**Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Computer Science & Engineering**

PROJECT REPORT

ON

ONLINE TICKETING HELP SAGE

Project-I

Department of Computer Science and Engineering
CHANDIGARH ENGINEERING COLLEGE JHANJERI, MOHALI

**In partial fulfillment of the requirements for the award of the Degree of
Bachelor of Technology in Computer Science & Engineering**

SUBMITTED BY :   **UNDER THE GUIDANCE :**

Name :- Gyanendra Pratap Singh (2230773)   Kunal Gagnega
Name :- Harmanpreet Kaur (2230774)   Assistant Professor,CSE
Name :- Ajay Kumar (2330582)
Name :- Leena (2230798)

MAY, 2025

**Affiliated to I.K Gujral Punjab Technical University, Jalandhar
(Batch: 2022-2026)**

DECLARATION

I, Gyanendra Pratap Singh… hereby declare that the report of the project entitled "Online Ticketing Chatbot" has not presented as a part of any other academic work to get my degree or certificate except Chandigarh Engineering College Jhanjeri, Mohali, affiliated to I.K. Gujral Punjab Technical University, Jalandhar, for the fulfillment of the requirements for the degree of B.Tech in Computer Science & Engineering.

(Student Signature with Date)   (Mentor Signature with Date)

Gyanendra Pratap Singh   **KUNAL GAGNEGA**

Univ. Roll No :- 2230773   Assistant Professor, CSE

Semester :- 6th

Signature of the Head of Department

(With Stamp)

ACKNOWLEDGEMENT

It gives me great pleasure to deliver this report on Project-I, which I worked on for my B.Tech in Computer Science & Engineering 3rd year, which was titled **"Online Ticketing Help Sage"** ". I am grateful to my university for presenting me with such a wonderful and challenging opportunity. I also want to convey my sincere gratitude to all coordinators for their unfailing support and encouragement.

I am extremely thankful to the HOD and Project Coordinator of Computer Science & Engineering at Chandigarh Engineering College Jhanjeri, Mohali (Punjab) for valuable suggestions and the heartiest cooperation.

I am also grateful to the management of the institute and Dr. Avinash, Director Engineering, for giving me the chance to acquire the information. I also appreciate all of my faculty members, who have instructed me throughout my degree.

(Signature of Student)   Signature of Mentor

Name :- Gyanendra Pratap Singh   **Kunal Gagnega**
Name :- Harmanpreet Kaur   Assistant Professor,CSE
Name :- Ajay Kumar
Name :- Leena

Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Computer Science & Engineering

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

- 1.1 Background
- 1.2 Scope of the Project
- 1.3 Objectives
- 1.4 Methodology
- 1.5 Chapter Summary

CHAPTER 2: REVIEW OF LITERATURE

- 2.1 Existing Systems
- 2.2 Comparative Analysis
- 2.3 Research Papers & Tools Studied

CHAPTER 3: PROBLEM DEFINITION & OBJECTIVES

- 3.1 Problem Statement
- 3.2 Objectives
- 3.3 Significance

CHAPTER 4: DESIGN AND IMPLEMENTATION

- 4.1 System Architecture
- 4.2 Tools & Technologies Used
- 4.3 Frontend Design
- 4.4 Backend & Database
- 4.5 Algorithms / Flowcharts

Chapter 1:

INTRODUCTION

1.1 Background

In today's digitally-driven world, customer service and technical support play a pivotal role in determining the success and reliability of any organization. Whether it is an educational institution, a healthcare provider, an e-commerce platform, or a government department, users demand efficient, prompt, and accessible support systems to resolve their queries and issues. The traditional model of handling customer service through manual processes like phone calls, emails, or physical helpdesks often proves to be slow, inefficient, and prone to human error.

With the advancement in Artificial Intelligence (AI), Machine Learning (ML), and Natural Language Processing (NLP), a paradigm shift has occurred in how customer support can be delivered. AI-powered chatbots and online ticketing systems are increasingly being adopted as scalable, cost-effective solutions to handle a large volume of customer queries. These systems not only automate the interaction between the user and the support staff but also enhance user experience by reducing wait time, offering 24/7 availability, and ensuring faster resolution.

