phone number formats are correct.	1. The user is not sign in into the system. 2. The input email is already registered with the system. 3. OTP entered by the user is incorrect or expired.
Steps	1. Access the account creation page. 2. Enter valid name, email, password, date of birth, phone number, and user type. 3. Submit the form. 4. Enter the correct OTP for verification.	5. Access the account creation page. 6. Enter a duplicate email or invalid format phone number. 7. Submit the form. 8. Enter an incorrect or expired OTP for verification.
Expected Results	1. The system validates input fields. 2. Generates and sends a 6-digit OTP to the user's email. 3. User enters the correct OTP. 4. Accounts are created, and the user is redirected to the login page.	1. System displays an error for duplicate email or invalid phone number and highlights the relevant field. 2. Displays an error for incorrect OTP, allowing retries.
Post-Conditions	The user account is successfully created and stored in the database.	The user remains on the account creation page and can correct inputs or retry OTP verification.

4.1.3 Forget Password

Table 55 *Test Case 03 Forget Password*

Field	TC03-01 (Valid Password Reset Test)	TC03-02 (Invalid Password Reset Test)
Test Case ID	TC03-01	TC03-02
Test Case Name	Valid Password Reset Test	Invalid Password Reset Test
Description	Test the functionality of resetting a password with valid email, correct OTP, and a valid new password.	Test the system behaviour when resetting a password with an invalid email, incorrect OTP, or a new password which is not according to the complexity requirements.
Pre-Conditions	1. The user is not logged into the system. 2. The email provided is registered in the system. OTP is generated and sent to the registered email.	1. The user is not logged into the system. The 2. email provided is not registered in the system or OTP entered is incorrect/expired. 3. New password is invalid.
Steps	1. Navigate to the "Forgot Password" from the login page. 2. Enter a registered email. 3. Enter the correct OTP. Enter 4. and confirm a valid new password. 5. Submit the form.	1. Navigate to the "Forgot Password" from the login page. 2. Enter an unregistered email or incorrect email format. 3. Enter an incorrect or expired OTP. 4. Enter an invalid new password. 5. Submit the form.
Expected Results	1. The system validates email and OTP. 2. System prompts for a new password. 3. User enters a valid, complex password. 4. Password is reset successfully, and user will be navigated to the login page.	1. The system displays an error message for invalid email, incorrect OTP, or invalid new password. 2. User remains on the reset page and can correct inputs or retry OTP.
Post-Conditions	The user's password field is updated in the database, and the user successfully redirected to the login page with a successful message.	The password reset fails. The user stays on the password reset page and can retry with corrected inputs.

4.1.4 Change Password

Table 56 *Test Case 04 Change Password*

Field	TC04-01 (Valid Password Change Test)	TC04-02 (Invalid Password Change Test)
Test Case ID	TC04-01	TC04-02
Test Case Name	Valid Password Change Test	Invalid Password Change Test
Description	Test the functionality of changing the password with the correct current password and a valid, complex new password that is different from the old one.	Test the system behaviour when the current password is wrong, or the new password input by user doesn't meet complexity requirements.
Pre-Conditions	1. The user is logged into the system. 2. The user knows their current password.	1. The user is sign in into the system. 2. The user enters an incorrect current password or a new password that does not meet complexity requirements.
Steps	1. Redirect to the profile page and select "Change Password". 2. Enter the correct current user password. 3. Enter and confirm a valid new password. 4. Submit the form.	1. Redirect to the profile page and select "Change Password" option. 2. Enter an wrong current password. 3. Enter a weak or invalid new password. 4. Submit the form.
Expected Results	1. System validate and verify the current password of user. 2. System will check the new password is according to the complexity requirements and is different from the old password. Password is successfully updated, all sessions are terminated, and the user is notified of the successful change.	1. The system will output an error message showing incorrect current password or invalid new password. 2. User remains on the change password page to retry.
Post-Conditions	The user's password is updated in the database, all active sessions are terminated, and the user is notified of the successful password change.	The password change fails. The user stays on the change password page and can retry with corrected inputs.

4.1.5 Edit Profile

Table 57 *Test Case 05 Edit Profile*

Field	TC05-01 (Valid Profile Update Test)	TC05-02 (Invalid Profile Update Test)
Test Case ID	TC05-01	TC05-02
Test Case Name	Valid Profile Update Test	Invalid Profile Update Test
Description	Test the functionality of updating the user's profile with valid information, including a valid phone number and profile picture.	Test the system behaviour when invalid information is provided, such as an incorrect phone number or invalid profile picture format.
Pre-Conditions	1. The user is sign in into the system. 2. The user has an existing profile with updatable fields.	1. The user is sign in into the system. 2. The user attempts to update the profile with incorrect or invalid data (e.g., wrong phone number format, invalid image).
Steps	1. Navigate to the profile management section. 2. View the current profile information. 3. Click on the update button. 4. Input valid information (e.g., phone number, profile picture). 5. Submit the changes.	1. Navigate to the profile management section. View 2. the current profile information. 3. Click on the update button. 4. Input invalid data (e.g., incorrect phone number format, invalid image). 5. Submit the changes.
Expected Results	1. System validates the input fields. 2. Profile is successfully updated in the database. 3. A notification email is sent. The activity is logged.	1. System displays error messages for invalid inputs (e.g., incorrect phone number format, invalid image file). 2. The user remains on the profile update page to correct the data.
Post-Conditions	The user's profile detail is successfully updated, and related notification is sent about the update.	The user's profile information is not updated, and error messages are displayed. The user remains on the profile update page to fix the issues.

4.1.6 API Services

Table 58 *Test Case 06 API Services*

Field	TC06-01 (Valid API Request Test)	TC06-02 (Invalid API Request Test)
Test Case ID	TC06-01	TC06-02

Test Case Name	Valid API Request Test	Invalid API Request Test
Description	Test the functionality of sending a valid API request with correct data and API key to process text reuse detection and plagiarism detection.	Test the system's behaviour when the API request is invalid due to incorrect API key or exceeding rate limits.
Pre-Conditions	1. The external platform has a valid API key. 2. The external platform complies with rate limiting policies. 3. The user has access to the API documentation.	1. The external platform sends a request with an invalid API key or exceeds the rate limit for requests. 2. The user does not comply with rate limiting policies.
Steps	1. External platform sends a request to the API with valid text data and API key. 2. System authenticates the API key. 3. System processes the request using AI models for text reuse detection. 4. System returns the results to the external platform. 5. System logs the API request and response.	1. External platform sends a request to the API with an invalid API key or exceeds rate limits. 2. System detects invalid API key or rate limit issue. 3. System returns an authentication error or rate limit exceeded error. 4. System logs the error and provides an appropriate error response.
Expected Results	1. System processes the text reuse detection request successfully and returns processed results. 2. The system logs the API request and response with a timestamp and user identifier.	1. System returns an error message indicating the invalid API key or rate limit issue. 2. The system logs the error with a timestamp and user identifier.
Post-Conditions	The external platform receives the processed results of the text reuse detection and plagiarism detection.	The external platform receives an error message explaining the reason for the failure, such as invalid API key or rate limit exceeded.

4.1.7 Rate Limit for API

Table 59 *Test Case 07 Rate Limiting*

Field	**TC07-01 (Valid Rate Limit Test)**	**TC07-02 (Invalid Rate Limit Test)**
Test Case ID	TC07-01	TC07-02
Test Case Name	Valid Rate Limit Test	Invalid Rate Limit Test

Description	Test the functionality of the rate limiting system by making API requests within the allowed rate limit.	exc the system's response when the rate limit is eeded and the system throttles the API request.
Pre-Conditions	1. API user has a valid API key. 2. API user is within the rate limit for API requests.	1. API user has a valid API key. 2. API user exceeds the rate limit for API requests.
Steps	1. API user sends a request to the API. 2. System checks the number of requests made by the user within the current time frame. 3. System processes the request because it is within the rate limit. 4. System logs the event.	1. API user sends a request to the API that exceeds the allowed rate limit. 2. System checks the number of requests made by the user and detects the limit has been exceeded. 3. System throttles the request and returns a rate limit exceeded error message. 4. System logs the event.
Expected Results	System processes the API request successfully and logs the event with a timestamp and user identifier.	1. System throttles the request and returns a "rate limit exceeded" error message. The 2. system logs the error with a timestamp and user identifier.
Post-Conditions	The API request is processed successfully, and the user is within the rate limit for future requests.	The API request is throttled, and the user is notified about exceeding the rate limit. The user is prevented from making additional requests within the time frame.

4.1.8 Authorization for API

Table 60 *Test Case 08 Authorization*

Field	**TC08-01 (Valid Authorization Test)**	**TC08-02 (Invalid Authorization Test)**
Test Case ID	TC08-01	TC08-02
Test Case Name	Valid API Authorization	Invalid API Authorization
Description	Test the system's handling of valid API keys and authorized access to API services.	Test the system's response when the API key is missing, invalid, or expired, leading to unauthorized access.

