MINOR PROJECT

MID SEM REPORT

on

SPARC : Semantic and People-centric Analysis for Rapid Complaints

Submitted By :

Name	**Roll No**	**Branch**
Harsh Bansal	A25305221034	B.TECH. CSE
Siddharth Singh	A25305221018	B.TECH. CSE

Under the guidance of

Dr. Sachin Chaudhary

Assistant Professor

Amity School of Engineering and Technology

Amity School of Engineering and Technology (ASET)

Amity University, Mohali

2023-24

Approved By

(Dr. Harvinder Singh)
Project Mentor

(Dr. Harvinder Singh)
Project Coordinator

ABSTRACT

This project endeavors to innovate and optimize complaint management processes within enterprises through the integration of advanced Machine Learning techniques and Microservices Architecture. Termed SPARC (Semantic and People-centric Analysis for Rapid Complaints), the system employs sophisticated Semantic Analysis Models to interpret and categorize complaints swiftly and accurately. Leveraging Microservices Architecture ensures scalability, modularity, and flexibility, allowing seamless integration of various components. The implementation encompasses diverse technologies, including Python, Docker, and Microsoft Azure Cognitive Services, tailored to address specific facets of complaint handling, such as input processing, semantic analysis, and complaint dispatch. Notably, the utilization of PostgreSQL for database transactions, coupled with Twilio for complaint dispatch via WhatsApp, underscores the project's commitment to practical usability and real-world applicability. Through meticulous design and implementation, SPARC aims to redefine complaint management paradigms, offering enterprises a comprehensive and efficient solution to address the burgeoning volume of complaints effectively.

Keywords: Semantic Analysis, Microservices Architecture (MSA), Scikit Learn, Flask, Docker

TABLE OF CONTENTS

Contents

1 Preface **5**

1.1 Background and Motivation 5

1.2 Bridging Research and Product Development 6

2 Introduction **6**

3 Objective **8**

4 Design **8**

4.1 Microservices Architecture (MSA) 8

4.2 SPARC Design 8

5 Implementation **9**

5.1 API Interface 9

5.2 MS1 - Input Processing and Semantic Analysis Pipeline 10

5.2.1 Microsoft Azure Cognitive Services for Translation and Multi-modal Input Handling 10

5.2.2 Semantic Analysis using TF-IDF Vectorization 10

5.3 MS2 - Complaint Processing and Delivery 11

5.3.1 Complaint Category to POC Mapping 12

5.3.2 Database Transactions 12

5.3.3 Dispatch using Twilio 12

5.3.4 Complaint Status Update 12

A APPENDIX I PROJECT CODE **14**

A.1 Directory Structure of the Project 14

A.2 Microservice 2 - Complaint Processing and Delivery 14

A.2.1 `app.py` 14

A.2.2 Helper - `message.py` 17

A.2.3 Helper - `twilio_client.py` 18

A.2.4 DB Model - `model.py` 18

A.2.5 Dockerfile 19

LIST OF FIGURES

List of Figures

1 *Complaints related to Banking Received by Banking Ombudsman/ORBIO[1]* 6
2 *Research to Product Development* 7
3 *Architecture Diagram of SPARC* 9
4 *Sample Input Form designed to simulate Complaint Submission* 11
5 *Mockup of complaint details delivery and confirmation* 12

1 Preface

This project report has been crafted to partially meet the requirements for the Minor Project of the B.Tech. Computer Science and Engineering program (Semester VI) during the academic year 2023-2024 at Amity University Punjab.

For preparing the Project Report, we have conducted extensive research on all major as well as minor aspects of the project, and consulted our mentor Dr. Sachin Chaudhary as well as other experienced people from the industry to ensure delivery of accurate and authentic information in this report.

The report commences with an exploration of the project's background, providing context for the subsequent discussion. This is followed by the project introduction, elucidating its purpose and significance. Next, the objectives, delineating the anticipated outcomes by the project's culmination, are presented. Subsequently, the design section comprehensively details the project's architectural framework. Following this, the implementation section outlines the practical execution of the project. Finally, the report concludes by summarizing key findings and implications.

