



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
HIMALAYA COLLEGE OF ENGINEERING
[CT-654]

A
PROJECT REPORT
ON
LUGGAGE TRACKING SYSTEM

SUBMITTED BY:

LOKENDRA JOSHI [HCE078BCT020]
RAGHABENDRA CHAUDHARY [HCE078BCT029]
SAMBAD KHATIWADA [HCE078BCT035]
SANTU JHANKRI MAGAR [HCE078BCT039]

**A PROJECT REPORT SUBMITTED TO DEPARTMENT OF
ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL
FULFILLMENT OF THE REQUIREMENT FOR BACHELOR'S
DEGREE IN COMPUTER ENGINEERING**

Department of Electronics and Computer Engineering

Lalitpur, Nepal

March, 2025

LUGGAGE TRACKING SYSTEM

Submitted by:

Lokendra Joshi [HCE078BCT020]

Raghabendra Chaudhary [HCE078BCT029]

Sambad Khatiwada [HCE078BCT035]

Santu Jhankri Magar [HCE078BCT039]

“A Project Report Submitted in Partial Fulfillment of the Bachelor’s
Degree in Computer Engineering”

Supervisor:

Er. Shiva Shrestha

Department of Electronics and Computer Engineering

Himalaya College of Engineering

Tribhuvan University

Lalitpur, Nepal

March, 2025

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Himalaya College of Engineering, our Head of Department **Er. Ashok Gharti Magar**. We are very thankful to our project coordinator **Er. Ramesh Tamang** for managing time to conduct the project defense and for providing the resources regarding the guidelines regarding the project. We would also like to express our thanks to supervisor **Er. Shiva Shrestha** for regularly guiding us in every step of project development and providing us valuable suggestions and necessary resources regarding the project and his efforts in every step from project development from providing resources to guiding till the end were very valuable. We would like to thank all other friends and colleagues who provided their supported and inspired and motivated us to complete the project on time.

Group Members

Lokendra Joshi (HCE078BCT020)

Raghabendra Chaudhary (HCE078BCT029)

Sambad Khatiwada (HCE078BCT035)

Santu Jhankri Magar (HCE078BCT039)

ABSTRACT

Mishandling and loss of luggage continue to be significant challenges in airport operations, causing passenger dissatisfaction. Existing luggage tracking methods rely on manual reporting, which is time-consuming, error-prone, and inefficient. This project proposes a web-based Luggage Tracking System that allows passengers to report lost items, view found luggage posts, and claim their lost luggage through a secure and automated platform. To enhance security and prevent unauthorized claims, the system implements OTP (One-Time Password) verification via email, ensuring that only the rightful owner can access or claim lost luggage. The backend, built with Express.js and MongoDB, ensures scalability and efficient data management, while a Luggage Tracking Pin (LTP) generated using the SHA-256 hashing algorithm strengthens the luggage identification process. Additionally, it employs hybrid NLP based fulfillment chatbot to provide real-time support and assist users in navigating the process. With features such as basic query regarding post, report status, complain register etc. It uses Natural Language Processing (NLP) to recognize user query and answer accordingly. The project is technically and economically feasible, leveraging open-source technologies to ensure accessibility and cost-effectiveness.

Keywords: Hybrid NLP Based Fulfillment Chatbot, Luggage Tracking System (LTS), Natural Language Processing (NLP), OTP Verification, SHA-256, Luggage Tracking Pin (LTP)

TABLE OF CONTENT

ACKNOWLEDGEMENT	iii
ABSTRACT.....	iv
LIST OF FIGURES	vii
LIST OF TABLES.....	vii
LIST OF ABBREVIATIONS	viii
CHAPTER 1: INTRODUCTION	1
1.1 Background.....	1
1.2 Problem Statement	2
1.3 Objectives	2
1.4 Scope and Application	2
1.5 Report Organization.....	3
CHAPTER 2: LITERATURE REVIEW	4
CHAPTER 3: REQUIREMENT ANALYSIS.....	7
3.1 Functional Requirements	7
3.2 Non-Functional Requirements	8
CHAPTER 4: SYSTEM DESIGN.....	9
4.1 Level 0 DFD	9
4.2 Level 1 DFD	10
4.3 Sequence Diagram	11
CHAPTER 5: METHODOLOGY	12
5.1 Block Diagram of The System.....	12
5.2 Flow Charts of The System.....	14
5.3 Implementation	17
5.4 Admin Panel.....	20
5.5 Chatbot Implementation:	22

5.6 Software Development Life Cycle Model	34
5.7 Tools Used	35
CHAPTER 6: RESULT AND ANALYSIS	36
CHAPTER 7: CONCLUSION	39
7.1 Conclusion	39
7.2 Future Enhancements.....	39
REFERENCES	40
APPENDICES	42

LIST OF FIGURES

Figure 1.1 Report Organization	3
Figure 3. 1 Use Case Diagram of The System.....	7
Figure 4. 1 Level 0 DFD	9
Figure 4. 2 Level 1 DFD	10
Figure 4. 3 Sequence Diagram.....	11
Figure 5. 1 Block Diagram Of System.....	12
Figure 5. 2 Flowchart Representation Of Report.....	14
Figure 5. 3 Flowchart Representation Of Post.....	15
Figure 5. 4 Flowchart Representation Of Claim Operation	16
Figure 5. 5 Chatbot Implementation	22
Figure 5. 6: SCRUM Framework [13].....	34
Figure 6. 1 Response time of several queries asked	36
Figure 6. 2 Single And Multi-Post Scenario	37

LIST OF TABLES

Table 6. 1 Comparison of Existing SITA Air Tracker And Luggage Tracking System.....	38
--	----

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheet
HTML	Hypertext Markup Language
IATA	International Air Transport Association
IEEE	Institute of Electrical and Electronics Engineers
JWT	JSON Web Token
LTP	Luggage Tracking Pin
NLP	Natural Language Processing
NLU	Natural Language Understanding
OTP	One Time Password
SHA	Secured Hashing Algorithm
TF-IDF	Term Frequency – Inverse Document Frequency
VS	Visual Studio

CHAPTER 1: INTRODUCTION

1.1 Background

Airport luggage handling systems have traditionally faced numerous challenges, including the risk of luggage being misplaced, delayed, or even lost. According to the International Air Transport Association (IATA) there has been 69 mishandled bags per 1000 passengers in 2023. Despite the advancements in technology and luggage handling systems, the issue of lost luggage remains a persistent challenge. The process for passengers to report lost luggage and for airport staff to track it is often manual, time-consuming, and prone to errors, leading to dissatisfaction and frustration among passengers. The report process consists of time-consuming paper works and documents for verification that might not be present at current time. Furthermore, the lost and found luggage details are uploaded to third party sites like Facebook, Instagram, etc. But there is no guarantee that people will upload there or even if it is uploaded, finding the information on is a tedious task.

To mend the problems stated above, The Luggage Tracking System focuses to track the lost luggage in the airport. The luggage not only includes bags, they can be citizenship cards, passport and other necessary documents or objects. This system collaborates with airport authorities and airlines digitally to serve the passengers that have lost their luggage.

Furthermore, to reduce the hectic process to contact the authority for basic query, lodging complaint, checking report status by calling them which are often unanswered is also solved by integrating an assistant chatbot that uses NLP (Natural Language Processing) and replies general user queries, lodges complaint of the user, checks report status as well as navigates users to various procedures. The answer is given in most understandable and user-friendly way as possible.

1.2 Problem Statement

Mishandling of luggage remains a significant challenge at airports worldwide, including in Nepal. Lost luggage imposes high costs on airlines and airports due to compensation claims, recovery efforts, and productivity losses.

In Nepal, there is no proper system for passengers to track their lost luggage. Currently, passengers must manually fill out forms and provide authorized identification (e.g., Photo ID, Passport, Driving License, Voter ID) when reporting lost luggage. Frequent visits to the airport for updates add to the inconvenience and asking simple query to the authority can also be time consuming.

This project proposes a web application to address these issues by digitizing the process of reporting and tracking lost luggage, providing a more efficient and customer-friendly solution. The chatbot integrated helps to answer basic user query and fulfill some basic records of the user.

1.3 Objectives

The main objective of the project is:

- To develop a web application that allows easy finding and claiming of lost luggage at airport and integrate a chatbot to help user with queries.

1.4 Scope and Application

The project is a web application that is used for reporting and recovering lost luggage at airports. This system enables the users to view/search their lost luggage or report and claim their luggage. The system's real-time updates enable passengers to receive timely notifications on the status of their luggage, significantly improving the overall customer experience.

The web application uses smart match functionality, it allows quick match lost luggage report with found post attempts, reducing recovery time and minimizing human error. The system also provides detailed reports and analytics, enabling airport management to identify trends and improve luggage handling operations. With its application across various airport terminals both for domestic and

international, this solution enhances operational efficiency, reduces customer frustration, and ensures better communication between passengers and airport staff, ultimately contributing to a smoother, more reliable luggage handling process and minimized the use of unreliable third-party sources.

Furthermore, chatbot is itself solving some basic user queries and if there are any complaints to lodge, it lodges the complains which are viewed and replied/responded by the admins. Users are also able to check their report status there.

1.5 Report Organization

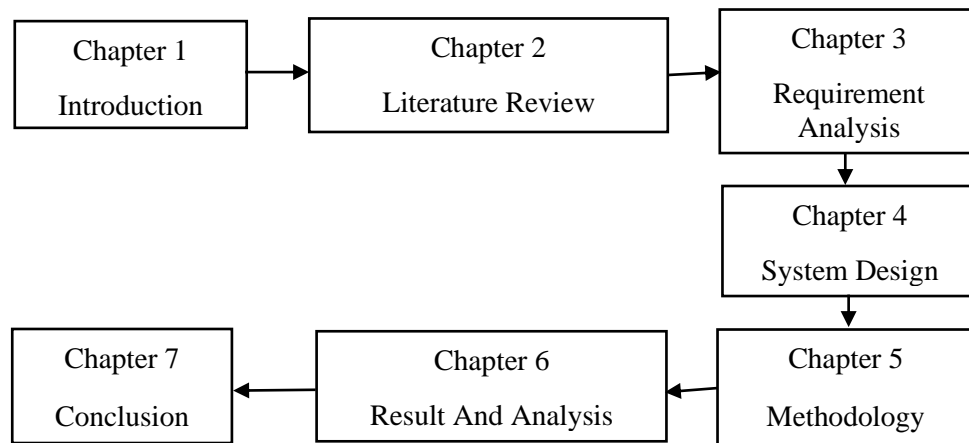


Figure 1.1 Report Organization

The project report is divided into several sections that provide a comprehensive overview of the project's objective, methodology and outcomes. The report begins with an introduction that outlines the background, objective and scope. The literature review shows the in-depth analysis and research regarding existing works referencing the project. The requirement analysis portrays the functional and non-functional requirements and various aspects of feasibility. The system designs show different diagrams regarding the system. The methodology explains how they are implemented and then result and analysis are discussed and then we draw the conclusion.

CHAPTER 2: LITERATURE REVIEW

Systems like SITA World Tracer demonstrate the value of global collaboration and advanced search technologies in managing baggage [1]. It is a solution for managing mishandled baggage, helping airlines, airports, and ground handlers track and recover lost or delayed luggage efficiently. It uses a centralized, cloud-based platform to record and share data globally. It has self-service tools for passengers to report lost baggage, automation in baggage delivery and reconciliation and Integration with airline systems using IATA standards and IEEE rules [2]. It is less flexible and less accurate it does not have a calculated mechanism. The tracing is based mainly on luggage appearance and works. The system is under IATA which is a global trade organization but this system is only present across North America and some European countries such as Switzerland, Finland etc. Even in those countries it only operable in central airports.

World Tracer is a baggage tracing and management system used by over 300 airlines, airports, and handling agents [3]. It provides functions such as long-term storage of baggage mishandling information, statistical analysis of baggage handling performance, and fraud prevention for baggage claims. The document outlines various World Tracer transactions and codes used for baggage tracing, including codes for reasons of baggage loss, damage, or pilferage. This facility is only available for business-oriented flights and not for general public.

Fly SAS is dedicated website to track luggage in Scandinavia [4]. The users are able to trace the bags through their reference number. The users can fill the form online and contact the authorities for updates. They can send the luggage photos to the authorities.

Furthermore, chatbot can be created conveniently using the dialog flow platform [5]. Where we can create our own chatbot, train it and import it to our website. This has been a growing trend for service-oriented web pages and the most salient feature of this is the collection of usage information that can be used in research as well as the improvement of the system.

BIARR [6] explains the use of chatbot that uses user query as input, splits the input to different units called tokens that acts as basic building block for NLP (Natural Language Processing) use Lemmatization to simplify the words, process them and give the result in the human understandable form. We can also give knowledge bases to the chatbot to increase the accuracy of the answer.