Pre-Conditions	1. API user has a valid API key. 2. API user includes the API key in the request header or parameters.	1. API user has either an invalid or expired API key or is missing the API key in the request.
Steps	1. API user includes the valid API key in the request. 2. System retrieves and verifies the API key. 3. System grants access to the requested service. 4. System processes the request and returns a valid response. 5. System logs the authorization attempt.	1. API user omits the API key, or the key is invalid or expired. 2. System retrieves and verifies the API key, detects the error. 3. System denies access and returns an authentication error. 4. System logs the failed authorization attempt.
Expected Results	System grants access to the requested service, processes the request, and logs the successful authorization attempt with a timestamp and user identifier.	System denies access to the requested service, returns an authentication or authorization error message, and logs the failed attempt with a timestamp and user identifier.
Post-Conditions	API user gains access to the requested service, and the authorization attempt is logged.	API user is denied access, and the error is logged. The system prevents further access until a valid API key is provided.

4.1.9 API logging

Table 61 *Test Case 09 API logging*

Field	**TC09-01 (Valid API Logging Test)**	**TC09-02 (Invalid API Logging Test)**
Test Case ID	TC09-01	TC09-02
Test Case Name	Valid API Logging	Invalid API Logging
Description	Test the system's ability to log API requests, responses, and activities when the logging system is functioning correctly.	Test the system's behaviour when the logging service is unavailable or encounters an issue (e.g., storage issues, system failure).
Pre-Conditions	1. API user has access to API services. 2. System has logging capabilities enabled.	1. API user has access to API services, but logging service is unavailable or encounters a failure.

Steps	1. API user sends a request to the API. 2. System processes the request and generates a response. 3. System logs the request details (timestamp, API key, request parameters, response status).	1. API user sends a request to the API. 2. System processes the request and generates a response. 3. Logging service is unavailable or fails (e.g., storage issue). 4. System alerts the admin and retries logging.
	4. System securely stores the log. 5. Admin accesses the logs.	
Expected Results	The system logs the request, response, and relevant details securely. Admin can access the logs for review.	The system queues the logs or retries logging once the service is restored. Admin receives an alert, and logs are not immediately recorded.
Post-Conditions	All API interactions are recorded and stored securely. Admin can access detailed logs for analysis.	Logs are queued for later processing, and admin is notified of the logging failure. The system attempts to log once the issue is resolved.

4.1.10 User History

Table 62 *Test Case 10 User History*

Field	**TC10-01 (Valid User History Test)**	**TC10-02 (Invalid User History Test)**
Test Case ID	TC10-01	TC10-02
Test Case Name	Valid User History Tracking	Invalid User History Tracking
Description	Test the system's ability to track and store user history correctly, allowing the user to view their actions and API calls from their dashboard.	Test the system's behaviour when it fails to record an action due to system error or insufficient permissions.
Pre-Conditions	1. User is authenticated and logged into the system. 2. System has tracking capabilities enabled.	1. User is authenticated and logged into the system. 2. System tracking capabilities are enabled, but there is a failure in recording actions (e.g., system error).

Steps	1. User performs an action or makes an API call. 2. System records the action details (timestamp, action type, parameters). 3. History is stored in the user's profile. 4. User accesses their history via their dashboard.	1. User performs an action or makes an API call. 2. System fails to record the action due to a system error (e.g., storage issue). 3. System notifies the user about the failure.
Expected Results	The action is logged successfully and stored in the user's history. User can access the history via their dashboard.	The system notifies the user of the failure to record the action and offers an option to retry or report the issue.
Post-Conditions	User's actions and API calls are successfully recorded and can be reviewed in the dashboard.	User is informed of the failure to record their action, and they can retry or report the issue to the admin.

4.1.11 API Key Management

Table 63: Test Case 11 API Key Management

Field	**TC11-01 (Valid API Key Management Test)**	**TC11-02 (Invalid API Key Management Test)**
Test Case ID	TC11-01	TC11-02
Test Case Name	Valid API Key Management	Invalid API Key Management
Description	Test the system's ability to allow a user to generate and manage API keys, including creating multiple keys, regenerating keys, and downloading them for backup.	Test the system's behaviour when a user attempts to generate more API keys than allowed or when the API key generation service is unavailable.
Pre-Conditions	1. User is sign in into the system. User 2. has the appropriate permissions to manage API keys.	1. User is logged into the system. User 2. attempts to generate more API keys than allowed, or the API key generation service is unavailable.

Steps	1. User redirects to the API key management section. 2. User chooses to generate a new API key. 3. System generates a unique API key. 4. User can create multiple API keys. 5. User downloads the API key in CSV or PDF format. 6. System logs the API key management event.	1. User navigates to the API key management section. 2. User attempts to generate more API keys than allowed (or service is unavailable). 3. System displays an error message or alerts the user about the issue.
Expected Results	System successfully generates and allows the user to download the API key. The event is logged.	System alerts the user with an error message (e.g., "API key limit exceeded" or "Service unavailable"). The user is given an option to retry or contact support.
Post-Conditions	API key(s) generated, downloaded, and logged successfully.	User is informed of the failure and can retry the action as necessary.

4.1.12 API Monitoring

Table 64 *Test Case 12 API Monitoring*

Field	**TC12-01 (Valid API Monitoring Test)**	**TC12-02 (Invalid API Monitoring Test)**
Test Case ID	TC12-01	TC12-02
Test Case Name	Valid API Monitoring and Quota Enforcement	Invalid API Monitoring (Failure in Quota Enforcement)
Description	1. Tests the system's ability to provide valid API usage analytics and enforce quota limits correctly, including notifying the user and suspending their access if exceeded.	2. Tests the system's behaviour when there is a failure in quota enforcement, such as the suspension of API services not occurring even if the quota is exceeded.
Pre-Conditions	1. User is logged in with an active account. 2. Admin is logged in with access to the monitoring dashboard.	3. User is logged in with an active account. 4. Admin is logged in with access to the monitoring dashboard. 5. The system's monitoring service is functioning correctly.

Steps	1. User accesses the API monitoring dashboard. 2. System displays usage analytics, including total API calls and remaining quota. 3. Admin accesses the dashboard and selects the user to monitor. 4. System displays detailed API usage statistics. 5. Admin reviews the analytics. If 6. the user exceeds the quota, the system suspends their access and sends an email notification. 7. System logs all monitoring activities.	1. Admin accesses the dashboard and selects the user. 2. Admin reviews usage statistics. User 3. exceeds the quota, but the system fails to suspend the service. The 4. system does not send an email notification.
Expected Results	1. The system displays accurate usage statistics and successfully enforces quota limits by suspending the user's access and sending an email notification.	1. The system fails to suspend the user's access after exceeding the quota and does not send an email notification, indicating a failure in quota enforcement.
Post-Conditions	API usage statistics displayed correctly. User's access suspended if quota exceeded and email sent. Monitoring activities logged.	User's access is not suspended, and no email notification is sent despite exceeding the quota.

4.1.13 Text Reuse Detection

Table 65 *Test Case 13 Text Reuse Detection*

Field	**TC13-01 (Valid Reuse Detection Test)**	**TC13-02 (Invalid Reuse Detection Test)**
Test Case ID	TC13-01	TC13-02
Test Case Name	Valid Reuse Detection and Report Generation	Invalid Reuse Detection (OCR Failure / Unsupported File Format)
Description	Tests the system's ability to detect text reuse accurately from manually inputted text or uploaded documents/images, process it using AI models, and generate a report.	Tests the system's behaviour when an uploaded image cannot be processed due to OCR failure or an unsupported file format, preventing reuse detection and report generation.

Pre-Conditions	1. User is logged in. 2. User has access to the reuse detection tool. 3. System has necessary AI models and OCR capabilities.	1. User is logged in. 2. User has access to the reuse detection tool. 3. The uploaded file is in an unsupported format or the OCR fails to process the text.
Steps	1. User navigates to the reuse detection tool. 2. User inputs two Urdu paragraphs manually or uploads a valid document/image. 3. OCR processes the text from the image (if applicable). 4. System displays extracted text. 5. User initiates reuse detection. System 6. processes texts using AI models. System calculates the reuse percentage. 7. System generates a report. 8. User downloads the report. 9.	1. User navigates to the reuse detection tool. 2. User uploads an unsupported file type (e.g., .docx, .jpg). 3. System prompts for a compatible format. 4. User uploads a clearer image, but OCR still fails to extract text. 5. The system shows an OCR failure notification and suggests uploading a clearer image.
Expected Results	1. The system successfully processes the texts and generates a detailed report on reused content with percentages, categories, and sources. User can download the report.	1. The system fails to process the uploaded file or text, displays an error or warning message about unsupported file format or OCR failure, and provides suggestions for resolution.
Post-Conditions	Reuse detection report is generated and available for download.	User is informed of the failure (either due to unsupported file format or OCR
		failure) and is prompted to retry or correct the issue.