1.1 Background and Motivation

In the contemporary landscape of organizational operations and law enforcement, the significance of feedback mechanisms cannot be overstated. For businesses like IBM[2], customer complaints serve as invaluable insights into the functioning of their various modules, establishing a direct link between the organization and its users. Similarly, police departments rely on mechanisms such as FIRs, DDRs, and verbal reports to track offenses and cater to the needs of the community effectively. However, with the evolving nature of offenses and the increasing volume of complaints, traditional systems have encountered challenges, particularly in terms of efficiency and responsiveness. According to a report from the Times of India, the National Commission for Women (NCW) received a staggering 28,811 complaints of crimes against women last year, with approximately 55 percent originating from Uttar Pradesh. The category with the highest number of complaints was "right to dignity," encompassing harassment beyond domestic violence, which accounted for 8,540 cases, as per NCW data[3]. The case is similar for the banking sector with rising number of frauds, which can be visualized through Figure 1. Delays in handling complaints, often attributed to cumbersome bureaucratic processes, necessitate a reevaluation of existing systems and the integration of modern technologies to streamline operations. Recognizing the imperative for innovation, the Chandigarh Police partnered with Infosys to host a National Level Hackathon. This collaborative effort aimed to solicit creative solutions to enhance the functionality of the city's police system, addressing key concerns and identifying opportunities for improvement.

Among the diverse challenges addressed during the hackathon, one significant focus area emerged: the development of a system leveraging artificial intelligence (AI) for the rapid resolution of complaints. After validating our idea at this National Level Hackathon, we finally decided to convert it into a fully-functional working software. The resulting project, SPARC (Semantic and People-Centric Analysis for Rapid Complaints), represents a pioneering endeavor to revolutionize police complaint handling. Implemented using industry-standard mechanisms, including a Microservice Architecture for scalability and reliability, SPARC embodies a modern approach to law enforcement operations. Leveraging advanced technologies and third-party services available through platforms such as Azure and AWS, SPARC sets a precedent for the integration of AI in police complaint handling.

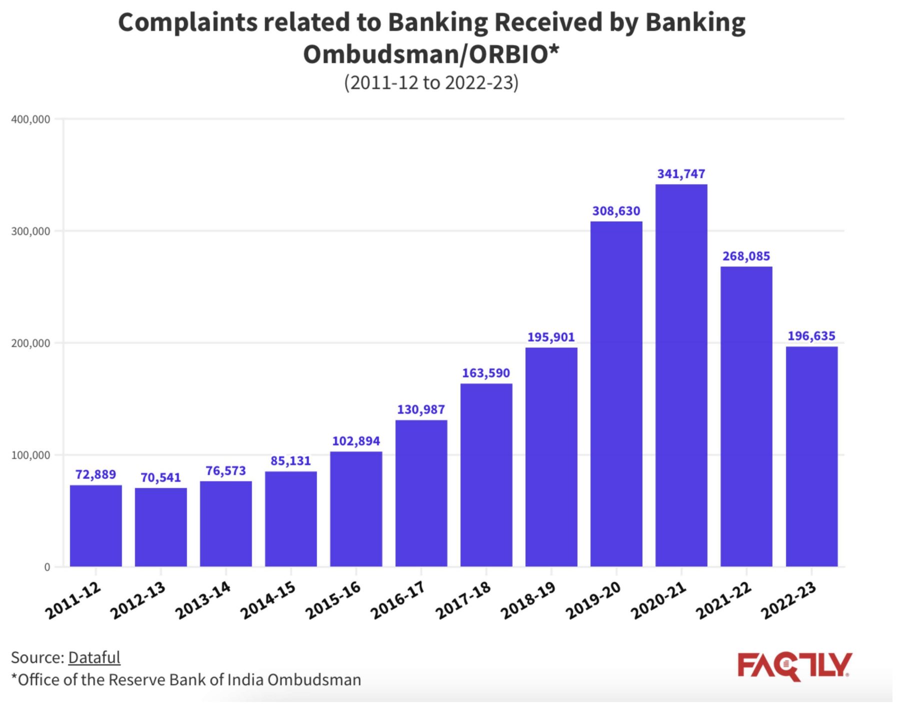


Figure 1: *Complaints related to Banking Received by Banking Ombudsman/ORBIO[1]*

1.2 Bridging Research and Product Development

Beyond academic research lies a critical gap between conceptual innovation and tangible impact. While engaging in research projects is commonplace for students, it's evident from experience that research outcomes often don't directly translate into actionable products. There exists a comprehensive process, illustrated in Figure 2, to bridge this gap and transform research findings into real-world solutions capable of effecting meaningful change. In the field of Artificial Intelligence and Machine Learning (AIML), this involves converting intricate models into functional backends and frontends, integrating essential features like authentication, testing, analytics, and monetization.