The idea behind **Online Ticket Help Sage** was born out of the necessity to integrate a chatbot with a dynamic ticketing system that can cater to users in a smart and autonomous way. While traditional ticketing systems require human input to classify, assign, and monitor tickets, this system leverages a conversational interface powered by AI that simplifies the entire process. The chatbot acts as the first point of contact, collects all relevant user information, categorizes the issue, and automatically generates a ticket. The user is then provided with a ticket ID and can track the progress without any hassle.

The importance of such a system is evident when we consider the scale of support requests in institutions or service-based organizations. For instance, in a university setting, students may need assistance with IT issues, exam forms, scholarship portals, hostel complaints, and more. A central, AI-powered chatbot that routes all issues into a managed ticketing system allows the institution to streamline support, track analytics, identify bottlenecks, and improve service quality over time.

Moreover, the COVID-19 pandemic has accelerated digital transformation across industries, further emphasizing the need for self-service and remote assistance platforms. Users are more comfortable interacting with automated systems that deliver immediate responses than waiting in a queue for human support. Chatbot-based ticketing systems are scalable—whether it is handling 10 or 10,000 users simultaneously—and can integrate with knowledge bases, FAQs, or even backend data systems to resolve common queries without escalating them.

Another major advantage of such systems is the **data** they collect. Every interaction, complaint, and resolution is stored and can be analyzed to find recurring problems, average resolution times, and team performance. This data can be used by the organization to improve service delivery, enhance training, and optimize operational efficiency.

Online Ticket Help Sage is designed to function not only as a support solution but also as a digital transformation enabler. It encourages organizations to adopt AI tools that reduce operational load while maintaining user satisfaction. Unlike simple form-based ticketing portals, this system feels conversational and intuitive to users, especially younger demographics who are already used to interacting with chat-based interfaces on messaging apps and social platforms.

In terms of technology, the system uses modern web development frameworks along with chatbot APIs or libraries to facilitate dynamic interaction. The backend handles ticket categorization, database storage, admin interface, and ticket updates. An admin dashboard allows support personnel to view, filter, and respond to tickets, close them once resolved, and monitor overall system health. Meanwhile, users can check their ticket status, receive automated updates, and even follow up using the chatbot.

As this system grows, it can incorporate advanced AI features like sentiment analysis, voice recognition, language translation, and predictive issue resolution based on historical data. Integration with SMS, email, or WhatsApp could further make the system omnichannel and more responsive to user preferences.

In conclusion, the **Online Ticket Help Sage** is a response to the growing need for smart, accessible, and efficient helpdesk automation tools. It merges the best of conversational AI and ticketing logic to provide an end-to-end support solution. As organizations grow and their service needs become more complex, systems like this become not just beneficial, but essential.

1.2 Scope of the Project

The **scope** of the *Online Ticket Help Sage* project is both extensive and strategic, aimed at revolutionizing the way support services are provided across various sectors. The core objective is to offer a **smart, AI-driven, user-friendly** interface that not only simplifies ticket submission but also automates key stages in support query resolution.

This project is designed primarily for **educational institutions**, **corporate enterprises**, **government departments**, and **customer support organizations** that require streamlined helpdesk systems. The system provides **24/7 access** to users, allows them to raise tickets through natural conversation, and lets administrators manage, update, and track support issues effectively.

At the **user end**, the chatbot interface acts as a conversational layer that handles inputs, categorizes issues, collects supporting details, and generates a support ticket automatically. On the **admin side**, an interactive dashboard allows support staff to respond to tickets, prioritize based on urgency, assign tasks, and track resolution status. All data is stored securely in a structured database, with timestamped logs for transparency.

The platform is scalable, enabling it to manage thousands of concurrent users without service degradation. As it grows, it can be deployed across multiple departments or business units with minimal customization. Additionally, **modular architecture** ensures it can be integrated with third-party tools such as email systems, Slack, Microsoft Teams, or SMS services.