4.1.14 Plagiarism Detection

Table 66 *Test Case 14 Plagiarism Detection*

Field	**TC14-01 (Valid Plagiarism Detection Test)**	**TC14-02 (Invalid Plagiarism Detection Test)**
Test Case ID	TC14-01	TC14-02
Test Case Name	Valid Plagiarism Detection and Report Generation	Invalid Plagiarism Detection (Unsupported File Format)

Description	Tests the system's ability to detect plagiarism from an uploaded text file containing Urdu content, process the text using online sources, and generate a report.	Tests the system's behaviour when an unsupported file type is uploaded, preventing plagiarism detection and report generation.
Pre-Conditions	1. User is logged in. 2. User has access to the plagiarism detection tool. 3. System has access to online search engines and databases for comparison.	1. User is logged in. 2. User has access to the plagiarism detection tool. 3. The uploaded file is in an unsupported format (e.g., .docx, .jpg).
Steps	1. User navigates to the plagiarism detection tool. 2. User uploads a text file with Urdu content in a supported format. System 3. extracts the text from the uploaded file. 4. System compares the extracted text with online sources. 5. System calculates the plagiarism percentage. 6. System identifies source URLs. 7. System generates a plagiarism report. 8. User downloads the report.	1. User navigates to the plagiarism detection tool. 2. User uploads an unsupported file format (e.g., .docx, .jpg). 3. System notify the user to upload a valid and accordant file.
Expected Results	1. The system processes the text, calculates the plagiarism percentage, retrieves the source URLs, generates a report, and allows the user to download the report.	1. The system output an error message popup box informing the user about the unsupported file format and prompts the user so it can upload a compatible file.
Post-Conditions	The plagiarism detection report is generated, and the user can download it.	The system prompts the user to upload a compatible file, and the plagiarism detection process is halted.

4.1.15 Contact & Query Form

Table 67 *Test Case 15 Contact & Query Form*

Field	**TC15-01 (Valid Contact Form Submission)**	**TC15-02 (Invalid Contact Form Submission - Missing Category)**
Test Case ID	TC15-01	TC15-02
Test Case Name	Valid Contact Form Submission and Query Handling	Invalid Contact Form Submission - Missing Message Category

Description	Tests the system's ability to successfully submit a query or complaint with all required fields, categorize the message, send it to the admin, track status, and notify the user.	Tests the system's behaviour when a user tries to submit a query without selecting a message category, which is a required field. The system should prompt the user to select one.
Pre-Conditions	1. User is logged in (optional depending on design). 2. User has access to the contact form tool.	1. User is logged in (optional depending on design). 2. User has access to the contact form tool.
Steps	1. User navigates to the contact form. 2. User fills in the query or complaint details. 3. User selects a message category. 4. User submits the form. 5. System sends the query to the admin. 6. Admin reviews and responds to the query. 7. User views their submitted query and admin response. 8. User tracks the query status. System 9. sends email notifications to the user on status updates.	1. User navigates to the contact form. 2. User fills in the query or complaint details. 3. User does not select a message category. 4. User attempts to submit the form. 5. System prompts the user to select a category.
Expected Results	The query is successfully submitted, sent to the admin, status is tracked, and the user is notified of updates.	The system notify the user to select a category and does not allow form submission until the user selects a message category.
Post-Conditions	The query is successfully submitted, tracked, and responded to by the admin. User receives email notifications as per status updates.	The system does not allow the query to be submitted until the user selects a category, and the user is notified of the error.

4.1.16 Notifications

Table 68 *Test Case 16 Notifications*

Field	**TC16-01 (Valid Notification)**	**TC16-02 (Invalid Notification - Invalid Content)**
Test Case ID	TC16-01	TC16-02
Test Case Name	Valid Notification to User	Invalid Notification - Invalid Content

Description	Tests the system's ability to successfully send a notification when a predefined important event occurs (e.g., task completion), respecting user preferences.	Tests the system's behaviour when an invalid notification is attempted (e.g., the notification content contains prohibited information).
Pre-Conditions	1. User is registered in the system. 2. System identifies an important event. 3. Admin has permission to send notifications.	1. User is registered in the system. 2. Admin has permission to send notifications. 3. Admin is attempting to send a notification with invalid content.
Steps	1. System identifies an important event (e.g., task completion). 2. System process a notification message. 3. System shall notify user by sending the email notification to the user. User 4. receives the notification.	1. Admin creates a notification with invalid content (e.g., prohibited words or sensitive information). 2. Admin attempts to send the notification. 3. System rejects the notification and alerts the admin.
Expected Results	The notification is successfully sent, and the user receives and views the notification in their email.	The system rejects the notification due to invalid content and alerts the admin with an error message.
Post-Conditions	The user receives a timely and relevant notification about the important event. The system logs the notification activity.	The notification is not sent, and the admin is alerted to correct the content of the notification before attempting to send it again.

4.1.17 Admin Dashboard

Table 69 *Test Case 17 Admin Dashboard*

Field	**TC17-01 (Valid Admin Dashboard Access)**	**TC17-02 (Invalid Admin Dashboard Access - Insufficient Permissions)**
Test Case ID	TC17-01	TC17-02
Test Case Name	Valid Admin Dashboard Access	Invalid Admin Dashboard Access - Insufficient Permissions
Description	Tests the system's ability to grant an admin access to the dashboard and display relevant metrics, logs, and activities for management.	Tests the system's behaviour when an admin lacks sufficient permissions to access certain dashboard functionalities or data.

Pre-Conditions	1. Admin is logged into the system with appropriate administrative privileges. The system has dashboard 2. capabilities enabled and populated with relevant data.	1. Admin is logged into the system but has limited privileges (e.g., restricted access to certain dashboard features). The 2. system has dashboard capabilities enabled.
Steps	1. Admin sign in into the system. 2. Admin redirects to the admin dashboard main page. 3. System displays relevant metrics such as user stats, API usage, etc. 4. Admin interacts with the dashboard, viewing logs and performing administrative tasks.	1. Admin logs into the system with limited permissions. 2. Admin navigates to the admin dashboard section. 3. System restricts access to certain dashboard functionalities and displays an appropriate message.
Expected Results	Admin can view the dashboard with the relevant data, and interact with the widgets to manage system operations effectively.	The system restricts access to certain areas of the dashboard based on insufficient permissions, and the admin receives an appropriate error message.
Post-Conditions	Admin has comprehensive visibility into system metrics and can perform necessary administrative tasks.	Admin is informed of the access restriction and may attempt to resolve the issue by contacting support or adjusting permissions.

4.1.18 System Management

Table 70 *Test Case 18 System Management*

Field	**TC18-01 (Valid System Management - Admin Task)**	**TC18-02 (Invalid System Management - Insufficient Permissions)**
Test Case ID	TC18-01	TC18-02
Test Case Name	Valid System Management - Admin Task	Invalid System Management - Insufficient Permissions
Description	Tests the system's ability to allow an admin to access and manage user accounts, complaints, queries, and perform administrative tasks like updating settings.	Tests the system’s behaviour when an admin lacks sufficient permissions to perform certain administrative tasks, such as configuring settings or managing API keys.

Pre-Conditions	1. Admin is logged into the system with appropriate administrative privileges. The system has user and task 2. management capabilities enabled.	1. Admin is logged into the system but has limited privileges (e.g., restricted access to certain system management functionalities). 2. System has user and task management capabilities enabled.
Steps	1. Admin accesses the system respective page from the admin dashboard. Admin analyse the list of system 2. users, their details, and statuses. Admin creates, updates, or deletes 3. user accounts. Admin manages complaints and 4. queries, and performs other administrative tasks.	1. Admin accesses the system management section from the admin dashboard. 2. Admin attempts to perform an action (e.g., configuring system settings) without required permissions. The 3. system displays an error message for insufficient permissions.
Expected Results	Admin can view and manage users, complaints, and queries, and perform administrative tasks such as configuring system settings.	The system restricts access to certain tasks due to insufficient permissions and displays an appropriate error message indicating the restriction.
Post-Conditions	Admin effectively manages users, complaints, queries, and system settings.	Admin is informed of the access restriction and may attempt to resolve the issue by contacting support or adjusting permissions.