This project serves as an extension of our research endeavors in the domain of Semantic Analysis. It leverages the breadth of knowledge acquired through our Computer Science coursework, offering us a unique opportunity to explore the journey from groundbreaking research to impactful products and businesses. By bridging theory with practice, this project ensures that our research isn't just academically significant but also practical and relevant to end-users. Through this endeavor, we gain valuable insights into the intricacies of product development and the potential of research to drive real-world innovation.

This preface sets the stage for the exploration and analysis presented in this research paper/project report. Through the examination of SPARC's development process, implementation, and outcomes, we aim to contribute to the ongoing discourse surrounding the optimization of law enforcement processes in the digital age.

2 Introduction

SPARC: Semantic and People-centric Analysis for Rapid Complaints, represents a pioneering initiative aimed at revolutionizing complaint management within enterprises. In today's fast-paced world, organizations encounter an ever-increasing influx of complaints from diverse

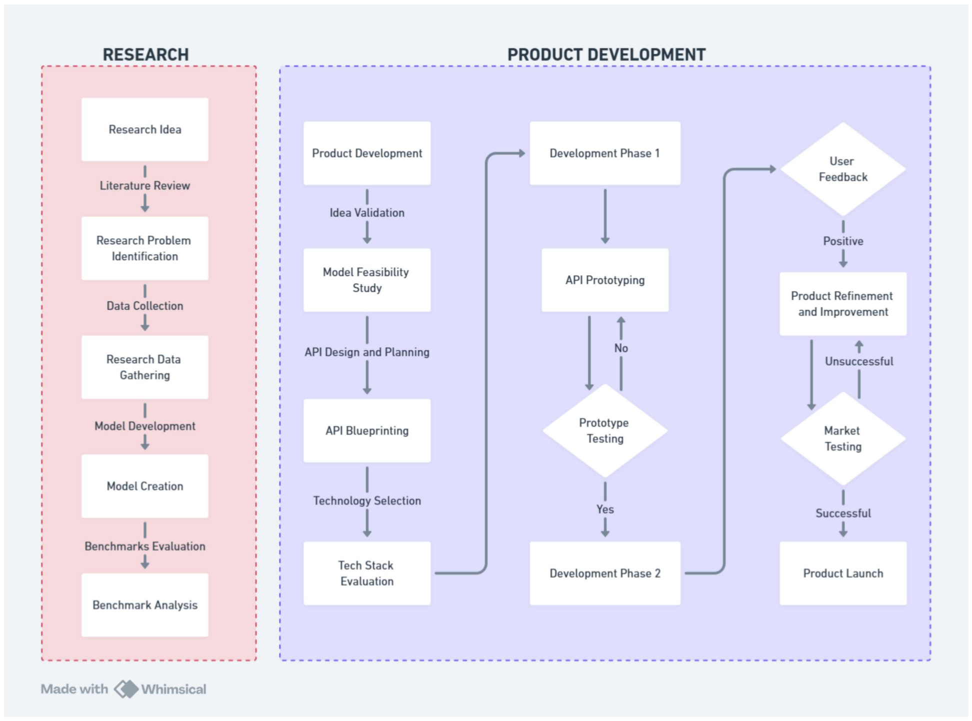


Figure 2: *Research to Product Development*

sources. These grievances serve as invaluable feedback channels, shedding light on operational inefficiencies, customer dissatisfaction, and emerging challenges. However, navigating through this deluge of feedback presents a significant challenge, often leading to bottlenecks, delays, and missed opportunities for swift redressal.

In response to these challenges, SPARC emerges as a beacon of innovation, offering enterprises a comprehensive solution for efficient and automated complaint handling. At its core lies the fusion of cutting-edge Machine Learning techniques with advanced Semantic Analysis systems, empowering organizations to decipher the intricate nuances of customer feedback and swiftly categorize complaints based on their underlying semantics. By leveraging Supervised Learning-based Models and meticulously curated datasets, SPARC pioneers a new paradigm in complaint classification, facilitating actionable insights and informed decision-making.