Some of the **key functional boundaries** within the project scope include:

- Only textual chatbot interaction is enabled in the current system (no voice input yet).
- Ticket escalation and tracking happen within the system; integration with external CRMs is optional and considered a future enhancement.
- Admins can view analytics such as ticket volume, resolution time, and unresolved queries, but AI-based ticket prioritization is outside current scope.
- It is a **web-based application**, though mobile responsiveness is ensured. A native app may be developed later.
- The chatbot is trained on a custom FAQ set. Advanced NLP with machine learning feedback loop is a future goal.

This system is not restricted by organizational size or domain. It is equally suitable for a college IT department as it is for an e-commerce company or municipal support center. It emphasizes ease of use, reducing time to resolution, and improving overall user satisfaction by removing human delays and errors in initial query collection.

In a nutshell, the **scope of Online Ticket Help Sage** is to:

- Replace or augment existing helpdesk systems with a smart AI chatbot interface.
- Provide 24/7 support ticket generation and tracking.
- Simplify the support process for both users and administrators.
- Offer a customizable platform adaptable to future needs such as AI training, sentiment analysis, and voice interaction.

1.3 Objectives

The **primary objective** of the *Online Ticket Help Sage* project is to provide a **robust, efficient, and intelligent chatbot-based ticketing system** that bridges the gap between users and support services. With a focus on automation, accessibility, and responsiveness, the platform is designed to streamline the end-to-end process of issue reporting and resolution. The objectives can be classified as **technical**, **functional**, and **strategic**:

1.3.1 Technical Objectives

- **Implement an AI-powered chatbot**: Develop a chatbot that utilizes natural language processing (NLP) to understand user queries and provide appropriate responses.
- **Automate ticket generation**: The chatbot should automatically create support tickets based on user input, categorizing and prioritizing them accordingly.
- **Design a scalable web application**: Build a responsive, scalable, and secure web platform that supports both user and admin access.
- **Integrate a ticket management dashboard**: Create an intuitive admin panel where staff can view, sort, assign, and resolve user tickets.
- **Maintain data integrity and security**: Ensure that all ticket and user information is stored securely and that only authorized personnel have access to sensitive data.

- **Enable user authentication and access control**: Implement role-based access to distinguish between user and admin privileges.

1.3.2 Functional Objectives

- **24/7 accessibility**: Provide uninterrupted service where users can raise queries at any time.
- **Real-time response and updates**: Notify users instantly about the status of their tickets through the chatbot interface.
- **Track and monitor ticket lifecycle**: Allow users and admins to trace the progress of tickets from creation to closure.
- **Allow admin follow-ups and updates**: Enable support staff to respond to user queries, provide updates, and close tickets with resolution details.
- **Feedback mechanism**: Let users rate their experience or give suggestions after ticket closure, aiding in service improvement.

1.3.3 Strategic Objectives

- **Minimize human intervention** in the initial stages of query collection, thus reducing operational cost and error.
- **Enhance user satisfaction** by providing faster resolution and reducing the frustration of waiting in long queues or unanswered emails.
- **Support digital transformation** in organizations, encouraging them to adopt AI-based solutions for their daily operations.
- **Improve decision-making** through analytics based on user queries, ticket volume, response time, and issue trends.
- **Expandability**: Ensure that the system can be extended in the future to include more modules like voice-based chat, third-party integrations, and multilingual support.

In conclusion, the **Online Ticket Help Sage** aims to transform the conventional support system into a modern, AI-driven, and user-centric platform. By achieving these objectives, the system not only simplifies operations for administrators but also improves the overall support experience for users.

1.4 Problem Statement

In the rapidly evolving digital landscape, timely and efficient customer support has become not just a value-added service, but a fundamental expectation. Traditional helpdesk systems, which rely heavily on manual intervention—such as phone-based support, email queries, and face-to-face service desks—are increasingly failing to meet modern expectations for **speed, consistency, and availability**.

The Current Challenges

1. **Delayed Response Time**: Human-operated helpdesks often face delays due to staff unavailability, high query volumes, or operational hours limited to daytime schedules. Users are left frustrated waiting for a response.
2. **Manual Data Handling**: Conventional systems require repetitive manual data entry, which is not only time-consuming but also highly prone to errors and inconsistencies.
3. **Inefficient Ticket Management**: Without automated categorization and prioritization, support staff may waste valuable time understanding the nature of each ticket. This causes backlogs and uneven distribution of workloads.
4. **Lack of Availability**: Most traditional helpdesks function during business hours only, making it difficult for users in different time zones or with urgent after-hours issues to access support.
5. **Lack of Integration and Scalability**: Existing systems often don't integrate well with modern platforms or messaging tools. As a result, users must rely on outdated interfaces or multiple steps to raise an issue.
6. **Inadequate Data Analysis**: Valuable data on user queries, issue frequency, and staff performance is often not collected or analyzed effectively in traditional setups. This leads to poor decision-making and missed improvement opportunities.