4.1.19 Report Generation

Table 71 *Test Case 19 Report Generation*

Field	**TC19-01 (Valid Report Generation - Admin Task)**	**TC19-02 (Invalid Report Generation - No Data for Filters)**
Test Case ID	TC19-01	TC19-02
Test Case Name	Valid Report Generation - Admin Task	Invalid Report Generation - No Data for Filters
Description	Tests the system's ability to allow an admin to generate reports with valid filters and export them in the desired format (CSV/PDF).	Tests the system's behaviour when an admin selects filters that result in no data being returned, such as a date range with no activity.

Pre-Conditions	1. Admin is logged into the system with appropriate administrative privileges. 2. Reporting capabilities are enabled, and the system has access to relevant data.	1. Admin is logged into the system with appropriate administrative privileges. 2. Reporting capabilities are enabled, and the system has access to relevant data.
Steps	1. Admin redirect to the report generation section in the admin dashboard. 2. Admin selects the report type (e.g., user activity, API usage). 3. Admin applies filters such as date range, specific users, or activity types. 4. Admin generates the report in CSV or PDF format. 5. Admin shall exports and download the report.	1. Admin shall redirect to the report generation section in the admin dashboard. 2. Admin selects report type (e.g., user activity). 3. Admin applies filters such as date range and activity type. 4. System retrieves no data based on selected filters. 5. System notifies the admin and suggests adjusting the filters.
Expected Results	The system generates the report and allows the admin to export it in the selected format (CSV/PDF).	The system notifies the admin that no data was found based on the selected filters and suggests adjusting the filters.
Post-Conditions	Admin successfully generates and exports the report in the desired format.	Admin is informed that no data was found and can adjust the filters or modify the report criteria accordingly.

4.1.20 Analytics Report

Table 72 *Test Case 20 Analytics Report*

Field	**TC20-01 (Valid Analytics Report Generation - Admin)**	**TC20-02 (Invalid Analytics Report Generation - Unauthorized Access)**
Test Case ID	TC20-01	TC20-02
Test Case Name	Valid Analytics Report Generation – Admin	Invalid Analytics Report Generation - Unauthorized Access
Description	Tests the system's ability to allow an admin to retrieve, filter, and export usage analytics data in a desired format.	Tests the system's response when a non-super user attempts to access analytics data without the necessary permissions.
Pre-Conditions	1. Admin is logged into the system. 2. The system has usage analytics data available.	1. User (not supers user) attempts to access the analytics section.

Steps	1. Admin navigates to the analytics section. 2. Admin selects the analytics report type (e.g., token usage, remaining quota). 3. Admin applies any desired filters (e.g., date range, user). 4. System generates the analytics report and displays the results. 5. Admin exports the report or requests the data in a formatted document.	1. Non-admin user attempts to access the analytics section. 2. System restricts access and notifies the user that they lack the necessary permissions.
Expected Results	Admin successfully retrieves the desired analytics data and exports it in the selected format.	System prevents access and shows an appropriate notification (e.g., "You do not have the necessary permissions to view analytics data.").
Post-Conditions	Admin successfully exports analytics data for further analysis.	Non-admin user is informed that they cannot access the analytics section and is restricted from further actions.

4.2 Testing

In our final year project, testing strategy is important. It ensures the reliability, efficiency, and security of our detection system. We are using a variety of testing methods in which each method is designed to a specific part of the system architecture, as described in the sections below.

4.2.1 Unit Testing

Unit testing is testing of individual components of the software which ensure that every piece of the program output as it was designed or supposed to be.

4.2.2 Black Box Testing

We test the web application by inputting various Urdu text examples to see if the system accurately detects reuse without considering the internal operations. This helps ensure that the system behaves as expected for the end-users who interact through the frontend.

4.2.3 White Box Testing

It implies testing for the inner workings or structure of our AI models as they correctly examine and compare the Urdu texts. Logic within code paths, decision statements in the backend, and integration points that would hook the AI model into the backend server are a few of those.

4.2.4 Integration Testing

Integration testing checks the connectivity and data transfer between multiple components of our system, which includes the frontend, backend, AI model server, and database server. The key areas include the following:

4.2.5 Frontend and Backend

Interaction between the user-facing frontend and the backend server to check whether user requests are properly parsed, processed, and the right results returned and shown.

4.2.6 Backend and AI Model Server

We concentrate on the data flow between these two components so that text sent for analysis is processed accurately by the AI models and that the results are correctly returned and handled by the backend.

4.2.7 Backend and Database Server

This testing ensures that the operations like storing user queries, fetching results, and managing user sessions are done without issues related to data integrity.

4.2.8 Acceptance testing

System testing is checking that the completed and fully integrated software application adheres to the stated requirements. For our system, these are:

4.2.9 Function Testing

All features of the system were tested in order to be sure they worked right, including testing of mechanisms in login, processes in submitting texts, accuracy of detection, and reporting features.

4.2.10 Performance Testing

This will be critical for our system because possibly huge volumes of text data might be analysed. It tests if the system would remain responsive and stable in heavy loads, simulating the performance by having multiple users.

4.2.11 Security Testing

Due to the sensitive nature of data and intellectual property, securing the user's data and the integrity of text analysis is paramount. We perform vulnerability scans and penetration testing to identify potential security threats and mitigate them.

4.2.12 Usability Testing

Guarantees that the interface is intuitive and user-friendly, considering that our system supports Urdu, where specific considerations about font rendering and text input are to be taken care of.

4.2.13 Compatibility Testing

This testing ensures that our web application functions across various browsers and devices, maintaining a consistent user experience regardless of user access methods.

Each testing phase is documented in detail in our report, reflecting how each contributes to the overall robustness and reliability of the Urdu text reuse detection system. These steps ensure our final year project not only meets the functional requirements but also the non-functional requirements mentioned above.

Chapter # 05

Conclusion

5 Conclusion

5.1 Problems Faced & Lessons Learned

We have faced several challenges during this final year project which are following.

5.1.1 Dataset Challenges

As, Urdu is digitally very low resource language. That's why a major problem we faced during this project is the lack of large corpus based on documents for Urdu text reuse detection and plagiarism detection. There is not even a single large scaled annotated benchmark dataset for our project available publicly on internet.

5.1.2 Language Complexity

Urdu is a linguistically rich language with unique syntactical structures (grammatical arrangement of words) and vocabulary making it challenging to detect reuse of text.

5.1.3 Computational Resources

We need high computational resources such as GPUs **[7]** for training deep learning models for the task of Urdu text reuse detection and experimentations.

5.1.4 Limited literature

Due to the unavailability of large-scaled benchmark dataset, linguistically complex language and less community of NLP researchers in Urdu, there is not much literature available for Urdu text reuse detection and plagiarism detection. So, there is very limited literature available for this final year project.

5.1.5 Lessons Learned

Through this project, we have learned state of the art technologies and methodologies of Machine Learning, Deep Learning and NLP, particularly for low-resource languages like Urdu. Key technical learnings include:

5.1.6 Machine Learning techniques

We learned how to extract features using traditional methods such as TF-IDF with cosine similarity and apply Machine Learning Algorithms for example support vector machine (SVM), Navie Bayes, KNN, decision tree, Random Forest as baseline text reuse detection models.

5.1.7 Deep Learning (DL)

We learned how to work with architectures such as recurrent neural networks (RNNs), long short-term memory (LSTM) Bi-LSTM, and convolutional neural networks (CNNs). We explored how these deep learning models capture semantic relationships in Urdu text.

5.1.8 Transformers and Large Language Models (LLMs)

We use transformer models like BERT and fine-tuning them for Urdu text reuse detection task for improvement. Experimenting multilingual models like LaBSE and fine-tuning them for low-resource language make our concept more solid about transfer learning.

5.2 Project Summary

We aim to develop 9 Urdu text reuse corpora. We will construct the corpora using back translation technique. The 5 corpora will be created by applying a 1-3-5-7-9 languages translation method. This process involves translating text from one language to other and then back to its original (e.g., Urdu to Arabic, Arabic to English, and then back in Urdu). The 4 corpora will be created by the back translating text using 4 most popular language families. These corpora will have source and reused text classified into 3 classes such as Partially Derived (PD), Wholly Derived (WD), and Non-Derived (ND).

We will also do some experiments with machine learning (ML), deep learning (DL), and large language model (LLM)-based algorithms to evaluate the performance of these 9 corpora. In addition, we will develop web-based applications which will be used detection of text reuse and also for text plagiarism detection for Urdu language. We will also develop Rest API for access to this tool, especially for academic and professional systems where they need Urdu text reuse detection.

5.3 Future Work

The following areas can be explored in the future to expand and enhance the project:

5.3.1 Dataset Expansion

Expansion of Urdu text reuse detection corpus can be considered as future work. A large dataset is very important for better training. It reduces the overfitting.

5.3.2 Cross-Language Detection

Now we are experimenting with only Urdu language for text reuse detection, but for future work we can train models for multi-lingual text reuse detection.