Bot framework SDK [7] provides a framework for creating customized bot. It is a framework service of Microsoft where we can customize bot behavior using editor tool called SDK tool.

The use of chatbots, mostly in healthcare, education and sales and marketing field for customer services [8]. It has greatly helped to reduce the time and money by reducing the efforts to solve basic problems of the customers. Its initialization was done using AIML and LSA, which was then succeed by char script, NLP and NLU. There are different platforms and framework now that help in creating bots simply by training, making module, creating API and using that API as desired.

Chatbots nowadays are simply implemented using APIs [9]. We can choose any platform. The basic process includes setting up bot environment, create bot intentions and entities, develop API integration logic, setting up web hooks and creating custom responses.

Natural language processing (NLP) is gaining momentum in management research for its ability to automatically analyze and comprehend human language [10]. Yet, despite its extensive application in management research, there is neither a comprehensive review of extant literature on such applications, nor is there a detailed walkthrough on how it can be employed as an analytical technique.

Dialogue systems have become recently essential in our life. Their use is getting more and more fluid and easy throughout the time. This boils down to the improvements made in NLP and AI fields [11]. The impact of NLP on information retrieval tasks has largely been one of promise rather than substance and processing this information and giving the response in user understandable form can also be done in simple to complex manners depending on scope, necessities and available resources.

Cosine similarity is a widely used metric that is both simple and effective [12]. This paper proposes a cosine similarity ensemble (CSE) method for learning similarity. In CSE, diversity is guaranteed by using multiple cosine similarity learners, each of which makes use of a different initial point to define the pattern vectors used in its similarity measures. The CSE method is not limited to measuring similarity using only pattern vectors that start at the origin. In addition, the thresholds of these separate cosine similarity learners are adaptively determined. The idea of using a selective ensemble is also implemented in CSE, and the proposed CSE method outperforms other compared methods on various data sets.

CHAPTER 3: REQUIREMENT ANALYSIS

3.1 Functional Requirements

- i. Users should be able to view the found/exchanged luggage posts containing luggage photo and claim that by validating their details and OTP.
- ii. Users should be able to report their lost luggage filling the required information.
- iii. Users should be able to post the found/exchanged luggage along with the photo of luggage.
- iv. Message / alert to the users in a real time as soon as the luggage is tracked.
- v. Users should be able to question the chatbot and get relevant answer quickly as well as lodge a complaint from the chatbot.
- vi. Separate admin login section and admin panel.
- vii. Admin should be able to view/edit all details as well as reply to complains.

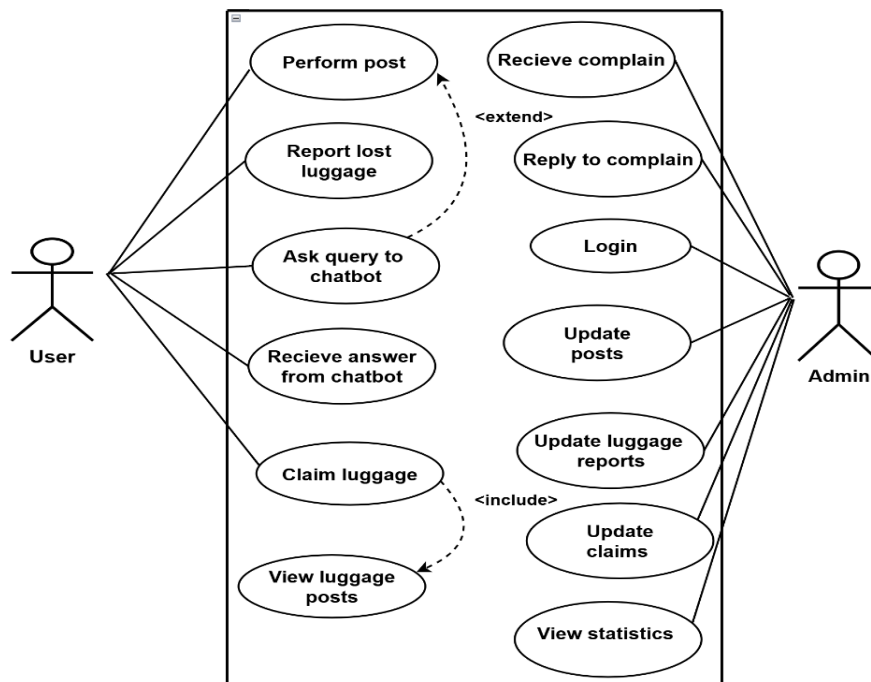


Figure 3. 1 Use Case Diagram of The System

3.2 Non-Functional Requirements

- i. Usability: The user interface should be easy to use.
- ii. Security: The system should keep the passenger data and admin separately with proper security standards.
- iii. Authorized claim: The claim of luggage should be authorized using OTP verification.
- iv. Compatibility: The system should be compatible with common web browsers such as Firefox, Chrome, Brave, Opera, etc.
- v. API integration: The system should support integration with third-party services and applications through APIs.

CHAPTER 4: SYSTEM DESIGN

4.1 Level 0 DFD

Figure 4.1 shows Level 0 DFD of the system. The users include the general user and admin. The input to system from users includes claim/post/report details, OTP and query to the system. The system provides OTP confirmation, reply as well as claim/post/report confirmation and mail to the user/passenger. Meanwhile, the admin provides reply to complaints, updated claim/post/report details, email, password and OTP to the system while the system provides claim/post/report details as well as the login confirmation.

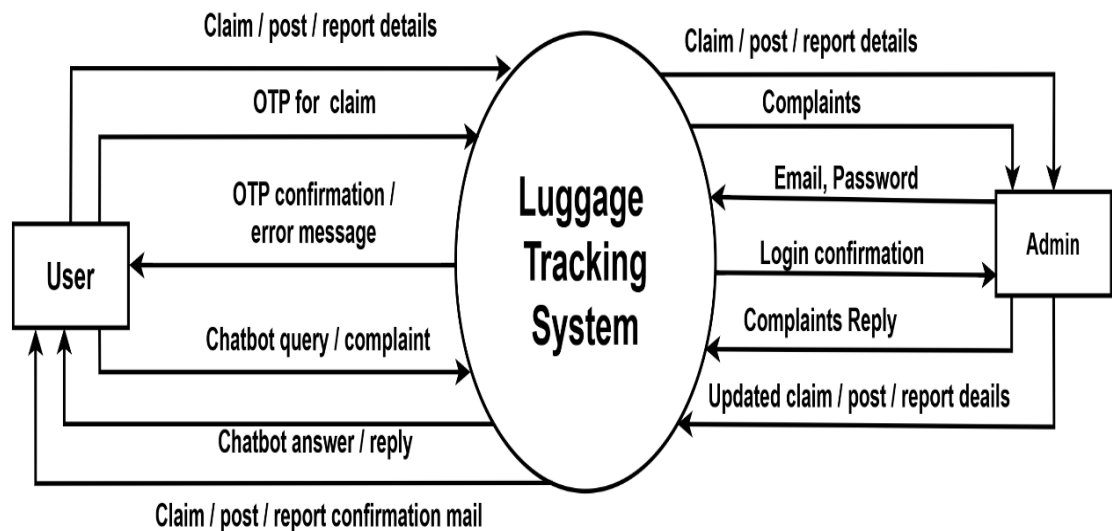


Figure 4. 1 Level 0 DFD

4.2 Level 1 DFD

Figure 4.2 shows Level 1 DFD of the system. It is in fact an expansion of Level-0 DFD. Here, we can observe that users can ask query or make complain to chatbot, receive answer. Perform post or claim or report operation, receive email for confirmation and guidelines. Also, admin can have access to and can update them and can also view complaints and reply to them and data flows across these processes and saved in database records accordingly.

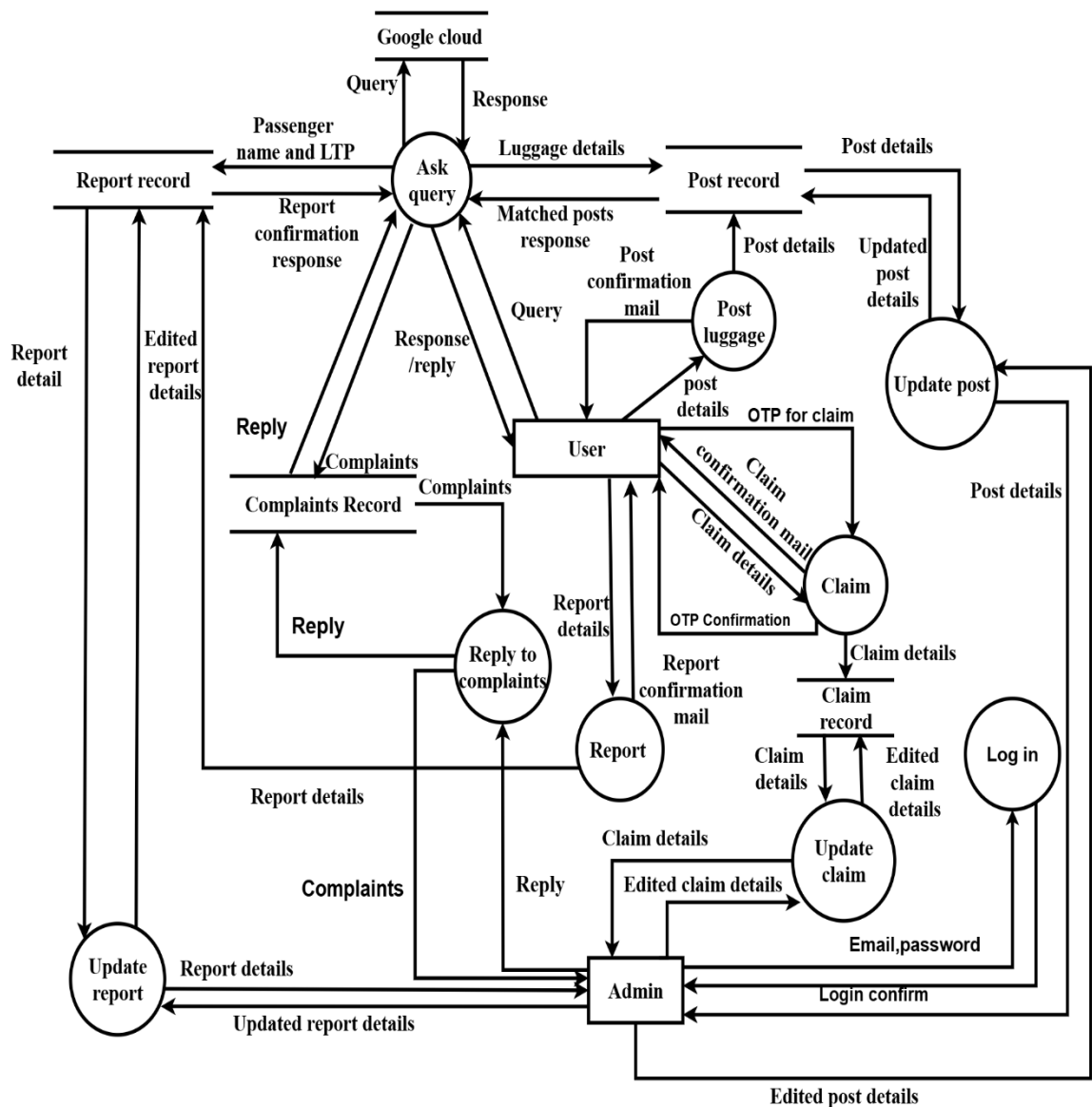


Figure 4. 2 Level 1 DFD

4.3 Sequence Diagram

Figure 4.3 shows sequence diagram of the system. Here, user asks query to the chatbot which checks answer to the databases based on general or fulfillment query. Users can post, claim and report their luggage, where different databases check and updated are done and the users are responded with confirmation/guideline. Similarly, admin can login and view/edit admin panel and reply to complains stored at database which is sent to customer through email.