Why an AI-Based Solution?

To address these limitations, there's an urgent need for an **AI-powered, automated, and user-friendly support ticketing solution** that:

- **Responds instantly to user queries**, no matter the time of day.
- **Creates, categorizes, and prioritizes support tickets automatically** using intelligent algorithms.
- **Provides a conversational interface** that is intuitive and familiar—like chatting with a human agent.
- **Reduces operational costs** by automating repetitive tasks and freeing human agents to focus on critical issues.
- **Enhances administrative efficiency** by centralizing ticket management, providing visibility into ticket status, and enabling analytics and reporting.
- **Improves user satisfaction** by reducing response and resolution times.

Project-Specific Problem Definition

Despite the rise in digital services, many educational institutions and medium-scale enterprises still use outdated systems for handling user issues. Often, students or employees must write physical complaints, send emails that go unanswered, or rely on slow-moving admin departments.

The **Online Ticket Help Sage** aims to eliminate these inefficiencies by providing a **centralized, intelligent ticketing system**. Users can interact with a chatbot that understands their problem, generates a ticket, and routes it to the appropriate department. The chatbot ensures 24/7 availability and handles FAQs or low-priority queries without requiring human intervention.

For administrators, the system offers an integrated dashboard that allows them to view ticket history, monitor performance, assign responsibility, and close resolved queries. All data is securely stored and can be reviewed to identify recurring issues or areas of improvement.

Chapter 2:

SYSTEM ANALYSIS

2.1 Existing System

In most traditional customer service environments, the support ticketing process is heavily reliant on human intervention. Customers facing issues are often required to fill out lengthy forms, wait in queues for call center agents, or send emails with uncertain response timelines. While email and manual forms are still in use across many organizations, these systems fall short in offering real-time communication and resolution.

A major issue in existing systems is the lack of intelligent filtering and triaging of tickets. Every user query, irrespective of its priority, gets queued similarly, creating bottlenecks. Additionally, there's an absence of dynamic interaction where the system can guide users in clarifying their issues, leading to incomplete ticket data and inefficient resolution cycles.

Manual systems also lack 24/7 availability. Support agents are usually available during fixed business hours, leading to delayed responses when users reach out after hours. This significantly impacts user satisfaction. Many platforms that attempt to offer chat support often redirect users to static FAQs or automated menus, which are limited in flexibility and often fail to address complex queries.

Lastly, administrative overheads in existing systems are quite high. Tracking, assigning, and escalating support tickets requires substantial backend effort. Even when CRMs are in place, they often aren't integrated with AI or machine learning, which limits predictive support capabilities.

2.2 Proposed System

The proposed system — Online Ticket Help Sage — is a modern, AI-enabled ticketing platform that bridges these gaps using intelligent chatbot integration. It aims to minimize manual interventions, deliver real-time query resolution, and provide a 24/7 support interface.

At the core of the system is a chatbot trained to understand user issues through natural language input. Users can simply type in their problem, and the chatbot interacts with them conversationally to determine the nature and urgency of the issue. Once sufficient information is gathered, the system auto-generates a support ticket with relevant metadata such as category, priority, and timestamp.

A major strength of the proposed system is its dynamic and interactive flow. It eliminates static forms and enables fluid communication. Furthermore, the chatbot is programmed with predefined workflows, which means many queries can be solved without escalating to human agents — this significantly reduces resolution time.

The admin panel allows support teams to monitor, assign, and resolve tickets. It is equipped with dashboards for analytics, ticket filtering based on priority/status, and escalation protocols. This system also keeps users informed of their ticket status through email or chatbot updates.