5.3.3 Model Optimization

Experimentation with more advanced NLP techniques and deep learning architecture which improves the performance such as accuracy and F1 score can be considered as future work of this project.

5.3.4 Mobile Application Development

In future we will make an native mobile application that supports both IOS and Android containing all the features and functionalities.

Chapter # 06

Implementation

6 Implementation

6.1 Home Page:

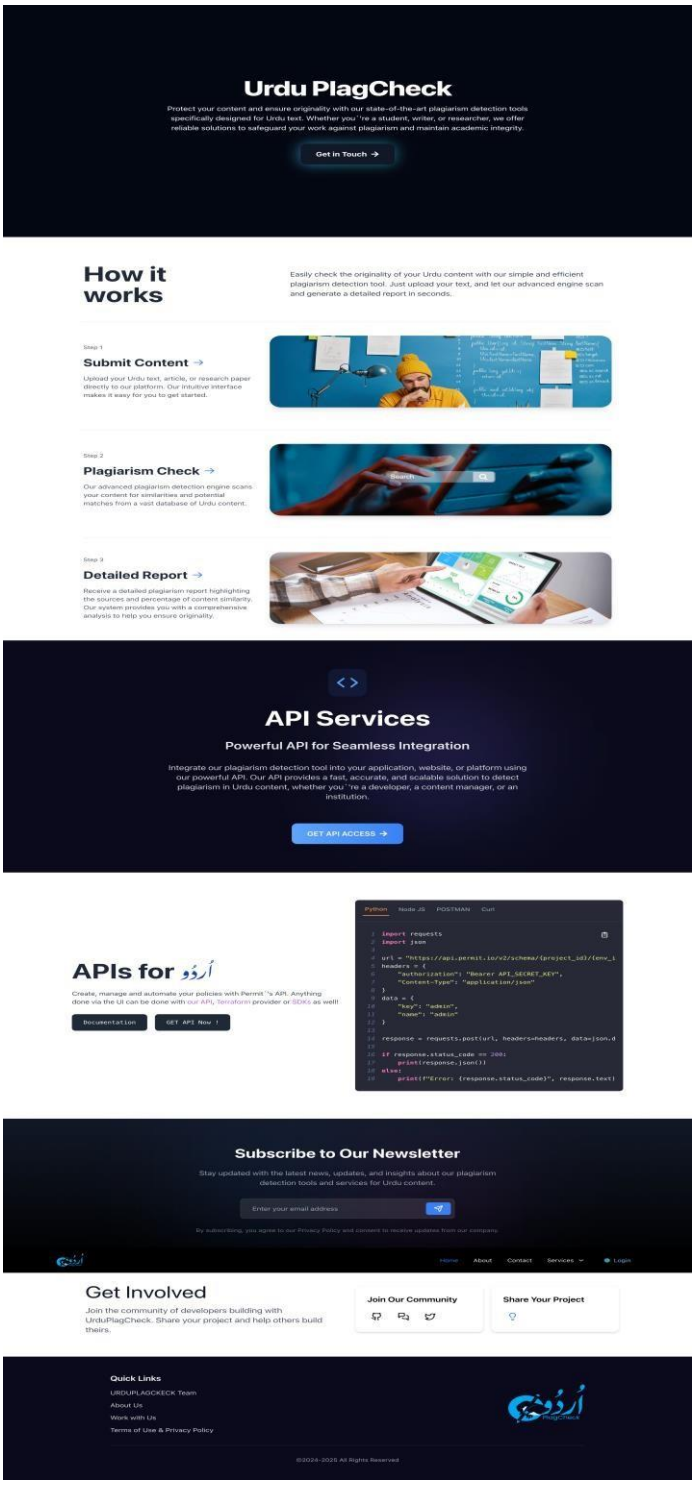


Figure 91 *Home Page*

6.2 Contact Us:

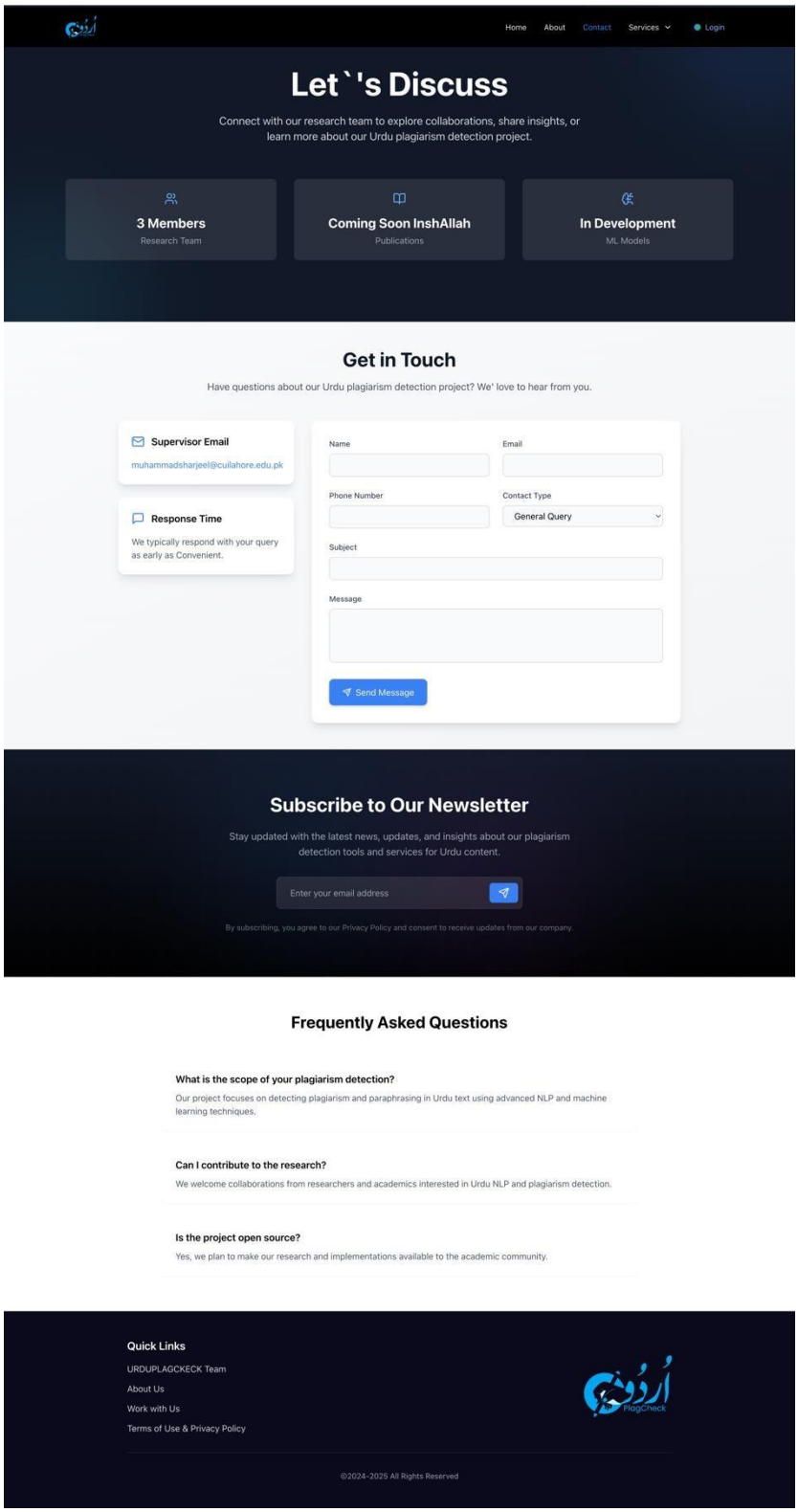


Figure 92 *Contact us page*

6.3 About:

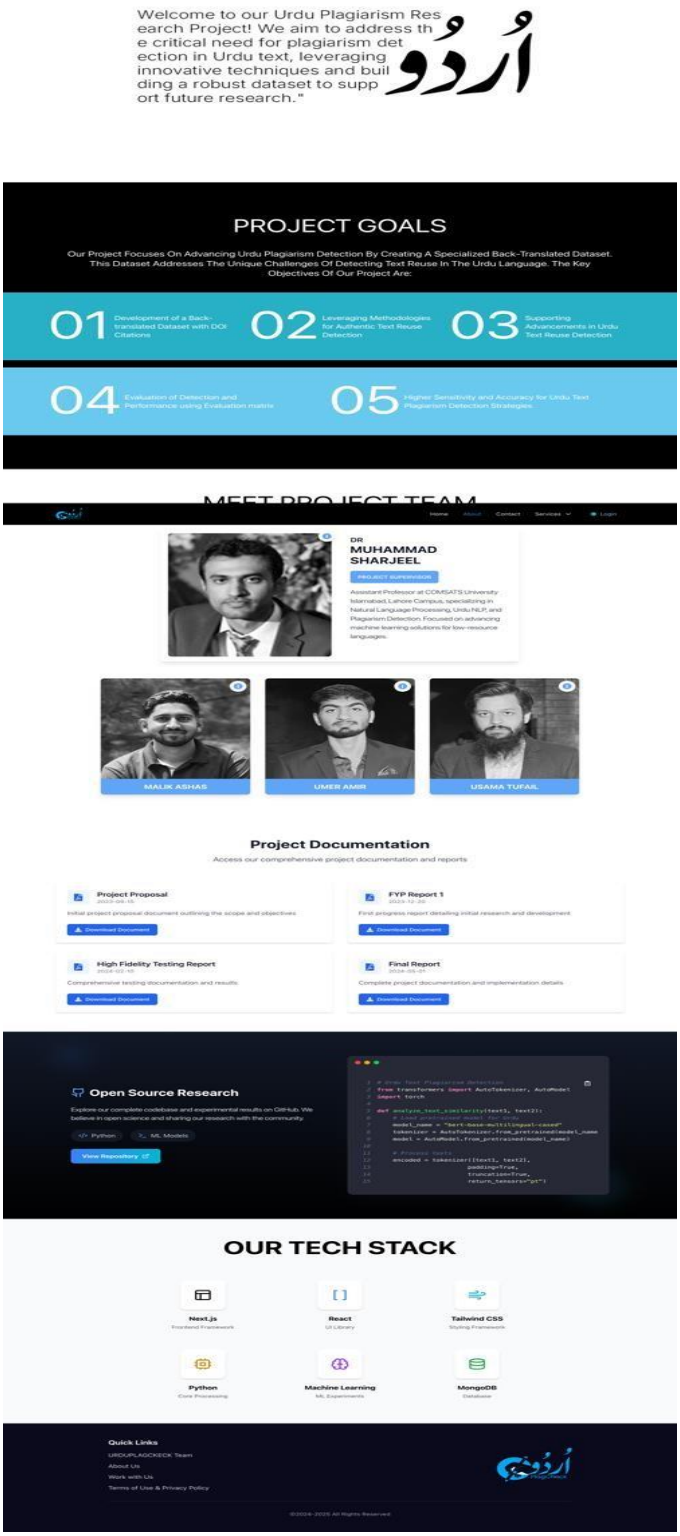


Figure 93 *About Page*

6.4 Login:

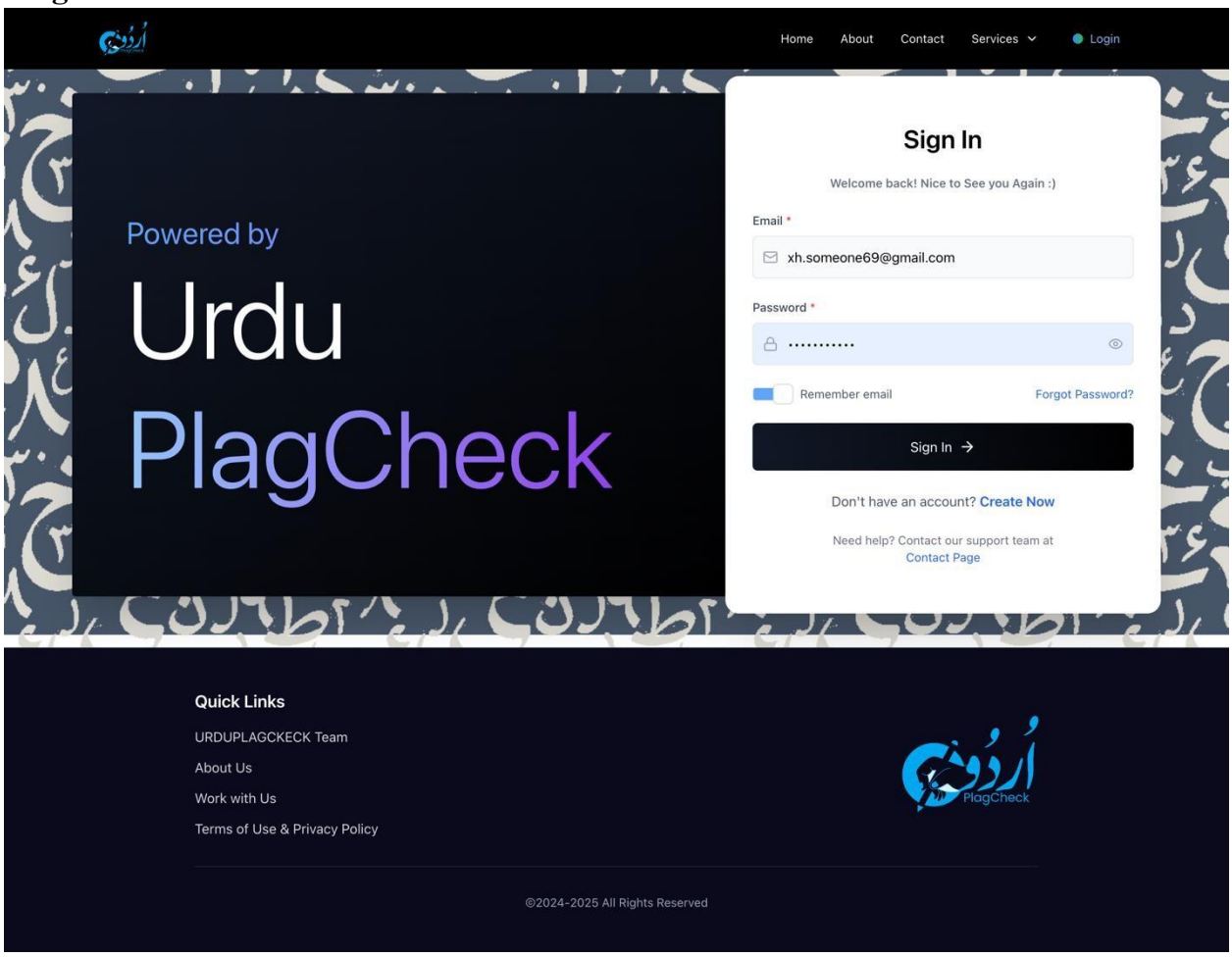


Figure 94 *Login Page*

6.5 Text Reuse Detection:

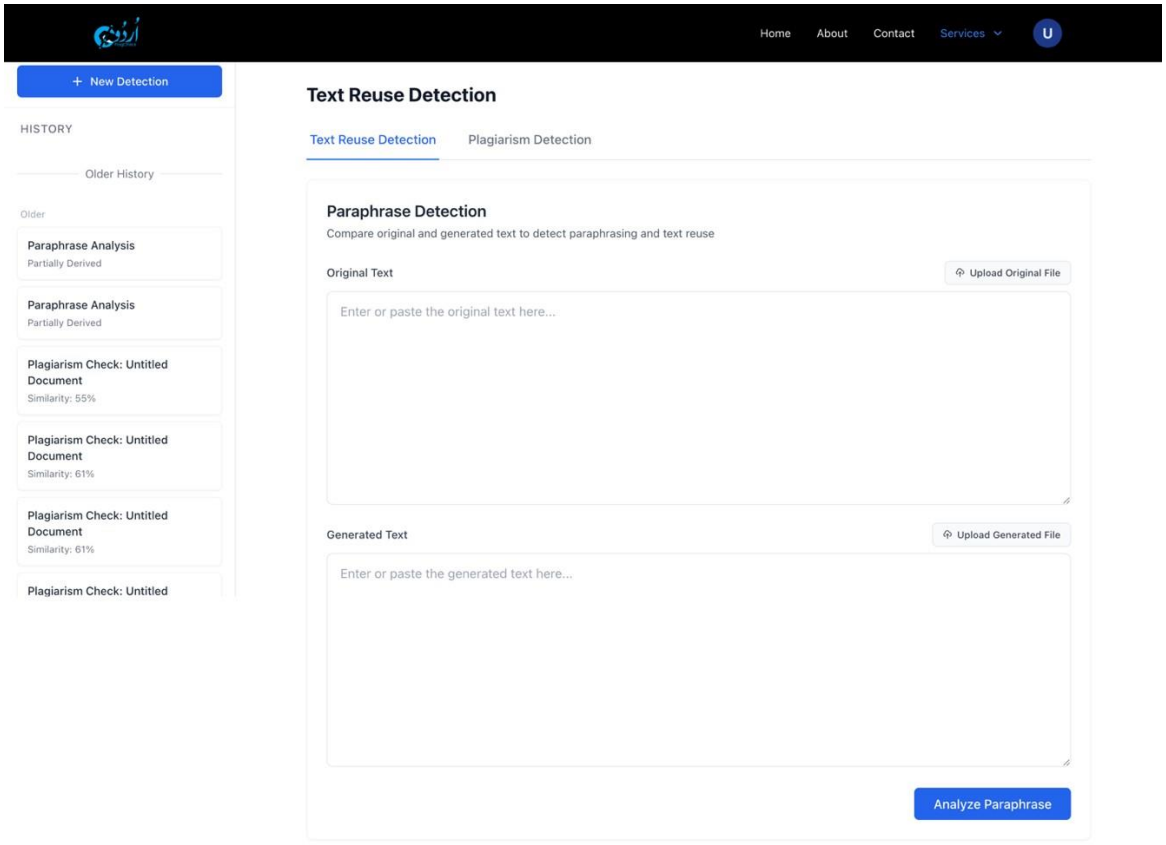


Figure 95 *Text Reuse Detection*

6.6 API Service:

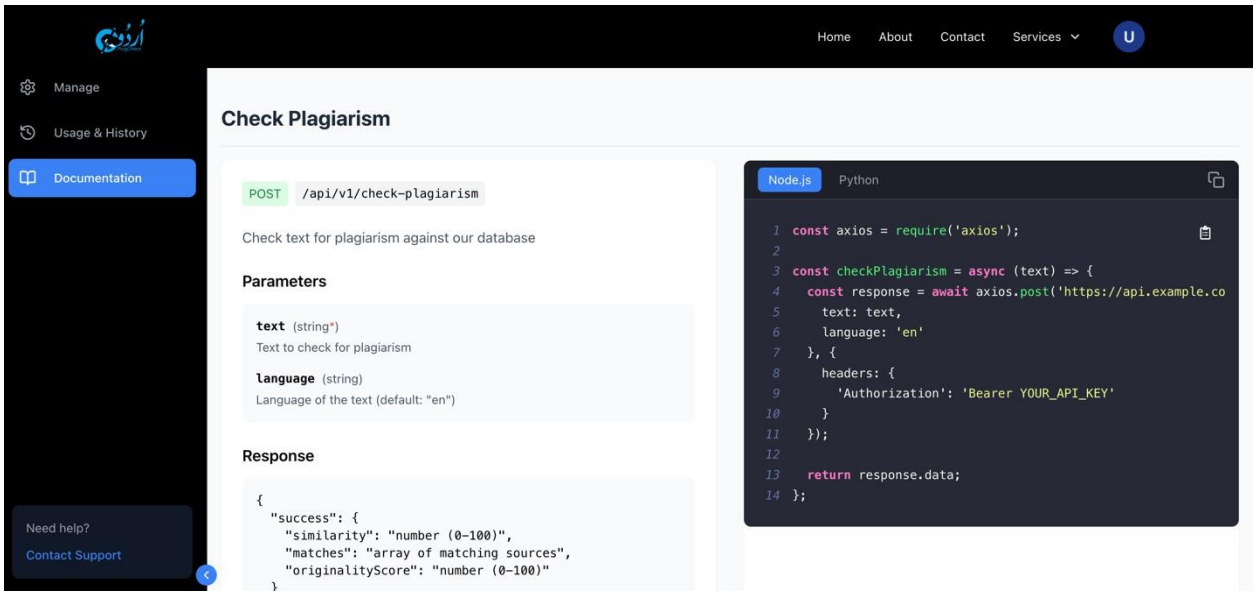


Figure 96 *API Service*

6.7 Plagiarism Detection:

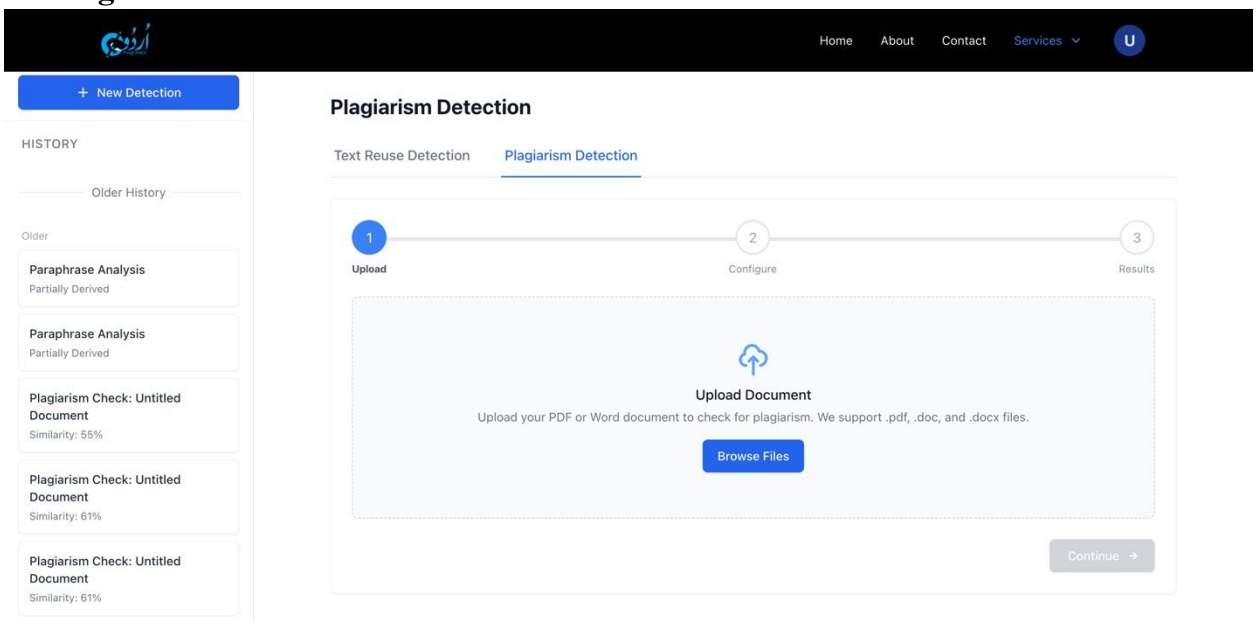


Figure 97 *Plagiarism Detection*

6.8 User Management:

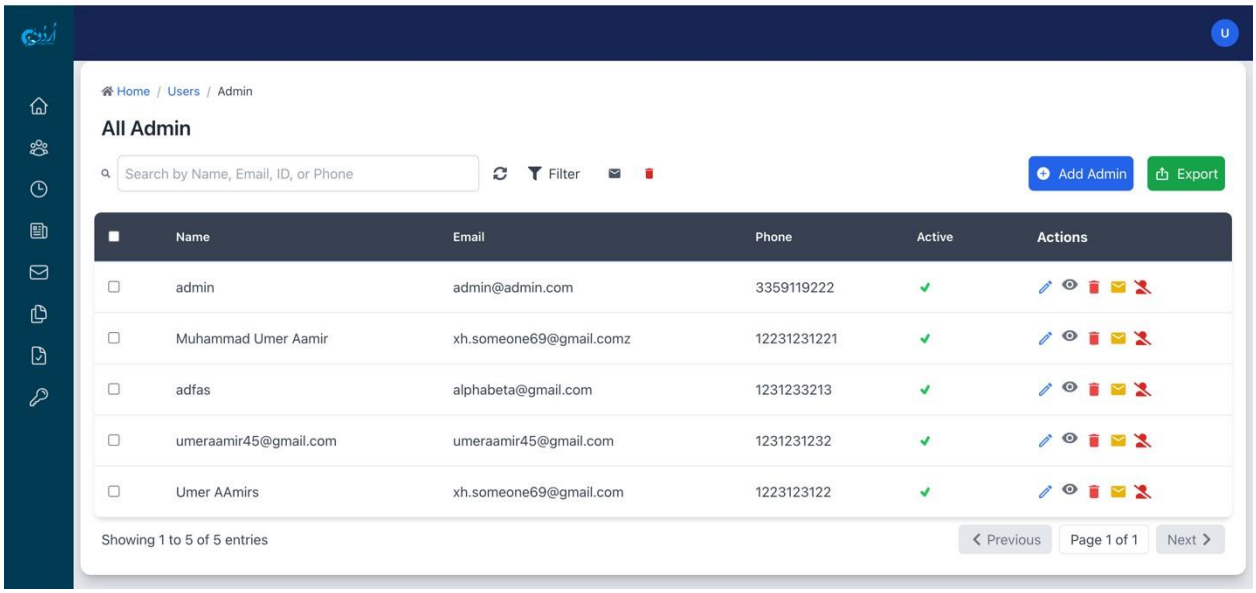


Figure 98 *User Management*

6.9 Admin Dashboard:

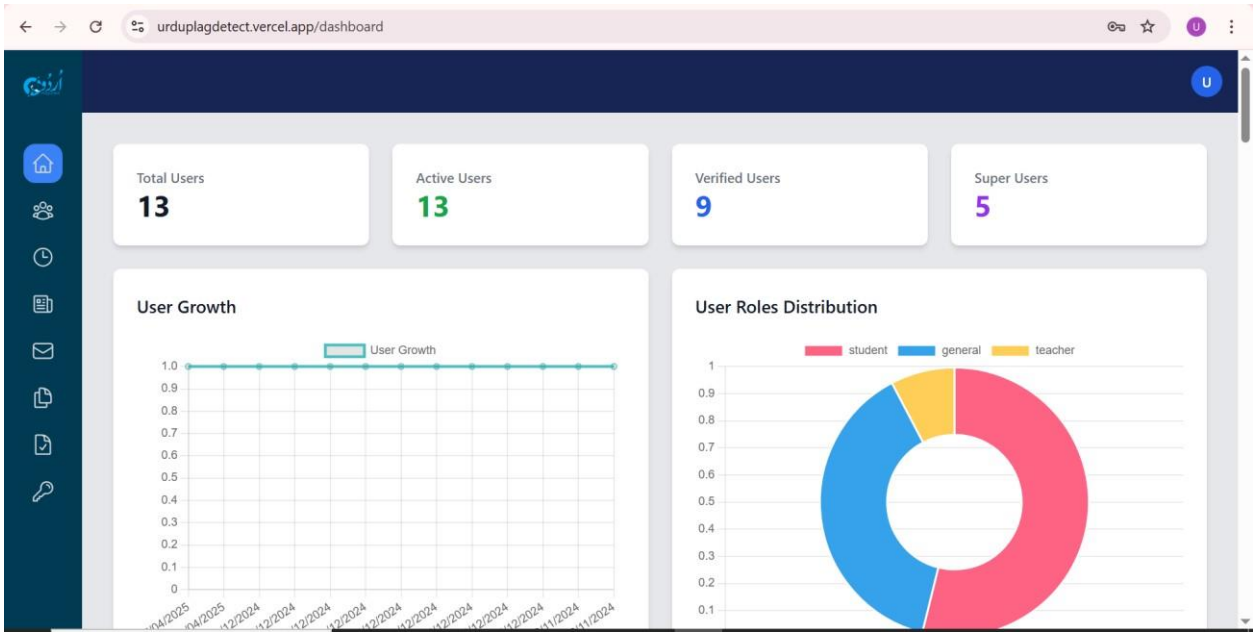


Figure 99 *Admin Dashboard*