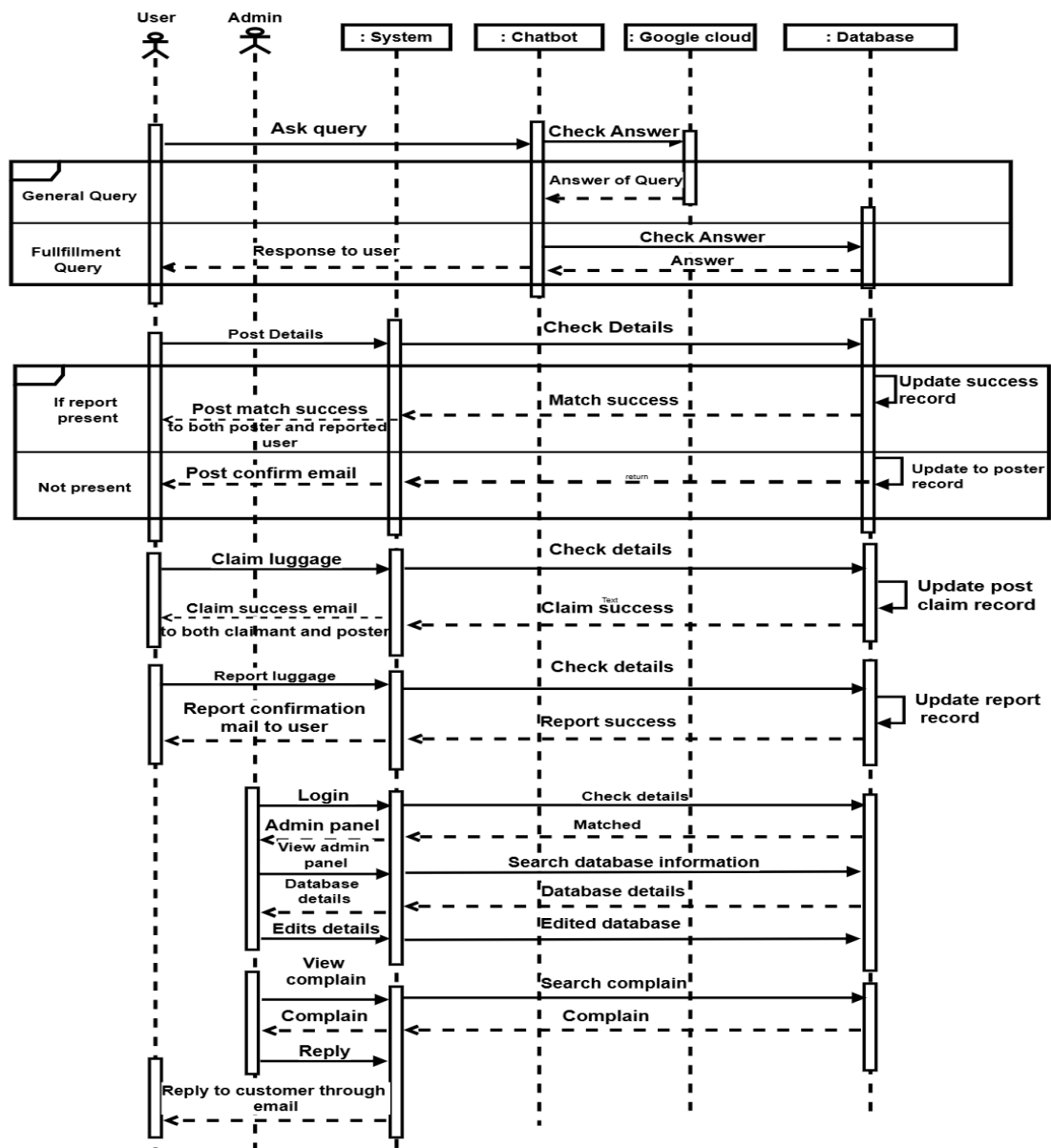


Figure 4. 3 Sequence Diagram

CHAPTER 5: METHODOLOGY

5.1 Block Diagram of The System

Figure 5.1 shows the block diagram of the system. The operations are performed under the collective implementation of system (frontend and backend), database and Gmail API which is provided/facilitated by the Nodemailer package.

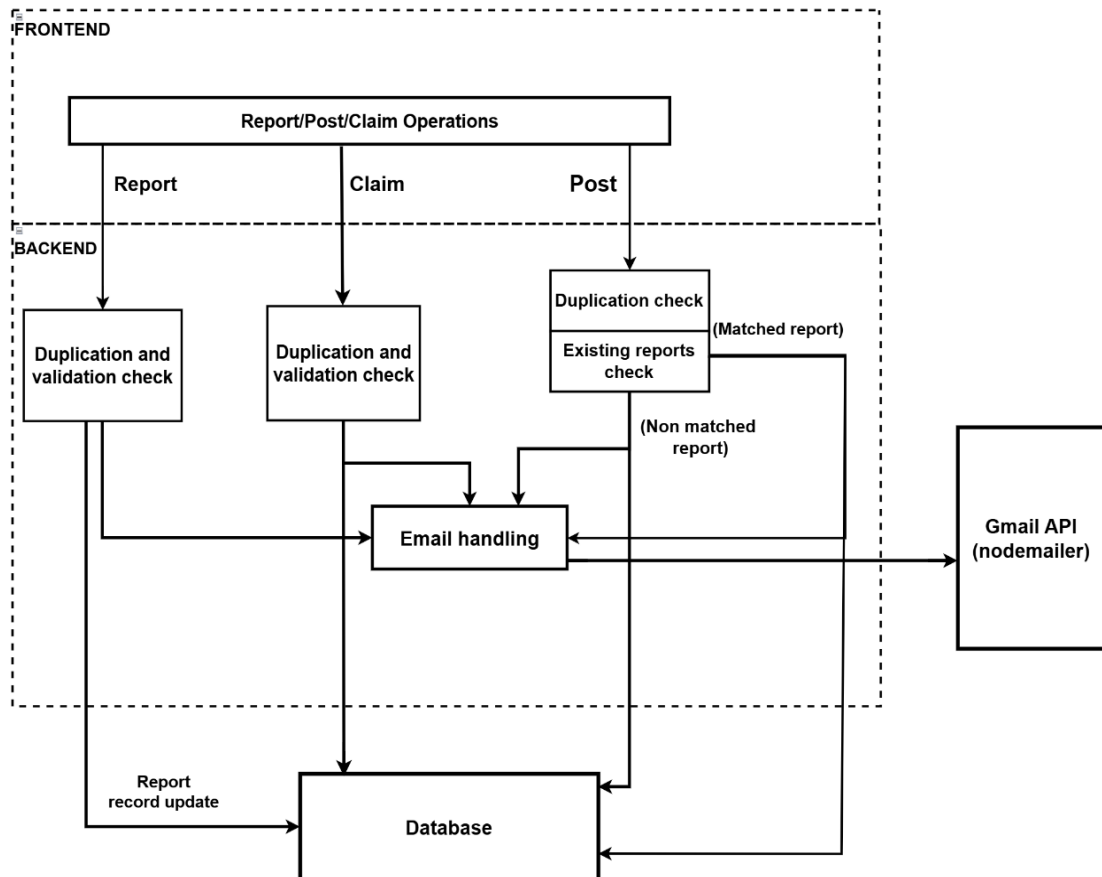


Figure 5. 1 Block Diagram Of System

5.1.1 User Level/Frontend Operation

The users are able to:

- i. Report their lost luggage by entering the required details.
- ii. Post the luggage if they have exchanged or found any.
- iii. Claim the luggage from the posts by entering the required details and verifying themselves by entering OTP (One Time Password).

5.1.2 Backend Operations

5.1.2.1 For Report Operation

- i. Duplication check is done in database to ensure if the report already exists to prevent that any multiple reports of same person/passenger exists in DB.
- ii. The details inserted by user are check against the details in database to ensure they are valid ensuring only the actual person is making the report.
- iii. After the duplication and validation confirmation, database is updated accordingly and email is sent to the user.

5.1.2.2 For Post Operation

- i. The poster/person makes posts regarding details on appearance and other aspects of airport, photo, ticket number on the tag, airport, email, name and phone of the poster/person who is attempting to post.
- ii. The report check is done to see if the report is already present of that luggage. If yes, the person who has posted and who has reported are mailed regarding each other information and further guidelines.
- iii. If not, then post is check against the existing posts to prevent duplication.
- iv. The database record is updated accordingly and email is sent to the user.

5.1.2.3 For Claim Operation

- i. The user can see the posts, search their luggage through details or simply by navigating and identifying photo.
- ii. If they find their lost luggage in the posts. The claim button popups the claim form, if the entered details are valid, the system asks OTP which is already sent to the valid/registered email of the passenger.
- iii. The database record is updated accordingly and email is sent to poster and claimant.

5.2 Flow Charts of The System

5.2.1 Report

Figure 5.2 shows the report operation. Duplication check is done to ensure there are not multiple same reports. If report is already present, popup message is sent saying the report already existing. If not, the entered values are checked if they are in correct format, then the details are checked with the details in the database. If the details don't match an error popup saying passenger not found or enter correct details. If details match, the report is successful, the email is sent to the user regarding the confirmation of report. The report record is updated too. This report is the pending state, i.e. it is in waiting state to be matched.

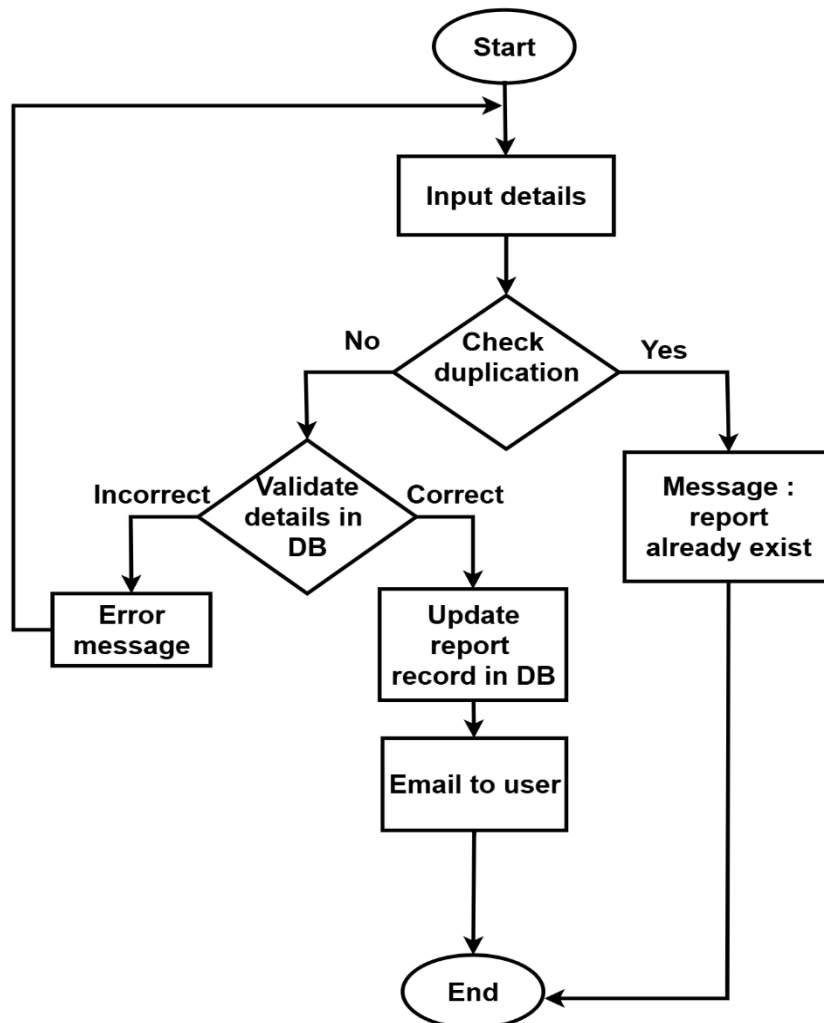


Figure 5. 2 Flowchart Representation Of Report

5.2.2 Post

Figure 5.3 shows the post operation. The luggage that is exchanged or found misplaced luggage is posted by the poster. During the post attempt, the duplication check is done to prevent multiple same post. If there is same post already available, the popup message appears saying post report already present. If there are no duplicate posts, the details provided by the poster are checked whether it is in proper format or not. If it is not in proper format, the popup error message appears. After checking format, the ticket number provided is checked with the ticket numbers in the report record. If ticket number is present, the reports are matched. The success record in the database is updated and both the person who has attempted to post and who has reported are both mailed regarding each other's contacts and further guidelines. Else, the post is successful and the record of poster and report found in database are updated.