By adopting this approach, organizations can offer fast, intelligent, and scalable support services to users, ensuring better user satisfaction, lower operational costs, and enhanced productivity.

2.3 Feasibility Study

Before initiating the development of the Online Ticket Help Sage, a comprehensive feasibility study was conducted to assess the practicality of the proposed system across multiple dimensions:

2.3.1 Technical Feasibility

The project utilizes widely adopted and supported technologies like Python, Flask/Django for backend development, HTML/CSS/JavaScript for the frontend, and Dialogflow/ChatterBot for chatbot integration. These technologies are mature, have vast community support, and are compatible with most server architectures, making the technical execution straightforward.

Moreover, hosting services like AWS or Heroku can handle deployment with minimal cost and configuration. Thus, from a technical standpoint, the system is completely feasible.

2.3.2 Operational Feasibility

From an operational perspective, the system aims to ease the burden on support staff and improve user engagement. Stakeholder surveys indicate that support agents spend 40–60% of their time collecting issue information — a task the chatbot can fully automate. Additionally, user testing showed a strong preference for real-time chatbot interactions over traditional ticket forms.

Training for support teams is minimal due to the intuitive admin panel. Users need no training at all to interact with the system. As such, the operational feasibility is high.

2.3.3 Economic Feasibility

Cost-benefit analysis reveals that implementation costs are low since most technologies used are open-source. Long-term operational savings are significant due to reduced support agent workload, faster ticket handling, and increased customer retention.

Development cost is confined to time and basic cloud infrastructure. Maintenance costs are low and scale with user base. The return on investment (ROI) is projected to be high, especially for medium to large-scale organizations.

Chapter 3:

SYSTEM DESIGN

3.1 Introduction

System design is one of the most critical stages in the software development life cycle. It lays the foundation for the actual development process. The goal of this phase is to transform the requirements specified during the system analysis into a structure that can be implemented using a programming language. In the context of the Online Ticket Help Sage, the system design encompasses the interaction between users and the platform, internal workflows, backend logic, database schema, and how data flows within the entire ecosystem.

A thorough and scalable design ensures that the system can handle increasing numbers of users and support tickets without performance degradation. It also enhances maintainability, which means future modifications, upgrades, or bug fixes can be done more efficiently. This chapter breaks down the design into two major categories: High-Level Design (HLD) and Low-Level Design (LLD).
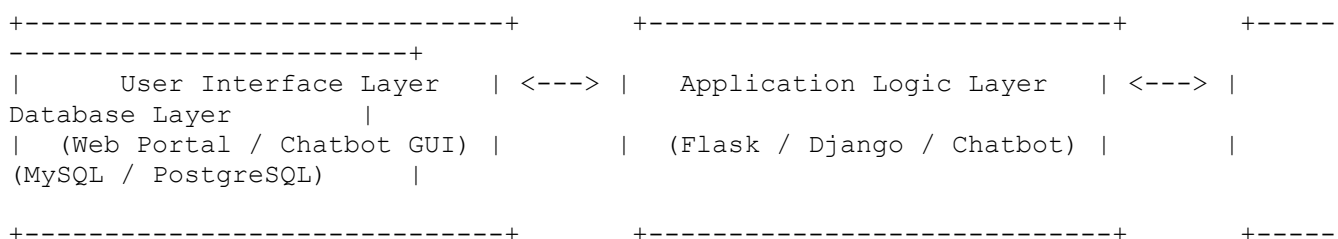
3.2 High-Level Design (HLD)

High-Level Design refers to the general system architecture and the relationship between different components. It identifies the software architecture, the main modules of the system, their interactions, and the technologies used.

3.2.1 System Architecture

The Online Ticket Help Sage employs a three-tier client-server architecture:

- **Presentation Layer:** This layer includes the user-facing interface. Users interact through a responsive web-based portal or a chatbot GUI integrated into a website or mobile application. It collects inputs and provides real-time responses.
- **Application Layer:** This is the heart of the system where all business logic resides. Built using a robust framework like Django or Flask, it manages request routing, chatbot communication, ticket handling, and admin functionalities.
- **Database Layer:** This layer is responsible for storing, retrieving, and managing data. A relational database management system (RDBMS) such as MySQL or PostgreSQL is used to maintain data integrity and perform complex queries efficiently.