Yet, SPARC's vision extends beyond mere research and development. Unlike traditional academic endeavors focused solely on theoretical exploration, SPARC endeavors to bridge the gap between theory and practice, translating complex AI algorithms into tangible solutions for real-world challenges. Central to this endeavor is the development of a robust and user-friendly API-based integration, tailored to the unique needs and operational workflows of enterprises. Drawing inspiration from industry best practices, SPARC adopts a Microservices Architecture, dividing the project into modular components that seamlessly interact with one another, ensuring scalability, flexibility, and ease of deployment.

Through a meticulous blend of innovation, collaboration, and technological prowess, SPARC seeks to redefine the landscape of complaint management, empowering enterprises to navigate the complexities of customer feedback with confidence and efficiency. As we embark on this transformative journey, the following sections of this report delve deeper into the intricate workings of SPARC, elucidating its design, implementation, and potential impact on the domain of complaint handling.

3 Objective

With a project timeline spanning across several months, the following objectives have been established to monitor progress. However, it's important to note that the project is not confined to these objectives alone; additional features may be incorporated as per evolving requirements during the course of development.

1. Construct a system (API) for automated classification of complaints to various departments.
2. Handling multi-modal inputs (audio, images, text-based, etc.) for added convenience of clients.
3. Named Entity Recognition (NER) for effective keyword-based information extraction, allowing extensive data analytics on collected data.

4 Design

Designing a project involves considering client requirements, anticipated application load, and deployment options, balancing factors like cost and resource availability.

4.1 Microservices Architecture (MSA)

Given the project's aim to efficiently handle large volumes of complaints from diverse sources and perform extensive data analysis, Microservices Architecture (MSA) was chosen. MSA structures an application as a collection of small, loosely coupled services, allowing for flexibility, modularity, and scalability[4, 5].

The traditional approach to software development, known as Monolithic Architecture (MA), was previously favored by major companies like Amazon and eBay. In MA, all functions are consolidated within a single application. While monolithic applications have certain advantages, such as simplicity in development, testing, and deployment, they pose challenges as they grow in complexity. As the application expands, the monolithic structure becomes unwieldy and difficult to manage and scale[6, 7].

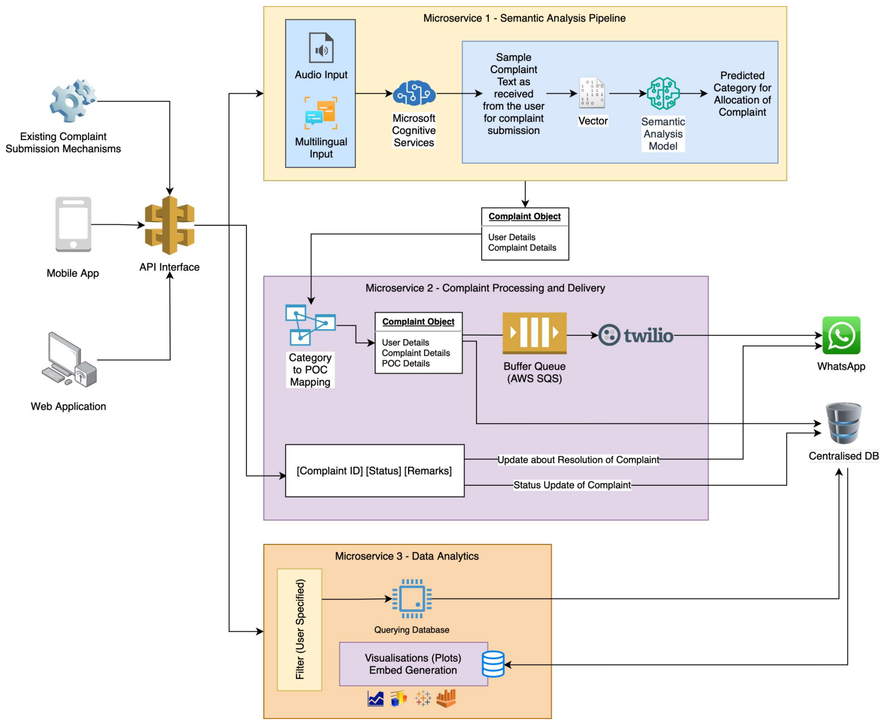
4.2 SPARC Design

The project is divided into three major microservices:

- **Microservice 1 - Input Processing and Semantic Analysis Pipeline**: Input Handling and using Machine Learning to decide the appropriate category for the Complaint Submitted.
- **Microservice 2 - Complaint Processing and Delivery**: Using appropriate datastructures to store complaint information, storing it in the database, and delivering the complaint to the concerned authority using Whatsapp.
- **Microservice 3 - Data Analytics**: For the clients to analyse influx and status of complaints at a scale, and make informed decisions for sustained business practices.