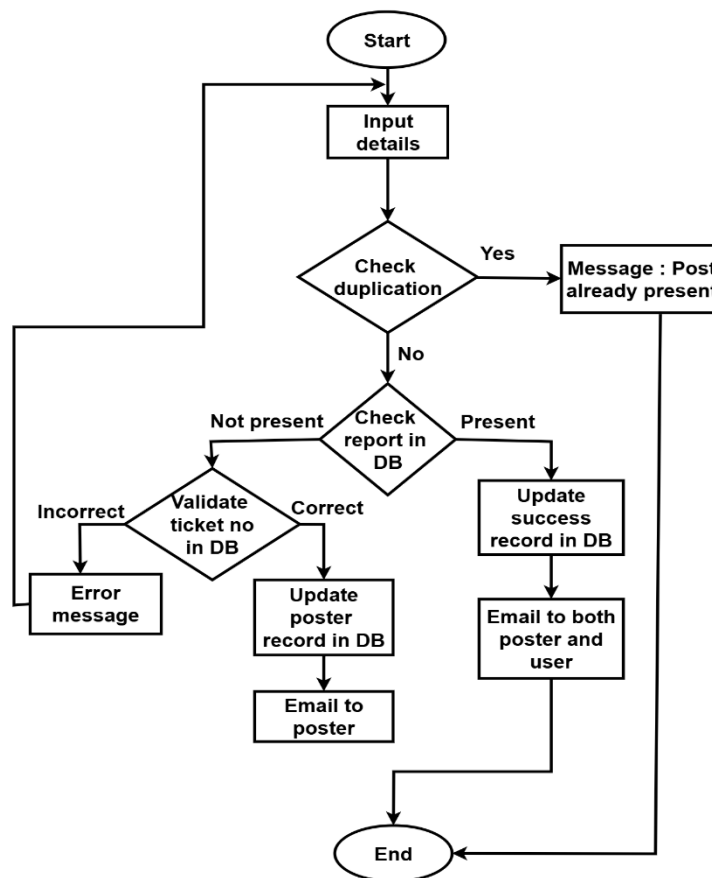


Figure 5. 3 Flowchart Representation Of Post

5.2.3 Claim

Figure 5.4 shows the claim operation. The person who has lost their luggage checks the posts if their luggage is present, they attempt to claim that. They enter their details in the claim form. If the entered details are incorrect or the format is incorrect, the message regarding the error is shown. If the entered details are correct, the OTP is sent to the person attempting to make the claim to make sure the correct person is attempting to claim. On entering the correct OTP, the claim process is success. The post-claims record collection in the database is updated and both poster and claimant are mailed the information regarding their contacts as well as further guidelines.

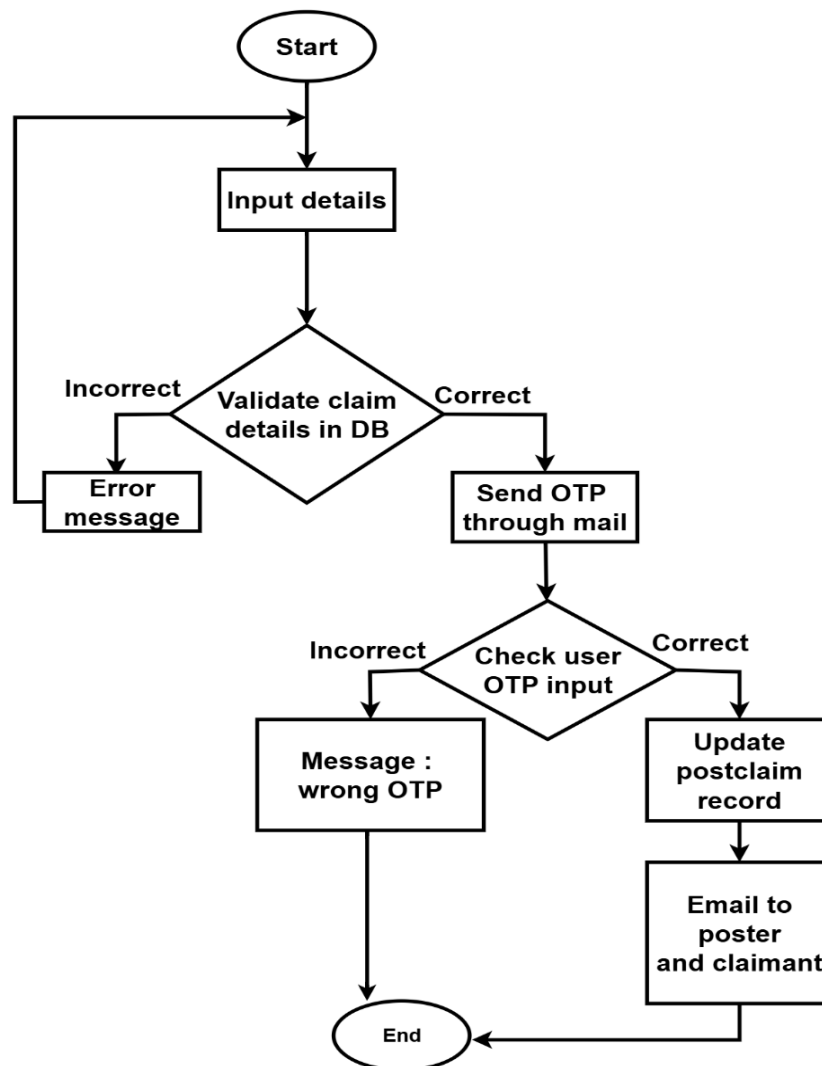


Figure 5. 4 Flowchart Representation Of Claim Operation

5.3 Implementation

5.3.1 Passenger Data Set

The dataset will be imported to our database along with the generated Luggage Tracking Pin number the data will be provided by data provider or airline authority.

We implemented the JavaScript code in backend with express. The code accepts data as well as filters the data older than one months. The dataset contains passenger name, contact number, ticket number and LTP number. As this is a project conducted within the educational vicinity and not initially for real operation purpose. Collecting the data from airport will be difficult. So, we will be generating the dummy data of about 8000 to 10000 users and 3 to 4 airlines in which these passengers were onboard.

5.3.2 Database Collections and Schemas

The database in MongoDB contains the following collections:

- i. Passengers: This collection includes the main record of the passengers who has travelled through the airline. The schema includes full name, ticket number, LTP number, phone number, airline and date.
- ii. Report: This collection stores the user data of the users that have reported or whose reports are pending. The schema includes full name, ticket number, LTP number, phone number, airline and date.
- iii. Poster: This collection includes the information of poster who has posted. full name, phone number, ticket number, airport, email, more details and upload date are the schemas in this collection.
- iv. Report-found: The collection includes the information of the passengers whose luggage is posted by the user. The schema is same as report collection.
- v. Success: This collection contains the information the poster and the reported passenger whose report is matched with the post attempt by the poster. The schemas include all the information of the report collection and poster name, email and date.

- vi. Post-claims: The collection includes the information of the poster who has posted and the person who has claimed i.e. claimant. The schema includes claimant name, claimant email, claimant phone, poster name, poster email, poster phone, ticket number, LTP number, airport, uploaded at, more details and image.

5.3.3 Luggage Tracking Pin Generation

We use SHA-256 hashing algorithm to generate the Luggage Tracking Pin number.

Algorithm for implementing 5-digits LTP number:

- i. Get the current time:
Retrieve the current timestamp (date and time) in ISO format using `new Date().toISOString()`. This gives a unique string every time the code is run, based on the time.
- ii. Hash the timestamp:
Use the SHA-256 hashing algorithm to convert the timestamp into a 256-bit hash (a long string of hexadecimal characters). This step ensures that the resulting value is unique, as the timestamp is always changing.
- iii. Extract the first 5 characters:
Take the first 5 characters from the hexadecimal hash string. Since the hash is a long string of letters and numbers, you only need the first 5 characters for the number.
- iv. Convert hexadecimal to decimal:
Convert the 5 characters (which are in hexadecimal) into a decimal number. This will give you a numerical value.
- v. Ensure the number is 5 digit:
- vi. Apply the modulo operation ($\% 100000$) to make sure the final number is always a 5-digit number. The modulo operation ensures the number is always 5-digit.
- vii. Return the result.

5.3.4 Input Formats

- i. The ticket number is first three uppercase characters and then 5-digit numbers.

E.g. SHR12345

- ii. The LTP number is a 5-digit number.

E.g. 12345

- iii. The name (of poster as well as user) is combination of first name and last name.

E.g. Full name: Ram Thapa

- iv. The phone number is a 10-digit number.

Eg:9812345678

- v. Email should be a Gmail address in proper format.

E.g. abc@gmail.com

- vi. The generated OTP is a six-digit number.

E.g. 342579

5.3.5 Email Services

The OTPs, action confirmations, complain replies, all are done through email. We have used Gmail service for this using Nodemailer which is a package of Node.js. An API is used in Nodemailer when sending emails, particularly through SMTP (Simple Mail Transfer Protocol) or third-party email services. So, we have used Gmail's SMTP API to send the information using Gmail service. We can use the API and send the email through the Gmail account whose two-factor authentication must be turned on. We have to generate an in-app password for this service, which is a very confidential password. This credential gives the access to the gmail account through which the mails are sent. We have to set Nodemailer as follows:

service: 'gmail',

authentication:

user: 'sender@gmail.com',

pass: 'app password'

Here, app password is obtained only when 2FA is enabled in the sender account.

5.3.6 OTP Generation

The OTP generation feature is included while claiming and admin login to make sure that it is being done by authorized person only. We have implemented a simple JavaScript code to generate a random six-digit OTP code and Nodemailer is used to send that OTP to the user through Gmail.

We have generated OTP as:

$$\text{OTP}=[100000+\text{random}()\times 900000]$$

Here,

Math.random() function generates a random floating-point number between 0 (inclusive) and 1 (exclusive). This is denoted as: random(). To scale the number, we multiply the number with 900000 and to ensure result between 100000-999999 we add with 100000 and round off the result to generate OTP.

5.4 Admin Panel

There is a fully functional admin panel where:

- i. Admin can enter their valid email and password and login. The system asks OTP. On correct OTP input, the admins can log in to the system where entire admin panel is accessible.
- ii. All the data of all the database collections are there.
- iii. Admin can update those data.
- iv. They can view the counts of total passenger data count, reported luggage, posts, successful match counts of the luggage, post claims count till date as well as the graphical representation of the data in form of pie charts and bar graphs.
- v. The users can view the complains as well as reply to them which will be sent to the users.
- vi. The log in session handles by JWT tokens.

5.4.1 Database Access

Each data are imported from the database collections on MongoDB which can be updated and deleted by the users and the database will apply changes accordingly. The database collection given access to the admin are report, reportfound, successes, postclaims, passengers and complaints. The count of data in these databases are calculated and also shown in a summarized way for easy monitoring.

5.4.2 Graphical Representations

For easy and quick review of overall status we have implemented the pie chart and bar chart representations of the data. For this chart JS is being used. It is a popular JavaScript library used for creating interactive and visually appealing charts on web applications. It is simple, flexible, and supports various chart types. Here, we imported the necessary libraries for implementing the pie and bar charts, then fetched all the data with their counts and then displayed them using a properly defined function. The pie chart describes the total lost reports, total active posts and the normal luggage. While the bar chart shows relative posts as well as pending reports based on airlines.

5.4.3 Complaint Handling

The complaint is registered by user through chatbots. The chatbot recognizes the complaint and saves it to the complaints record in the database.

These complaints consist of the schemas:

- i. Email: The email address of the user who sent the complaint.
- ii. Complaint: The details regarding the complaint.
- iii. Creation time: The date/time when the complaints are handled.
- iv. Status: Determines whether the admin has read the complaint or not.

The complaints are displayed in the stack format on the frontend with the latest complaints at the top so that latest complaints are accessed and addressed easily. Then, the complaints can be replied by the admin which is sent to the user in the formal format by using Nodemailer package.

5.5 Chatbot Implementation:

The chat bot is a hybrid NLP based fulfillment bot that replies to the user query as well as serves user by letting them complain, provide report status as well as find/recommend the posts based on the details.