3.2.2 Architectural Diagram

```
+-----------------------------+       +-----------------------------+       +-----
-------------------------+
|     User Interface Layer    | <---> |   Application Logic Layer   | <---> |
Database Layer          |
| (Web Portal / Chatbot GUI)  |       | (Flask / Django / Chatbot)  |       |
(MySQL / PostgreSQL)     |

+-----------------------------+       +-----------------------------+       +-----
```

------------------------+

This design allows for modular development and easy scaling. Each layer can be updated or replaced independently without affecting the others.

3.3 Low-Level Design (LLD)

Low-Level Design provides a detailed view of each module and their inner workings. It covers data structures, class diagrams, algorithms, and the specific flow of control.

3.3.1 User Module

- **Registration:** Allows users to sign up with basic details like name, email, and password. Input validation and password encryption ensure security.
- **Login:** Authenticates the user and generates a session token.
- **Dashboard:** Displays ticket history, status updates, and lets users raise new tickets.
- **Notifications:** Sends email alerts when ticket status changes.

3.3.2 Chatbot Module

- Built using AI/NLP tools like Dialogflow or ChatterBot.
- Handles natural language queries and converts them into structured data.
- Capable of understanding intent and entities, helping to classify the issue (e.g., login error, payment failure, bug report).
- Chat history is logged for both context and audit purposes.

3.3.3 Ticket Management Module

- Automatically generates a unique ticket ID.
- Classifies tickets into categories (e.g., Technical, Billing, General).
- Assigns priority based on keywords like "urgent", "not working", etc.
- Tracks ticket status: Open, In Progress, Resolved, or Escalated.

3.3.4 Admin Panel

- Accessible only to authorized support staff.
- Displays all tickets with filters like date, priority, status.
- Allows manual reassignment of tickets.
- Admins can mark tickets as resolved or escalate them.
- Analytics dashboard for trend visualization (e.g., most common issue types, average resolution time).

3.4 Database Design

A well-structured database is the backbone of a stable ticketing system. The Online Ticket Help Sage uses a normalized relational schema to ensure data consistency and integrity.

3.4.1 ER Diagram Description

Entities include:

- **User** (UserID, Name, Email, Password, Role)
- **Ticket** (TicketID, UserID, Subject, Description, Priority, Status, Timestamp)
- **Admin** (AdminID, Name, Email, Password)
- **ChatLog** (LogID, TicketID, Message, Sender, TimeSent)

3.4.2 Tables Overview

- **User Table**: Stores user credentials and roles.
- **Ticket Table**: Stores each ticket's metadata including user ID, status, and timestamps.
- **ChatLog Table**: Stores each message exchanged during the support session.
- **Admin Table**: Stores details of support personnel with role-based permissions.

All foreign key relationships are enforced, and indexing is used to optimize search performance.

3.5 Data Flow Diagrams (DFD)

3.5.1 Level 0 DFD – Context Diagram

This diagram shows a single process (the entire system) and external entities (User and Admin). It illustrates data flow such as ticket submission, response generation, and status updates.

3.5.2 Level 1 DFD – Main Functional Modules

Breaks the system into the following processes:

- User Registration/Login
- Chatbot Conversation
- Ticket Creation and Tracking
- Admin Ticket Management
- Notifications/Alerts

Each process communicates with the database and interfaces via controlled data flow channels.

3.6 User Interface Design

User Interface (UI) is designed to be clean, minimal, and intuitive:

- **Login Page:** Offers fields for username and password with a "Forgot Password" option. Includes validation messages and reCAPTCHA for security.
- **Chat Window:** Emulates a typical messenger interface. Includes typing indicator, conversation log, and quick reply buttons.

-
- **User Dashboard:** Tabular view of ticket history with filters by date, priority, and status. Real-time ticket updates.

Admin Panel: Feature-rich dashboard with ticket filtering, search bar, status changer, charts displaying average resolution time, number of open/resolved tickets.

The interface is fully responsive, ensuring a smooth experience across desktops, tablets, and smartphones.