Figure 3 illustrates the architecture of the system and interconnections between these microservices.

Each microservice exposes multiple API methods for interaction, facilitating specific functionalities.



Microservice 1 - Semantic Analysis Pipeline
Audio Input
Multilingual Input
Microsoft Cognitive Services
Sample Complaint Text as received from the user for complaint submission
Vector
Semantic Analysis Model
Predicted Category for Allocation of Complaint
Existing Complaint Submission Mechanisms
Mobile App
API Interface
Web Application
Complaint Object
User Details
Complaint Details
Microservice 2 - Complaint Processing and Delivery
Category to POC Mapping
Complaint Object
User Details
Complaint Details
POC Details
Buffer Queue (AWS SQS)
twilio
WhatsApp
Centralised DB
[Complaint ID] [Status] [Remarks]
Update about Resolution of Complaint
Status Update of Complaint
Microservice 3 - Data Analytics
Filter (User Specified)
Querying Database
Visualisations (Plots) Embed Generation

Figure 3: *Architecture Diagram of SPARC*

A crucial component of the architecture is the API Interface, responsible for integrating loosely coupled microservices. It dynamically updates to track microservice addresses and manages communication between the client and individual microservices. Through API calls, complaints are directed to the appropriate microservice.

Detailed implementation strategies are discussed in the Implementation Section.

5 Implementation

This section delves into the comprehensive implementation methodology for each service outlined in the previous section. It entails a thorough examination of the technologies and tech stack employed.

The entire project is developed using the Python programming language, chosen for its versatility and extensive libraries available for machine learning and web development. To streamline workflow management and facilitate collaboration, project files are hosted and managed using GitHub, offering version control and collaboration features crucial for a project of this scale.

In addition to Python, Docker containers have been leveraged to ensure consistency and portability across different environments. Docker enables encapsulation of each microservice and its dependencies, simplifying deployment and ensuring reproducibility. Docker containers among various cloud providers has made them a trendy solution for developers [8]. Flask has been used as the API Framework, as it is a popular choice for integrating Machine Learning Projects with a front-end or client-friendly interface [9].

5.1 API Interface

Build with Flask, the API interface serves as the central point of contact for all the entities involved in the entire project. It dynamically updates connections to various micro-services and is responsible for carrying out transactions between client and the services as well as individual services themselves.

The following pseudo-code explains the functioning of the API interface at a glance.

Algorithm 1 API Interface Pseudo-Code

```
Define Flask app

SERVICE_1_URL → "http://service1:5001"
SERVICE_2_URL → "http://service2:5002"
SERVICE_3_URL → "http://service3:5003"

Define route "/" {Health check}
    return "API is Running!"

Define route "/api/service1/complaint" with HTTP method POST
    Get request data
    Send POST request to SERVICE_1_URL + "/process_complaint" with request data
    if response status code is 400 (error) then
        return the response with error
    else
        Send POST request to SERVICE_2_URL + "/queue_complaint" with response data
        return the response (complaint details)
    end if

Run the Flask app on host "0.0.0.0" and port 5005 with debug mode enabled
```

When a complaint is received, it is directed to the API interface from various input sources such as a client's existing Complaint Portal, a dedicated app, or a website, all capable of making API requests. Subsequently, the API interface transfers the complaint data to SERVICE 1 for processing, where subsequent actions take place.

5.2 MS1 - Input Processing and Semantic Analysis Pipeline

This microservice, known as the Input Processing and Semantic Analysis Pipeline, serves as the second stage in the system's interaction with a complaint. It receives the entire form data from the API interface. Like all other services, Flask forms the foundation of this microservice, facilitating interaction with designated endpoints.

- **INPUT** : Complaint Data and User Data (in any language as text or an Audio File in Hindi)
- **OUTPUT** : Complaint Data, processed for different types of inputs and standardized in English; User Data and Predicted Category for the complaint allocation.