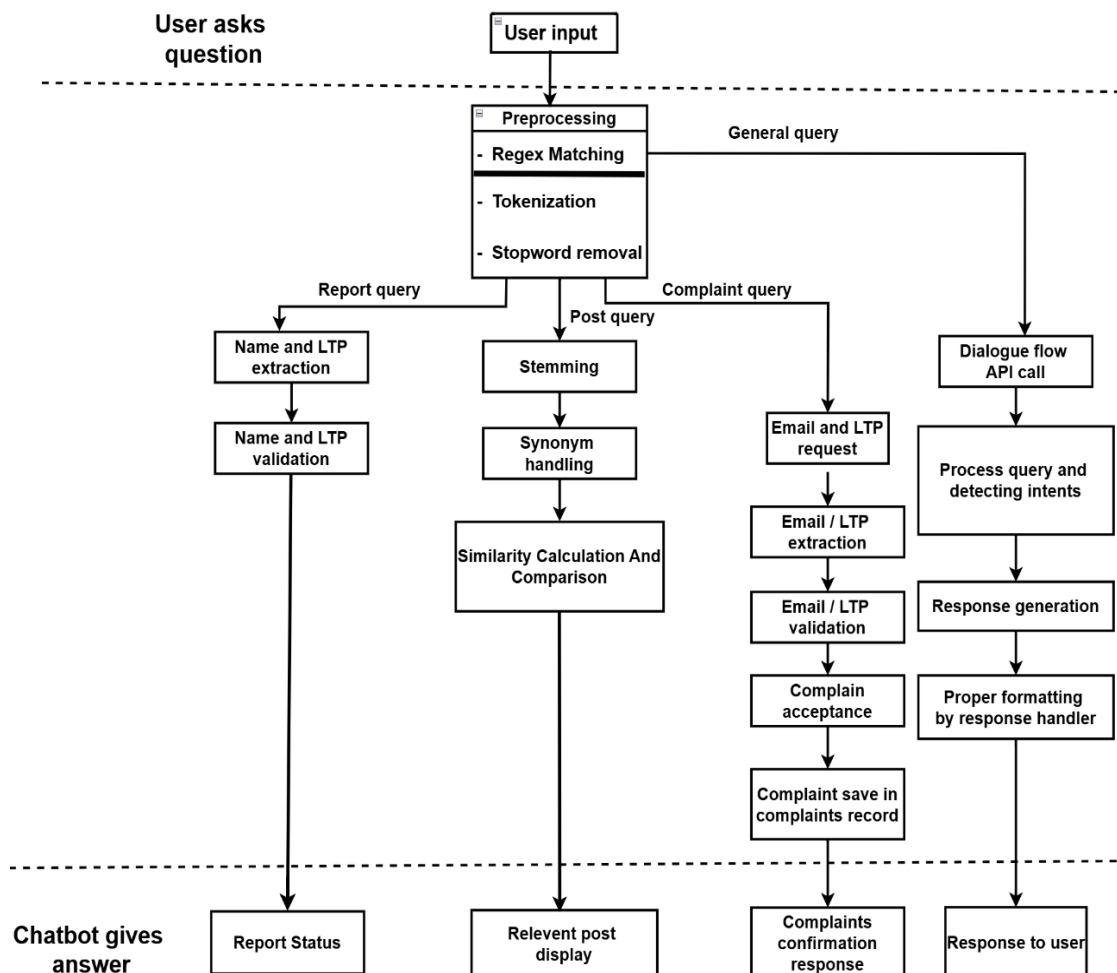


Figure 5. 5 Chatbot Implementation

Figure 5.5 shows processing of user input in a system that handles different types of queries, including report queries, post queries, complaints, and general queries.

5.5.1 User Input

The system starts with user input, which can be a general, complaint, report, or post query.

5.5.2 Preprocessing Stage

The input undergoes preprocessing which is common for all queries which includes:

5.5.2.1 Regex Matching

Regular Expression (Regex) are used to extract and validate user-provided information. This includes detecting intent, extracting email addresses, and identifying LTP numbers. The primary patterns used are:

```
/ ( ? : my name is | i am |this is | i'm | name | from)\s+([\w\s]+?)(?=\s*( Ltp | number  
|with|,| is | is there a report|$))/ i / \b\d {5} \ b / i
```

Example: Given input: "I am Alice Brown with LTP number 56789."

Regex match extracts:

Full Name: Alice Brown

LTP Number: 56789

5.5.2.2 Tokenization

Tokenization is the process of breaking text into individual words or tokens. It helps in structuring textual data into meaningful units that can be processed further. The system uses WordTokenizer from the natural library. This step is useful for analyzing user messages.

Tokenization Function: $T(m) = \{t_1, t_2, \dots, t_n\}$ Where $T(m)$ is the set of tokens extracted from message.

Example: Input: "Can you check my report status?"

Tokenized: ["Can", "you", "check", "my", "report", "status"]

Now, these tokens are further processed so that only meaningful tokens are extracted and only meaningful information is understood/proceeds to give correct response.

5.5.2.3 Stop Words Removal

Common words (e.g., the, a, in etc.) do not add significant meaning and are removed to improve processing efficiency using the stopword library.

After stop words removal: ["check", "report", "status"]

5.5.3 Report Query

At first the system should determine if the extracted information corresponds to a report query. To identify and extract user information, the system uses regex patterns.

```
/(report|check|status|lookup|find|is there a report)/i
```

Example: "My name is John Doe, and my LTP number is 12345. Is there any lost report?"

This ensures that the input is actually about a report query and then ask for the full name and LTP number and further processing starts.

5.5.3.1 Name and LTP Extraction

The system extracts the full name and LTP number using regex matching.

```
/(?: my name is | i am |this is | i'm | name | from)\s+([\w\s]+?)(?=\s*( Ltp | number |with|,| is | is there a report|$))/i /\b\d {5} \b/i
```

These patterns help to extract:

Full Name: From phrases like *"My name is Ram Thapa"* (capturing "Ram Thapa").

LTP Number: A 5-digit number (e.g., 12345).

Eg query: Is there a report from Ram Thapa with ltp 12345

Extracted name: 'Ram Thapa'

Extracted LTP: '12345'

So, this name and LTP are extracted and are proceed for further matching in the main passenger database.

5.5.3.2 Database Verification and Report Status

Once the system extracts the full name and LTP number, it checks the MongoDB database and give the report status. The possible outcomes are:

- i. If match is found then system respond with “Yes, there is a report of yours in the record.”
- ii. If no match is found then system respond with “Sorry, no records of your report.”

5.5.4 Post Query

At first the system should determine if the extracted information corresponds to a post query. To identify and extract user information, the system uses regex patterns.

```
/\b(post|submit|raise|file|report an issue|log a case|register)\b/i
```

Example: "I lost my blue bag with a TSA lock."

This ensures that the input is actually about a post query and further processing starts.

5.5.4.1 Stemming

Stemming is the process of reducing words to their root form, helping in better matching between variations of words. This is done using PorterStemmer.

Example:

Input: ["lost", "blue", "bag", "TSA", "locking"]

Stemmed: ["lost", "blue", "bag", "TSA", "lock"]

Wheels: ['rollers', 'casters', 'spinners']

Handle: ['grip', 'strap', 'handgrip']

5.5.4.2 Synonym Handling

Synonyms are mapped to standard words to improve recognition similar meanings across different words.

Example:

Synonyms of "bag": ["sack", "backpack", "luggage", "pouch"]

If input contains "backpack", it is considered as "bag".

5.5.4.3 Similarity Calculation

The system then computes similarity scores between the user input and each report using various techniques.

Overlap Score

The overlap score measures the proportion of common tokens between the tokens extracted from the user's message and the tokens from each post. It uses common tokens and overlap score.

- i. Common tokens: This array contains tokens from the message that match any of the tokens in the post (considering synonyms).
- ii. Overlap score: It is computed as the ratio of the number of common tokens to the maximum number of tokens between the message and the post.

$$\text{Overlap Score} = \frac{\text{Number of common tokens}}{\max(\text{Total Tokens In message}, \text{Total Tokens in post})}$$

Exact Match

It determines whether the extracted tokens and the tokens to check has exact match. The synonyms of the words are also considered as exact match.

- i. If there is exact match between words, synonyms:
Score= 1
- ii. If there is no exact match between words, synonyms:
Score=0

Key Descriptor Weight

The key descriptors are the salient features of the object need to identify it, we adjust separate weight to these characteristics tokens to prioritize them. The key descriptor

weight is calculated as the ratio of the number of common tokens that are key descriptors to the total number of key descriptors. This means that if more key descriptors are found in common, the weight will be higher. The key descriptors can be logo, build, specific colors etc.

$$\text{Key descriptor weight} = \frac{\text{Number of key descriptor matches}}{\text{Total number of key descriptors}}$$

Context Weight

The context weight is calculated as the ratio of the number of common tokens to the total number of tokens in the user's query. This can be expressed mathematically as:

$$\text{Context weight} = \frac{\text{Number of common tokens}}{\text{Total number on tokens in user query}}$$

TF-IDF (Term Frequency-Inverse Document Frequency):

TF-IDF is a statistical measure that evaluates the importance of a word in a document relative to a collection of documents. It assigns higher weights to words that appear frequently in a specific document but rarely in others, making them significant for comparison.

$$\text{Formula: TF-IDF} = \text{TF} * \text{IDF}$$

Where:

- i. TF (Term Frequency) = Number of times a term appears in a document / Total number of terms in the document.
- ii. IDF (Inverse Document Frequency) = $\log \left(\frac{N}{df} \right)$, where N is the total number of documents and df is the number of documents containing the term.

Cosine Similarity

Cosine similarity measures the angle between two vectors in a multi-dimensional space. It is used to compare the similarity of two text inputs based on their TF-IDF representations.

$$\text{Formula: } \cos(\theta) = \frac{A \cdot B}{|A| * |B|}$$

Where:

- i. A and B are TF-IDF vectors of the user query and post details.
- ii. A.B is the dot product of the vectors.
- iii. |A| and |B| are their magnitudes.

Cosine similarity in our system is used to measure the similarity between the TF-IDF (Term Frequency-Inverse Document Frequency) vectors of the user's message and the stored posts. It's a way to quantify how similar two documents are by comparing their vector representations.

Levenshtein Distance:

Levenshtein distance calculates the number of single-character edits (insertions, deletions, or substitutions) needed to convert one string into another.

$$\text{Formula: } d_{\text{lev}}(a,b) = \text{min edits required to convert a into b}$$

Example:

Query: "luggage"

Post: "lugage"

Distance = 1 (since one "g" is missing in "lugage")

Levenshtein similarity score is calculated as:

$$S = 1 - \{d_{\text{lev}} / (\max(|a|, |b|))\}$$

Final Similarity Calculation:

Final Score = (0.4 * Overlap Score) + (0.1 * Exact Match) + (0.1 * Levenshtein Score) + (0.4 * Context Weight) + (0.4 * Key Descriptor Weight)

Comparison With Threshold:

If there are multiple matches, the system calculates the highest similarity score and filters posts within a defined threshold difference:

- Threshold Difference: 0.2
- If multiple posts have scores within 0.2 of the highest score, they are all returned.
- If only one post has the highest score exceeding the threshold, it is returned as a single match.
- Score difference = score of one post – score of another post
- If score difference < 0.2, multiple posts with less difference with each other are displayed if greater > 0.2 the post with higher score is show.

The system efficiently matches user queries with lost luggage reports using NLP techniques and similarity calculations. The system displays the most relevant posts to the user based on the similarity scores.

Example Workflow for Singular Case:

User Input: "I lost my blue bag with a zip."

Processed: ["blue", "bag", "zip"]

Matched Posts:

A: "Navy bag zipper" (Score: 0.85)

B: "Pink bag with zip" (Score: 0.55)

Output: "Yes, there is a post related to your detail. Check the post by John Doe."

Example Workflow for Multiple Cases:

User Input: "I lost my red suitcase."

Matched Posts:

A: "Scarlet suitcase" (Score: 0.82)

B: "Crimson luggage" (Score: 0.81)

Output: "There are multiple posts related to your detail. Check the posts by Alice and Bob."

5.5.5 Complaint Handling

At first the system should determine if the extracted information corresponds to a complaint handling. To identify and extract user information, the system uses regex patterns.

```
/\b(file|register|submit|raise|lodge|)\s+(a|an|the)?\s*(complaint|issue|problem|grievance|case)\b/i
```

Example: "I want to lodge a complaint"

This ensures that the input is actually about a complaint and further processing starts. The chat bot requires to know who is trying to lodge the complaints, so it tends to know the identity of the user trying to make the complaint.

5.5.5.1 Email and LTP Request

After confirming that the query is about complaint, the bot responds by asking email and LTP to validate the user to ensure who has made the report. The email and LTP is input by users in the chat which is processed further.

5.5.5.2 Email and LTP Extraction

After user inputs the email and LTP, their extraction is handled dynamically.

Regex Pattern: `/[\w.-]+@[\w.-]+\.\w+\/`

Example: Message: "My email is test@example.com"

Extracted Email: test@example.com

For LTP:

Regex Pattern: `/\b\d{5}\b/`

Example: Message: "My LTP number is 12345"

Extracted LTP: 12345

Eg query: my email is ramshyam@gmail.com and ltp is 23456

Extracted email: ramshyam@gmail.com

Extracted LTP: 23456

Now, these extracted details are further proceeded to check with the passengers record in the database. The report collection of the database is checked, each and every records are checked through using loops.

5.5.5.3 Email/LTP Validation

After extracting email and LTP, the system verifies these details by querying the database.

Database Lookup: Query = {email, LTP}

- i. If a match found, the system proceeds to complaint submission.
- ii. If verification fails, the user is prompted to re-enter valid details.
- iii. If the answer is beyond this handler, other handles handle the query.