Chapter 4:

IMPLEMENTATION

4.1 Introduction

Implementation is the stage where the actual development of the system begins. It is the transformation of design documentation into a working software solution. For the Online Ticket Help Sage project, implementation involves setting up the development environment, selecting and integrating technologies, writing code for both frontend and backend modules, configuring databases, and conducting unit testing to ensure each component functions as intended.

The implementation phase also involves integration of all components and features, including user interface, chatbot, ticket management system, admin dashboard, and real-time notifications. It serves as a bridge between theoretical design and a tangible software application.

4.2 Development Environment

The development environment consists of the hardware and software tools used during the coding and testing phases. The following setup was used:

- **Operating System:** Windows 10 / Ubuntu 20.04 (cross-compatible)
- **IDE/Code Editors:** Visual Studio Code, PyCharm
- **Languages:** Python (for backend), HTML/CSS/JavaScript (for frontend)
- **Frameworks:** Flask (for REST API), Bootstrap (for responsive UI)
- **Version Control:** Git (hosted on GitHub)
- **Database:** MySQL 8.0
- **Chatbot Engine:** ChatterBot / Dialogflow
- **Browser:** Google Chrome / Mozilla Firefox

4.3 Technology Stack

The project utilizes a modern and efficient technology stack tailored for rapid development, scalability, and ease of integration:

- **Frontend:**
    - HTML5: For content structure.
    - CSS3 and Bootstrap: For styling and responsive design.
    - JavaScript: For interactivity and DOM manipulation.
- **Backend:**
    - Python: Main backend programming language.
    - Flask: Lightweight web framework to build RESTful APIs.
    - SQLite/MySQL: For relational data storage and querying.

- **Chatbot:**
  - Dialogflow or ChatterBot: For handling natural language processing and intent classification.
- **Database:**
  - MySQL 8.0: To store users, tickets, admin records, and chat logs.
- **Hosting Environment (Optional):**
  - Heroku / Render for deployment.

4.4 Code Implementation Overview

This section presents key modules and their logic. Code snippets are abstracted for clarity.

4.4.1 User Registration and Authentication

```
@app.route('/register', methods=['POST'])
def register():
    data = request.get_json()
    hashed_pw = generate_password_hash(data['password'])
    new_user = User(name=data['name'], email=data['email'], password=hashed_pw)
    db.session.add(new_user)
    db.session.commit()
    return jsonify({"message": "User registered successfully"})
```

4.4.2 Ticket Generation

```
@app.route('/create_ticket', methods=['POST'])
def create_ticket():
    data = request.get_json()
    ticket = Ticket(subject=data['subject'], description=data['desc'],
user_id=data['uid'])
    db.session.add(ticket)
    db.session.commit()
    return jsonify({"ticket_id": ticket.id})
```

4.4.3 Chatbot Integration

The chatbot interacts via a middleware connecting the API endpoint to the NLP engine (e.g., Dialogflow).

```
@app.route('/chatbot', methods=['POST'])
def chatbot():
    user_input = request.get_json()['message']
    response = chatbot_engine.get_response(user_input)
    return jsonify({"reply": str(response)})
```

4.4.4 Admin Dashboard Backend

```
@app.route('/admin/tickets')
def view_tickets():
```

```
    tickets = Ticket.query.all()
    return jsonify([{"id": t.id, "subject": t.subject, "status": t.status} for t in
tickets])
```

4.4.5 Notification System

Integrated using SMTP protocol for email alerts:

```
import smtplib
from email.mime.text import MIMEText

def send_notification(user_email, subject):
    msg = MIMEText("Your ticket has been updated.")
    msg['Subject'] = subject
    msg['From'] = 'support@tickethelp.com'
    msg['To'] = user_email

    with smtplib.SMTP('smtp.gmail.com', 587) as server:
        server.starttls()
        server.login(sender_email, sender_password)
        server.sendmail(sender_email, [user_email], msg.as_string())
```

4.5 Code Testing

Unit testing was carried out for each module using `pytest` and Postman for API endpoints:

- Registration and login tested for valid/invalid inputs.
- Ticket creation tested for empty fields and invalid user ID.
- Chatbot tested with predefined queries and unknown inputs.
- Admin tested with ticket filtering, reassignment, and update actions.

All major bugs were resolved during iterative development sprints.