7 References

[1] Areeba, "Medium," Medium, 19 September 2024. [Online]. Available: https://medium.com/languagelab/urdu-a-rich-mix-of-multiple-languages-80d9027d9f5e.

[2] M. A. S. N. Shameem Yousuf, "A review of plagiarism detection based on Lexical and Semantic Approach," in *International Conference on Emerging Trends in Communication, Control, Signal Processing & Computing Applications (C2SPCA)*, 2013.

[3] M. A. M. S. Usman Khan, "Urdu Natural Language Processing Issues and Challenges: A Review Study," *Springer Nature,* 2020.

[4] R. M. ,. A. A. R. ,. S.-U. H. Hafiz Rizwan Iqbal, "Urdu paraphrase detection: A novel DNN-based implementation using a semi-automatically generated corpus," *Cambridge Press,* 2023.

[5] I. M. M. S. M. A. A. R. M. A. N. Hamza Hafeez, "Urdu Short Paraphrase Detection at Sentence Level," *ACM,* 2023.

[6] M. S. R. M. A. N. P. R. Sara Sameen, "Measuring Short Text Reuse For The Urdu Language," *IEEE,* 2017.

[7] "Run AI," [Online]. Available: https://www.run.ai/guides/gpu-deeplearning#:~:text=Why%20Use%20GPUs%20for%20Deep,without%20sacrificing%20efficiency%20 or%20power.

Plagiarism Report:

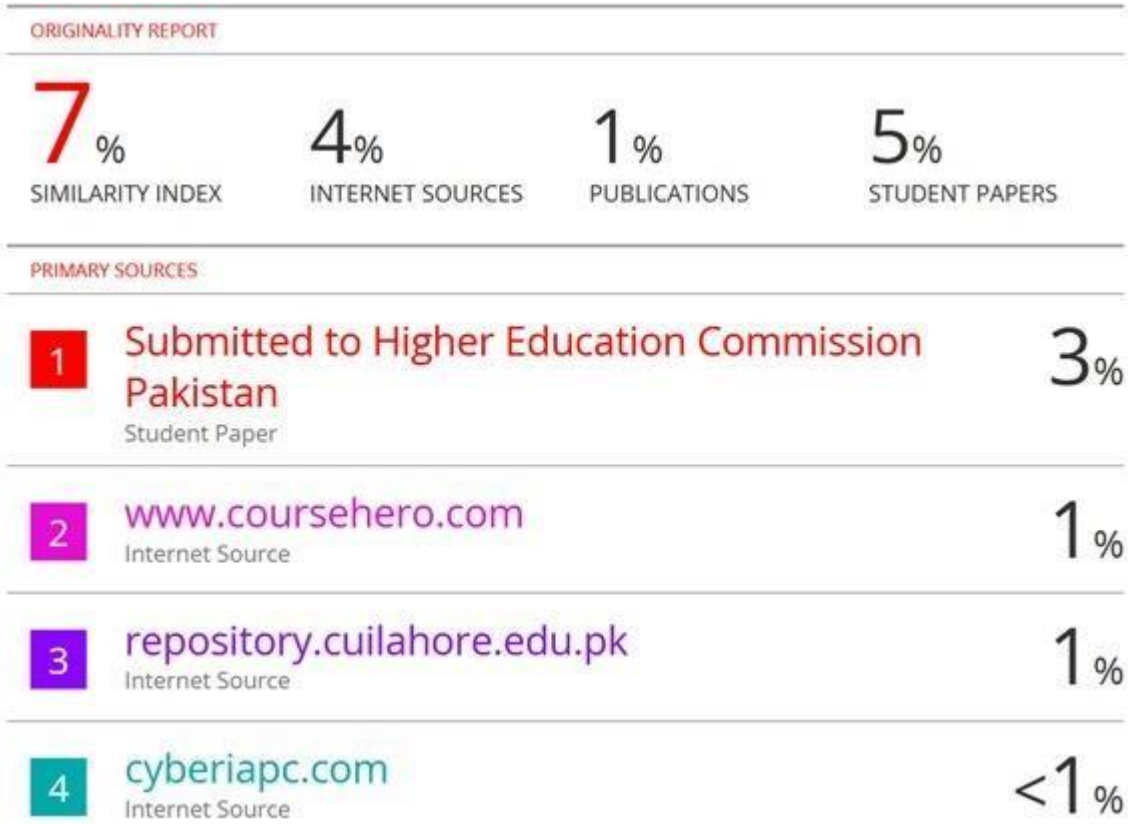


Figure 100 - Report

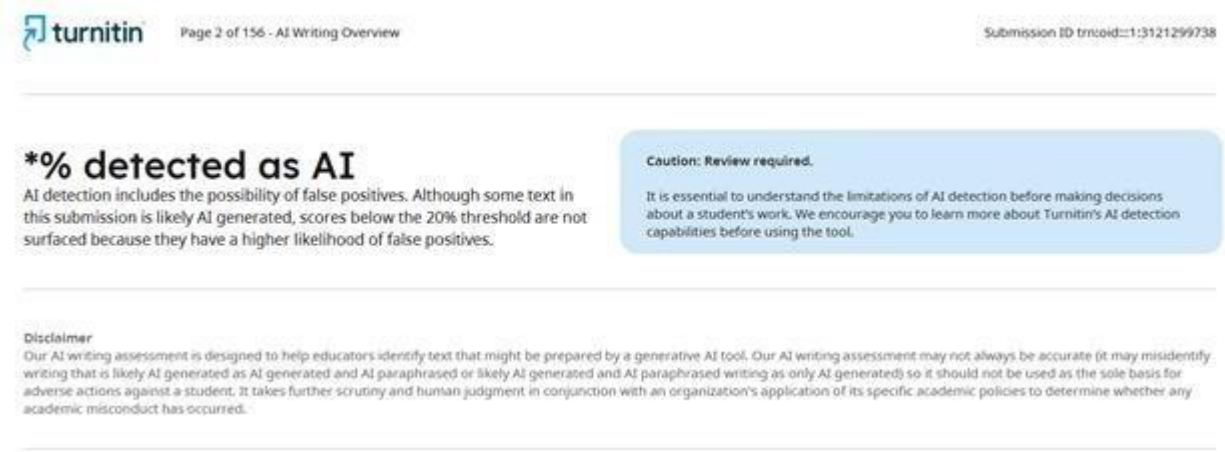


Figure 101- AI Report