5.5.5.4 Complaint Acceptance and Save In DB

If verification succeeds, the system collects the complaint message and stores it in the database.

5.5.5.5 Complain Confirmation

- i. If successful: "Your complaint has been submitted. We will look after it."
- ii. If an error occurs: "There was an error submitting your complaint. Please try again later."

5.5.6 General Query

If the query is not related to the report, post and complain then the query is general query. A general query is when a user is seeking information, guidance, or clarification. To identify and extract user information, the system uses regex patterns.

`/\b(how|what|where|when|why|who|help|support|assistance|info|information|details|guidance|explain|clarify|tell me| understand |know about |question)\b/i`

Example: "I need assistance with my lost bag"

This ensures that the input is actually about general query and further processing starts as the Regex pattern is not of post, report or complaint query.

5.5.6.1 Dialogue Flow API Call

It is implemented using Dialogflow intent detection system using Node.js. It enables communication between user and google Dialogflow's natural language processing API. It processes user queries and returns appropriate responses based on trained intents. We must give an access to the account through which we have created the dialogflow agent.

We must enter:

- i. Project ID: It is the id we obtain after we create a project in dialog flow.
- ii. Google Application Credential: It is a file in JSON format which we import in the backend that handles the dialogflow API call.

These details give the account the admin access or permitted user access as we have customized.

5.5.6.2 Intent Detection

The API analyzes the user's input to determine the intent behind the query. This involves understanding what the user wants to achieve or the action they want to perform. It maintains the context of the conversation to ensure that follow-up queries are understood in relation to previous interactions. This involves natural language understanding (NLU) techniques.

5.5.6.3 Response Generation

Based on the detected intent and context, the API generates an appropriate response. This involves selecting the most relevant information or action to address the user's query. The response is written in HTML format, which is imported by our JS code in a fully rendered form.

5.5.6.4 Response Formatting

The generated response is formatted by structuring and presenting the system's reply to a user query in a way that is clear, organized, and easy to understand. Proper

response formatting ensures that the information is delivered effectively, enhancing the user experience.

5.5.6.5 Response to User

The final response is sent back in an understandable form to user. The response can be regarding the answers to basic query regarding various cases as well as navigations regarding the navigation which are rendered through HTML links.

5.6 Software Development Life Cycle Model

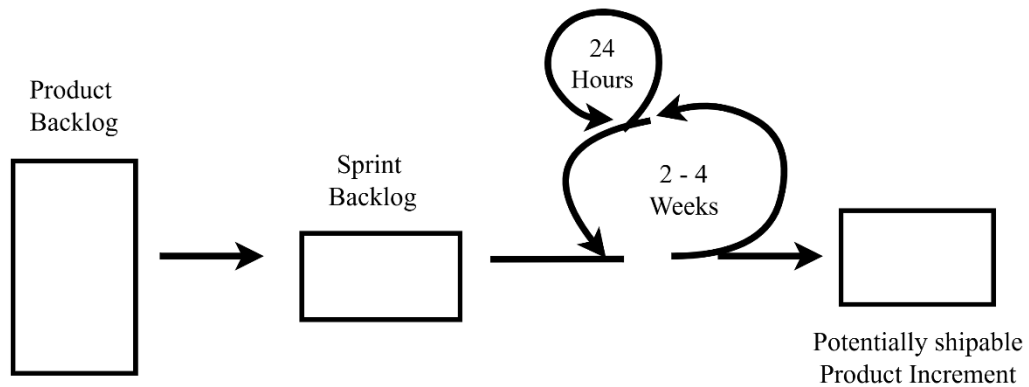


Figure 5. 6: SCRUM Framework [13]

Figure 5.6 shows an iterative and incremental approach for software development using agile framework i.e. Scrum. It shows iterative process product backlog, moving to sprint backlog followed by sprints and daily stand ups and finally producing potentially shippable product.

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Benefits of agile include its ability to help teams in an evolving landscape while maintaining a focus on the efficient delivery of business value. The collaborative culture facilitated by agile also improves efficiency throughout the organization as teams work together within the framework.

Scrum is a lightweight adaptive framework that helps people, teams and organizations generate value through adaptive solutions for complex problems and minimize the need for prescribed events used in Scrum. All events are time-boxed. Once a Sprint begins, its duration is fixed and cannot be shortened or lengthened. The remaining events may end whenever the purpose of the event is achieved, ensuring an appropriate amount of time is spent without allowing waste in the process.

5.7 Tools Used

The system is built on a 64-bit computer operating with Windows 11.

Software Components

- i. VS code: A text editor popular among developers for fast performance, built in debugging capabilities extensive library of extensions.
- ii. MongoDB Compass: A software for generating database at local computer for visualizing the application interaction with database before connecting with actual MongoDB database.
- iii. Figma: A user-friendly software for designing layout of the web application before the frontend development actually starts.
- iv. Git hub: For collaborating with team. A common code sharing platform.

Technologies:

- i. HTML/CSS/React: To build the web pages and interfaces for the website.
- ii. React-Router: Used for routing in multiple pages of multi-page website.
- iii. Chart JS: A framework of JavaScript used to generate pie charts and bar graphs.
- iv. Express: To use JavaScript in backend development for server-side rendering and database access.
- v. Nodemailer: A library of Node.js for sending emails and OTPs.
- vi. Natural: Natural Language Processing (NLP) library for Node.js. It provides various NLP functionalities such as tokenization, stemming, classification, phonetics, and more.
- vii. Bcrypt: Bcrypt is a popular password hashing library used in Node.js for securely storing passwords.

CHAPTER 6: RESULT AND ANALYSIS

The report, post and claim operations are successful with real time alert and proper OTP handling and appropriate hashing mechanisms to secure the data of the user as well as the admin.

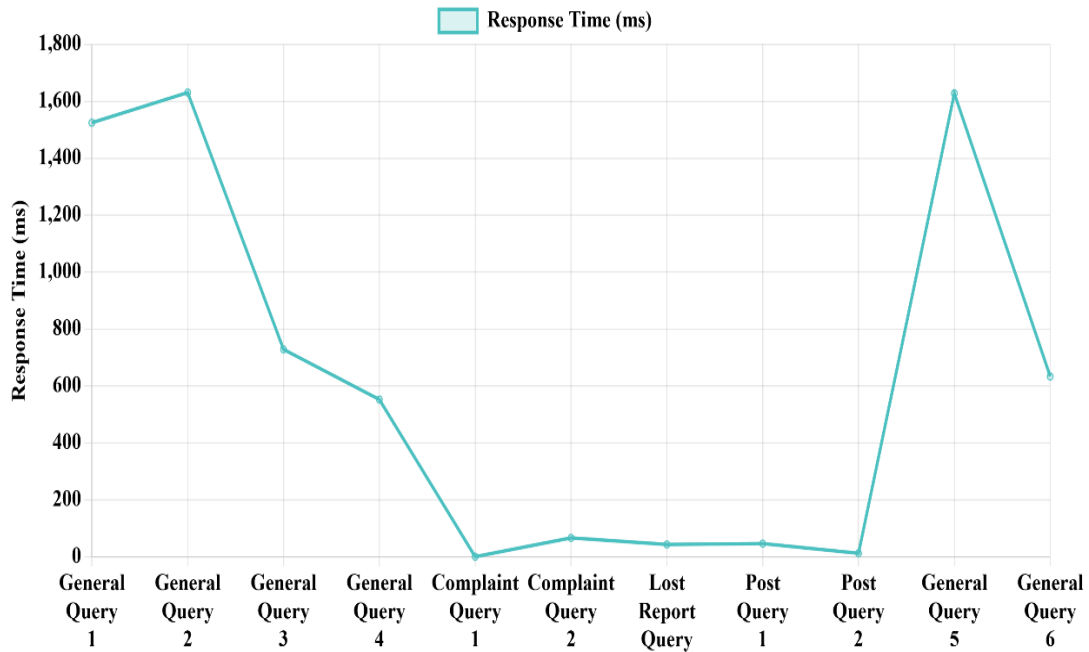


Figure 6. 1 Response time of several queries asked

Figure 6,1 shows handling cases of report, post, general as well as complaint query. Latency for general query, the API call is performed while for the fulfillment-based query we access our database directly so there is certain delay for response to general query i.e. approximately 700 to 1000ms. This variability can depend on the complexity of the query and the system's current load.

The post recommending or finding feature (i.e. post query) is not accurate due to some error margins in the stemmer and logically/mathematically correct yet practically incorrect meanings.

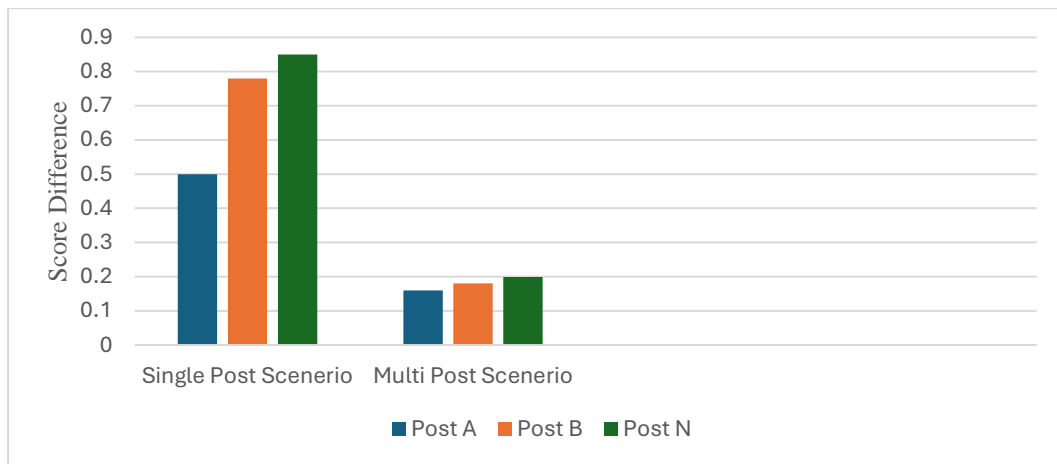


Figure 6. 2 Single And Multi-Post Scenario

Figure 6.2 shows single and multi-post scenario. If the score difference is less than 0.2 which is threshold, all the posts below/ less than threshold is recommended or shown. But main issue here is the error of porter stemmer which is 5 to 15%. And hence there can be certain unavoidable error in few cases. This is mainly because in some situation, there is over stemming and in some other cases there is under stemming leading to error. And in some cases, the entire meaning of the token is altered.

But, implementing this similarity calculation and threshold comparison feature led to significant improvement in the post recommendation/finding feature.

The table below shows comparison between the SITA Air Tracker And Luggage Tracking System we developed:

Criteria	SITA Air Tracker	Luggage Tracking System
Usability and complexity	Advanced software used by airport authority only and require specific trained staff to use.	Simple software designed to be used at user level and does not require specific training.
Updates	Not real time updates and alert, delay might be experienced in response	Real time and immediate mailing to the user.

	as it is controlled at staff level in airport.	
User support	No proper address to customer queries and untimely response.	Integrated chatbot to address the query and quick response to them as well as perform fulfillment tasks like checking status, recommending posts and lodging complaints.
System Dependency	Hardware components also used in the tracking process which has chances of failure.	A simple fully software-based system used by user, no concern for hardware .
Cost	High installation and maintenance cost due to involvement of both hardware and advanced software components	Only software component, comparatively cheaper to install and use.

Table 6. 1 Comparison of Existing SITA Air Tracker And Luggage Tracking System

Our objective was to develop a web application to ensure easy finding and claiming the luggage and to make interactive chatbot to answer user queries and fulfill some basic user requirements which we have achieved successfully.

CHAPTER 7: CONCLUSION

7.1 Conclusion

Hence, our project has effectively achieved its objectives of creating web application to make the process of finding and claiming the lost luggage with ease. We also integrated an assistant chatbot that helped users with queries and complaints. The system provides real time updates on all transactions and activities through email services. The hashing and OTP implementation enhanced the further security and prevents unauthorized or improper claim. The UI is quite simple and easy to use by users and the process is smoother. The chatbot also addresses most of the user queries and also generates the answers understandable by the general users. It is also capable enough to navigate users through pages and also receives complain which can be viewed by the admin. The admin can view all the database records and make changes too. They can also view the statistics in numerical as well as graphical manner. Hence, we have developed a fully functional web application as per our objectives.

7.2 Future Enhancements

Although our site is fully functional, there are some limitations of our websites such as non-real data and limited capability of chatbot. There are several future enhancements that we can implement in future to make the system more advanced, functional, robust and reliable. The future enhancement can be:

- i. Although NLP is being used by the chatbots, it can further be enhanced/improved to learning based using AI/ML.
- ii. The data can be made valid based on real airport data which is possible only when used on market/real use.
- iii. Instead of email alert, we can implement SMS to make the message passing more reliable.