5.2.1 Microsoft Azure Cognitive Services for Translation and Multi-modal Input Handling

Utilizing the Microsoft Azure Cognitive Services Suite, developers can construct advanced AI applications using pre-built and customizable APIs and models. We harnessed the Translator and Speech-to-Text functionalities to manage multi-lingual and multimodal inputs from users. Standardization of all complaint descriptions to English was ensured for subsequent processing steps.

5.2.2 Semantic Analysis using TF-IDF Vectorization

A proprietary dataset comprising 505 complaints spanning various categories was curated for this task, originating from a Police-related Hackathon. These categories predominantly pertain to police and crime incidents.

Taking inspiration from the approach followed in a study by Sundaram *et. al.*[10], to transform complaint text into machine-interpretable numeric form, TF-IDF Vectors were employed for vectorization. This process yielded a feature space serving as input for subsequent models.

Subsequently, prediction of complaint categories was executed. Several models, including Logistic Regression and Support Vector Classifier, were tested, ultimately favoring a tuned Logistic Regression model with 77% accuracy.

Integration of this model and vectorizer with Flask was achieved using the *joblib* package, facilitating seamless interaction of Sci-kit Learn-based models with Python applications.

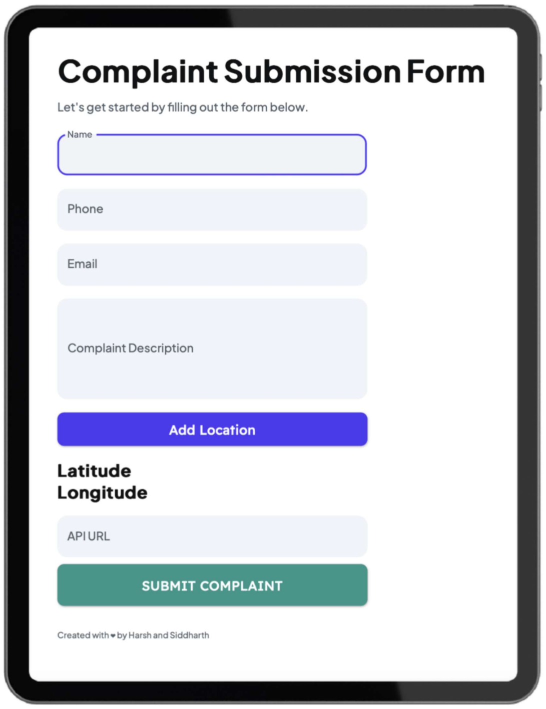

Complaint Submission Form
Let's get started by filling out the form below.
Name
Phone
Email
Complaint Description
Add Location
Latitude
Longitude
API URL
SUBMIT COMPLAINT
Created with ♥ by Harsh and Siddharth

Figure 4: *Sample Input Form designed to simulate Complaint Submission*

5.3 MS2 - Complaint Processing and Delivery

Following MS1, this microservice is engaged once MS1 completes processing the input. Upon receiving the predicted category of the complaint from MS1, the API interface triggers MS2 with relevant information for complaint allocation.

- **INPUT**:
  1. Complaint Data with Predicted Category and Complainant Details; and/or
  2. Status update for existing complaint.
- **OUTPUT**:
  1. Dispatched Complaint Data to designated Point of Contact / Authorities.
  2. Create an entry in the database for the Complaint
  3. Updated Complaint Status

5.3.1 Complaint Category to POC Mapping

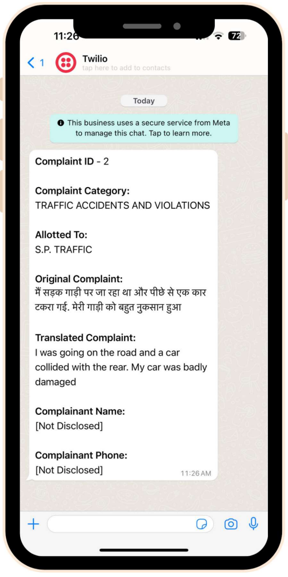
Upon categorizing the complaint, a nested hashmap is implemented to store contact details of the designated Point-of-Contact (POC). These details are utilized to dispatch the complaint via WhatsApp to the respective individual.