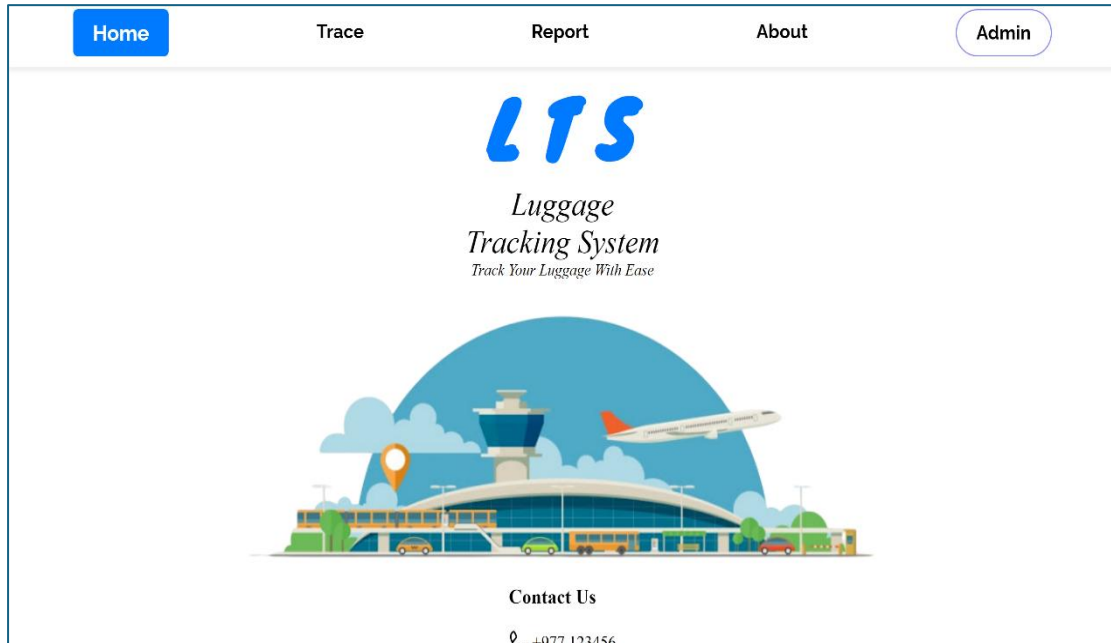
REFERENCES

- [1] G. E. R. Campos, World Tracer Basic Training, USA: CGHD Publication, 2012.
- [2] A. Benassi, "Aviation Information Data Exchange," 2022. [Online]. Available: <https://iata.org>. [Accessed 3 Nov 2024].
- [3] B. Bedford, "Air Services Agreement," 20 May 2002. [Online]. Available: <https://www.lawinsider.com>. [Accessed 25 Nov 2024].
- [4] D. Bererd, "SAS workflow," Dec 2012. [Online]. Available: <https://www.flysas.com>. [Accessed 29 Nov 2024].
- [5] A. F. Muhammad, D. Susanto, A. Alimudin, F. Adila, M. H. Assidiqi and S. Nabhan, "Developing English Conversation Chatbot Using Dialogflow," in *2020 International Electronics Symposium (IES)*, Surabaya, 2020.
- [6] M. P. Zillman, Bot and Intelligent Agent Research Resources, Domain Publications, 2022.
- [7] K. Gaddam, Building Bots with Microsoft Bot Framework, Birmingham, U.K., and Mumbai, India: Packt Publishing Ltd., 2021.
- [8] E. Adamopoulou and L. Moussiades, "An overview of chatbot technology," in *Artificial Intelligence Applications and Innovations*, Cham, Springer, 2020, pp. 429-438.
- [9] S. Punjabi, V. Sethuram, V. Ramachandran, R. Boddu and S. Ravi, "Chat bot Using API: Human to Machine Conversation," in *2019 Global Conference for Advancement in Technology (GCAT)*, Bangalore, India, 2019.
- [10] R. Mihalcea, H. Liu and H. Lieberman, "NLP (Natural Language Processing) for NLP (Natural Language Programming)," in *Computational Linguistics and Intelligent Text Processing. CICLing 2006*, Berlin, Heidelberg, Springer, 2006, pp. 319-330.

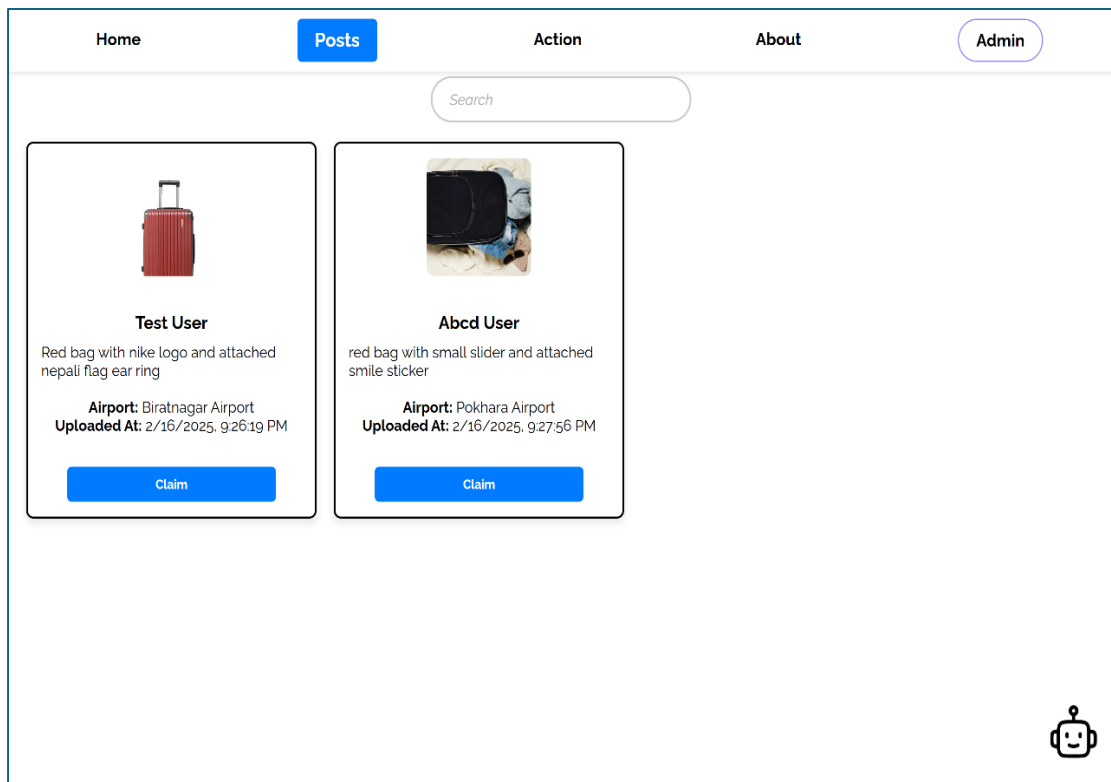
- [11] T. Lalwani, S. Bhalotia, A. Pal, V. Rathod and S. Bisen, "Implementation of a chatbot system using AI and NLP," *International Journal of Innovative Research in Computer Science and Technology (IJIRCST)*, vol. 6, no. 3, pp. 1-6, 2018.
- [12] T. Korenius, J. Laurikkala and M. Juhola, "Principal component analysis, cosine and Euclidean measures in information retrieval," *Information Sciences*, vol. 177, no. 22, pp. 4893-4905, 2007.
- [13] S. Prodes, "Scrum Images – Browse 33,371 Stock Photos, Vectors, and Video," Adobe Stock, 2024. [Online]. Available: <https://stock.adobe.com/search?k=scrum>. [Accessed 26 Nov 2024].

APPENDICES

APPENDIX A: HOMEPAGE



APPENDIX B: POST PAGE



APPENDIX C: REPORT PAGE

POST FORM

[Home](#)[Posts](#)[Action](#)[About](#)[Admin](#)

[Post Luggage](#)[Report Lost](#)

Post Luggage

Full Name

Ticket Number

Phone Number

Select Airport


Choose File

No file chosen

Email

More Details (eg: color, zip, logo or some specific feature. Be more precise)

Submit



REPORT FORM

[Home](#)[Posts](#)[Action](#)[About](#)[Admin](#)

[Post Luggage](#)[Report Lost](#)

Report Lost

Full Name


LTP Number

Phone Number

Email

Select Airline

Submit



APPENDIX D: ADMIN LOG IN PAGE

[Home](#) [Posts](#) [Report](#) [About](#) [Admin](#)

Log In As Admin

Log In

APPENDIX E: ADMIN PANEL

Home

Posts

Action

About

Admin

sambadkhatiwada8@gmail.com

Log Out

Pending Reports

Posts

Success

Users

Post Claimed

Stats

Statistics

998

Total Data

1

Pending Reports

5

Active Posts

2

Complaints

2

Successes

6

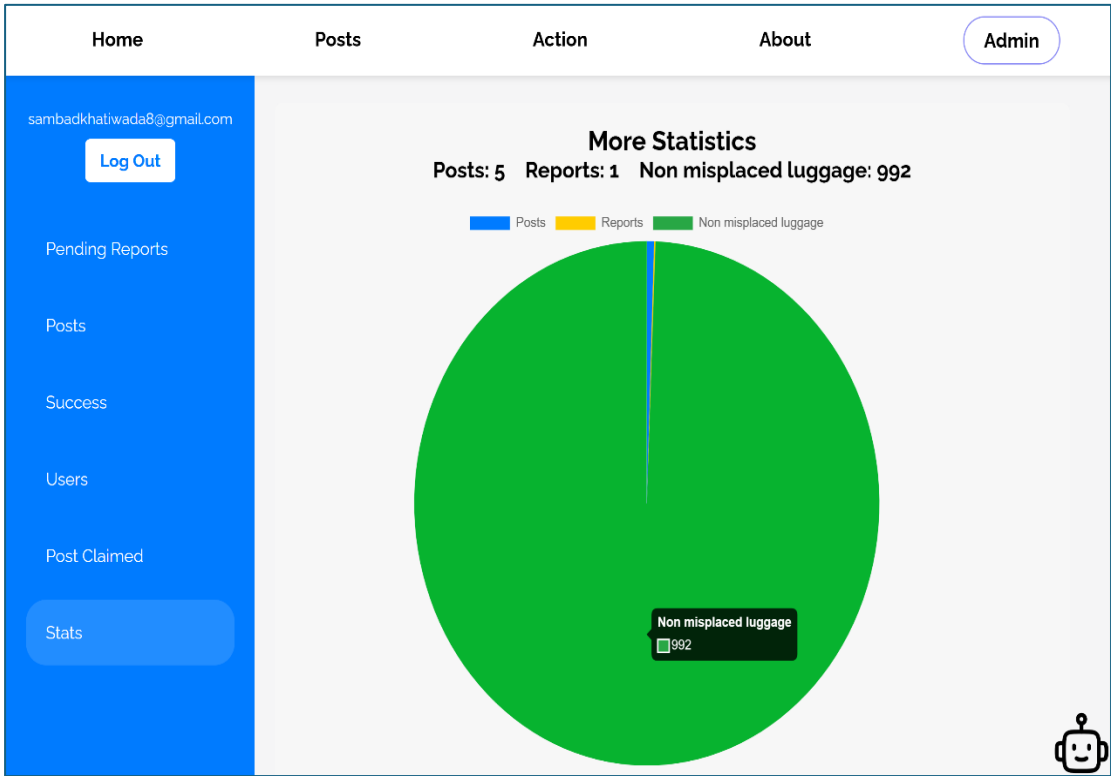
Post Claims

View Complaints

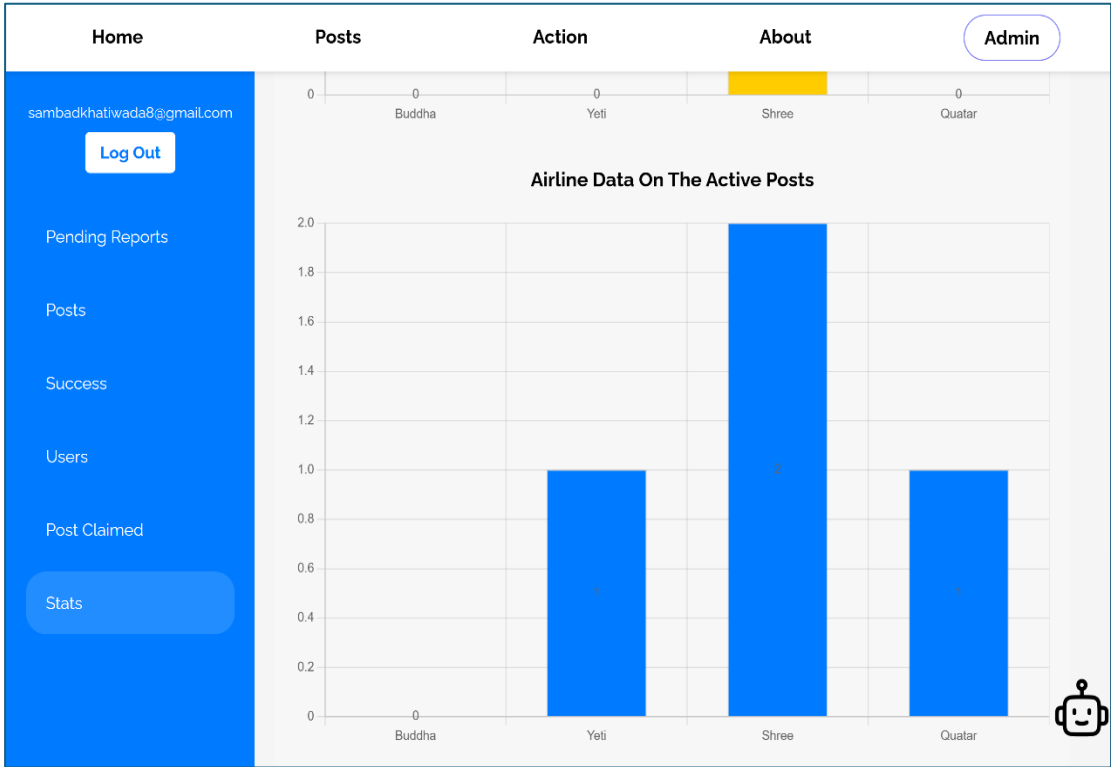
More Statistics

Posts: 5 Reports: 1 Non misplaced luggage: 992

APPENDIX F: PIE CHART REPRESENTATION



APPENDIX G: BAR CHART REPRESENTATION



APPENDIX H: BASIC QUERY TO THE CHATBOT

Chat with Us

✕

how to post

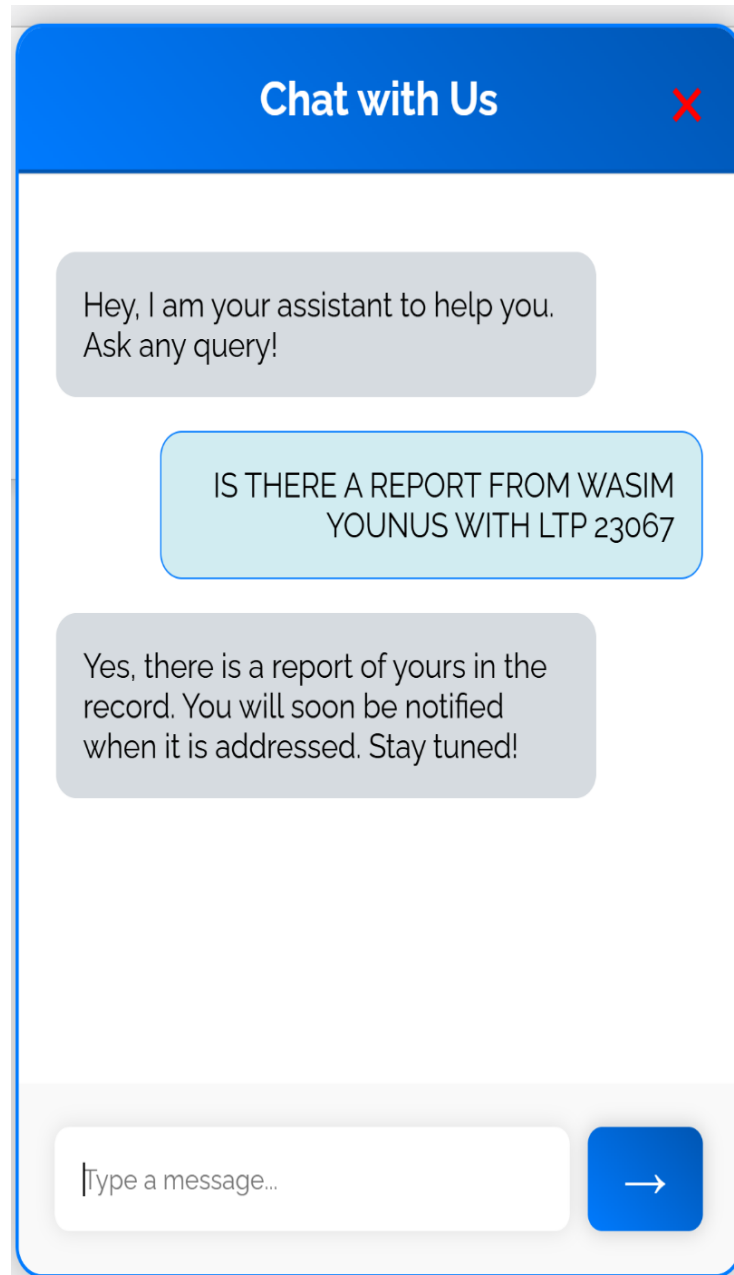
To Post:
Browse Action: [here](#)
1. Select the **Post** form.
2. Fill out the form with correct details.
3. You will be emailed regarding the post match if the pending report is already present or the post confirmation mail.
You can confirm your post at Post section [here](#)

Note: After the post is claimed, both you and the claimer will be notified in real-time regarding the information and guidelines.
Ask any query if you have any.

Type a message...

→

APPENDIX I: REPORT QUERY TO THE CHATBOT



APPENDIX J: COMPLAINT QUERY

Chat with Us

complaint

Sure, what is your email and LTP?

i have email
minorprojectsspl@gmail.com and
my ltp is 23067

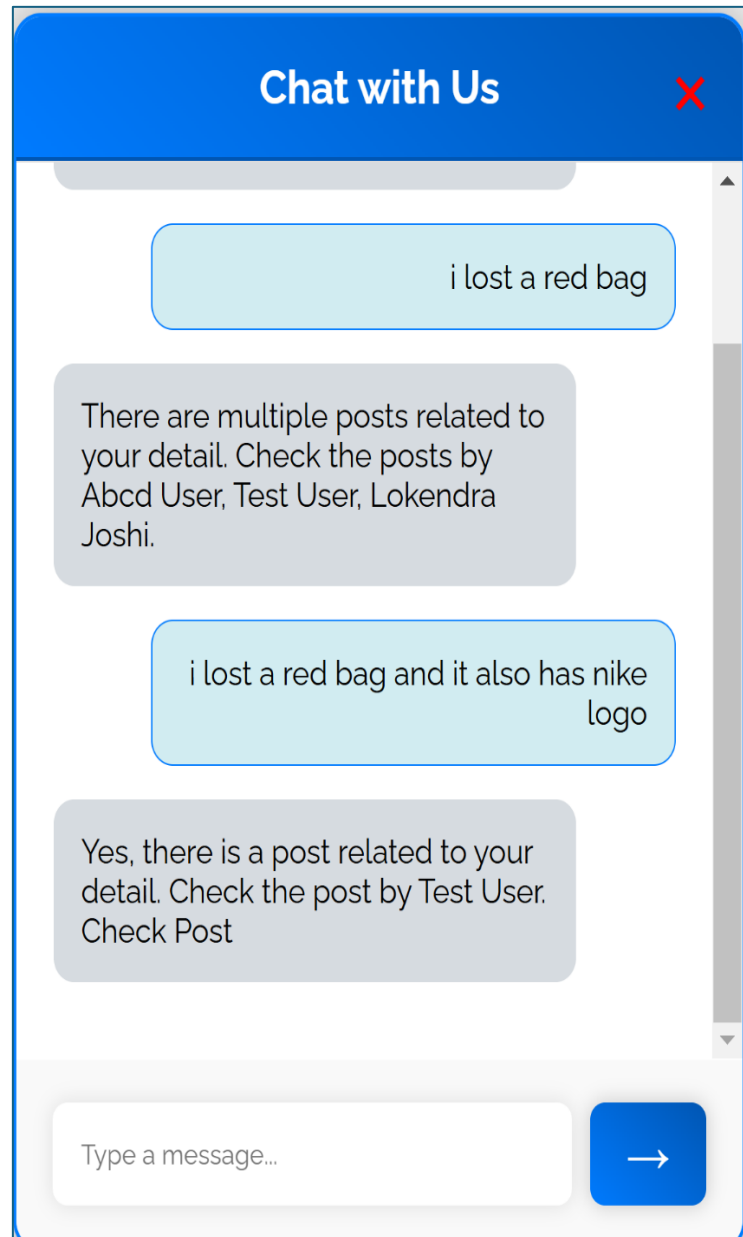
Your email and LTP are verified.
Please enter your complaint.

my report is not being registered

Type a message...

→

APPENDIX K: POST QUERY



APPENDIX L: CODE SNIPPET OF DIALOGFLOW API CONFIGURATION

```
process.env.GOOGLE_APPLICATION_CREDENTIALS = path.join(__dirname, 'sam-fdbo-e0a002caab9b.json');

const projectId = 'sam-fdbo';
const languageCode = 'en';

const sessionClient = new dialogflow.SessionsClient();

async function detectIntent(message, sessionId) {
  const sessionPath = sessionClient.projectAgentSessionPath(projectId, sessionId);
  const request = {
    session: sessionPath,
    queryInput: {
      text: {
        text: message,
        languageCode: languageCode,
      },
    },
  },
};
```

APPENDIX M: CODE SNIPPET OF COSINE SIMILARITY IMPLEMENTATION

```
const cosineSimilarity = (vec1, vec2) => {
  const dotProduct = vec1.reduce((sum, value, index) => sum + value * vec2[index], 0);
  const magnitudeVec1 = Math.sqrt(vec1.reduce((sum, value) => sum + value * value, 0));
  const magnitudeVec2 = Math.sqrt(vec2.reduce((sum, value) => sum + value * value, 0));

  if (magnitudeVec1 === 0 || magnitudeVec2 === 0) {
    return 0;
  }

  return dotProduct / (magnitudeVec1 * magnitudeVec2);
};
```

APPENDIX N: NODEMAILER CONFIGURATION

```
const transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: 'minorprojectsspl@gmail.com',
    pass: 'idhy nenw vghh wdus'
  }
});
```

APPENDIX O: SAMPLE OF REGEX VARIATIONS

```
const lostReportPatterns = [
  /my name is\s+([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})[\\W\s]*(?:what is|tell|give) my report statu
  /my name is\s+([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})[\\W\s]*(?:what is|tell) my report status/i,
  /my name is\s+([a-z\s]+?)\\W\s*(?:what is|tell) my report status/i,
  /has\s+([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})[\\W\s]*reported/i,
  /name\s+([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})[\\W\s]*/i,
  /i am\s+([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})[\\W\s]*have i reported/i,
  /([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})[\\W\s]*/i,
  /name\s+([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /i am\s+([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})[\\W\s]*have i reported/i,
  /i am\s+([a-z\s]+?),?\s*have i reported any(?:lost\s+)?luggage\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /i am\s+([a-z\s]+?),?\s*did i file a report\s*(?:under\s*)?ltp\s*(?:is\s*)?(\\d{5})/i,
  /i'm\s+([a-z\s]+?),?\s*did i submit a report for(?:lost\s+)?luggage\s*(?:under\s*)?ltp\s*(?:is
  /has\s+([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})[\\W\s]*reported/i,
  /have\s+([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})[\\W\s]*reported/i,
  /is there(?:any\s*)?report\s*([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /is there a record of(?:lost\s+)?luggage\s*([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /was a report filed\s*([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /did\s+([a-z\s]+?)\s*file a report\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /can you find a report for(?:lost\s+)?luggage\s*([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /is there a(?:lost\s+)?luggage report\s*with\s*([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /confirm if a report was made\s*([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /find a(?:lost\s+)?luggage report\s*([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /does\s+([a-z\s]+?)\s*have a(?:lost\s+)?luggage report\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /i am\s+([a-z\s]+?),\s*have i reported any(?:lost\s+)?luggage\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /my name is\s+([a-z\s]+?),\s*did i submit a report for(?:lost\s+)?luggage\s*ltp\s*(?:is\s*)?(\\
  /i'm\s+([a-z\s]+?),\s*any(?:lost\s+)?luggage report\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /check for report\s*([a-z\s]+?)\s*ltp\s*(?:is\s*)?(\\d{5})/i,
  /lost luggage report,\s*ltp\s*(?:is\s*)?(\\d{5}),\s*name\s+([a-z\s]+)/i,
];
```