5.3.2 Database Transactions

Utilizing PostgreSQL as the database backend, once the POC is identified, the data is structured and stored in the database. Additionally, a ”STATUS” column is included to track the resolution status of each complaint. Leveraging *sqlalchemy*, the project is designed for easy modification of SQL structure and future migration to alternative DBMS with minimal adjustments.

5.3.3 Dispatch using Twilio

Twilio, an API interface, facilitates sending and receiving WhatsApp messages. The complaint data is formatted into a legible message and transmitted to the POC’s WhatsApp number via Twilio. The message includes a unique complaint ID, enabling the authorized recipient to update the complaint status with commands like ”RESOLVED.”


11:26
72
1
Twilio
tap here to add to contacts
Today
This business uses a secure service from Meta to manage this chat. Tap to learn more.

Complaint ID - 2

Complaint Category:
TRAFFIC ACCIDENTS AND VIOLATIONS

Allotted To:
S.P. TRAFFIC

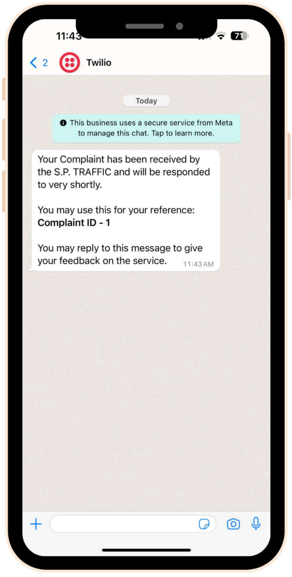
Original Complaint:
मैं सड़क गाड़ी पर जा रहा था और पीछे से एक कार टकरा गई. मेरी गाड़ी को बहुत नुकसान हुआ

Translated Complaint:
I was going on the road and a car collided with the rear. My car was badly damaged

Complainant Name:
[Not Disclosed]

Complainant Phone:
[Not Disclosed]
11:26 AM

(a) *To Authorities - Complaint Details*


11:43
71
2
Twilio
Today
This business uses a secure service from Meta to manage this chat. Tap to learn more.

Your Complaint has been received by the S.P. TRAFFIC and will be responded to very shortly.

You may use this for your reference:
Complaint ID - 1

You may reply to this message to give your feedback on the service.
11:43 AM

(b) *To Complainant - Confirmation*

Figure 5: *Mockup of complaint details delivery and confirmation*

5.3.4 Complaint Status Update

An added functionality enables the POC to update the complaint status directly via WhatsApp. By sending a message in the format *”[Complaint ID] [Status] [Remarks]”* on the same thread, the status is updated in the database, and a notification is dispatched to the complainant, informing them of the resolution status.

References

[1] "Data: Complaints to Ombudsman about Electronic Banking Increase While Those Related To ATM/Debit Cards Decrease — factly.in." `https://factly.in/data-complaints-to-ombudsman-about-electronic-banking-increase-while-those-related-to-atm-debit-cards-decrease/`.

[2] "Trend analysis — ibm.com." `https://www.ibm.com/docs/en/siffs/2.0.2?topic=surveillance-trend-analysis`.

[3] "28,811 complaints of crimes against women received in 2023, over 50 pc from UP: NCW data — India News - Times of India — timesofindia.indiatimes.com." `https://timesofindia.indiatimes.com/india/28811-complaints-of-crimes-against-women-received-in-2023-over-50-pc-from-up-ncw-data/articleshow/106453699.cms`.

[4] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2015.

[5] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation," *IEEE Cloud Computing*, vol. 4, no. 5, pp. 22–32, 2017.

[6] L. De Lauretis, "From monolithic architecture to microservices architecture," in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, IEEE, Oct. 2019.

[7] T. Salah, M. Jamal Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi, "The evolution of distributed systems towards microservices architecture," in *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 318–325, 2016.

[8] S. Malik, H. El-Sayed, M. A. Khan, and H. Alexander, "Application of containerized microservice approach to airline sentiment analysis," in *2020 14th International Conference on Innovations in Information Technology (IIT)*, pp. 215–220, 2020.

[9] A. Yaganteeswarudu, "Multi disease prediction model by using machine learning and flask api," in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 1242–1246, 2020.

[10] V. Sundaram, S. Ahmed, S. A. Muqtadeer, and R. Ravinder Reddy, "Emotion analysis in text using tf-idf," in *2021 11th International Conference on Cloud Computing, Data Science; Engineering (Confluence)*, IEEE, Jan. 2021.