AI-Powered Customer Support and Ticketing Platform: Hackathon Report

Kshitij Sonje, Anya Gupta, Priyanshi Peswani

May 30, 2025

1 Introduction

In today's hyperconnected world, customer expectations for service quality, responsiveness, and personalization are at an all-time high. Customers demand instant answers, consistent experiences across channels, and efficient resolution of their problems. However, traditional customer support systems often fall short—plagued by long wait times, repetitive queries, and limited adaptability to diverse customer needs.

Support agents face their own challenges. From managing large volumes of tickets to manually searching for information across disjointed systems, agents are under constant pressure to deliver timely and effective solutions. Moreover, companies lack real-time visibility into support operations and insights that can drive proactive improvements.

1.1 Motivation

The need for intelligent, automated, and scalable customer support solutions is more pressing than ever. The widespread adoption of Artificial Intelligence (AI) presents a powerful opportunity to revolutionize support operations—automating routine tasks, empowering agents with contextual assistance, and providing strategic insights into customer behavior and satisfaction.

1.2 Problem Statement

Current customer support platforms often lack flexibility, fail to leverage organizational knowledge effectively, and do not provide intelligent assistance or predictive capabilities. There is a growing demand for a programmable, AI-first platform that can be tailored by companies to match their services, adapt to their workflows, and continuously learn from interactions.

1.3 Objective of the Hackathon

This hackathon aims to address these challenges by designing and building a prototype of an AI-powered customer support and ticketing platform. The objective is to:

- Create a multi-tenant, programmable platform that allows businesses to onboard and configure their support environment.
- Implement AI-driven chatbots for customers and agents, capable of understanding natural language, maintaining context, and suggesting resolutions.
- Automate ticket management using AI for intelligent categorization, prioritization, and solution recommendations.
- Leverage AI for agent performance monitoring, customer sentiment analysis, and operational insights.
- Provide a foundation for scalable, efficient, and proactive customer experience (CX) management.

Through this report, we document the system design, features implemented, AI integrations, and key learnings from the development of this next-generation support platform.

2 System Overview

The AI-powered customer support and ticketing platform developed during the hackathon is a full-stack, multi-tenant solution that integrates traditional support tools with modern AI capabilities. The platform is deployed and accessible at: `https://hackxsupport.streamlit.app/`.

2.1 Key Modules

The system is divided into the following core modules:

- **Customer Portal:** Provides users with an interface to raise new support tickets, view past queries, and interact with an AI-powered chatbot that offers real-time assistance.
- **Agent Dashboard:** Equips support agents with tools to manage tickets, view customer history, receive AI-suggested responses, and interact with a contextual AI assistant to resolve queries faster.
- **Admin Panel:** Enables tenant administrators to configure support settings, monitor system usage, assign roles, and view analytics regarding ticket inflow and resolution efficiency.
- **AI Integration Layer:** Facilitates natural language understanding, ticket categorization, priority prediction, and automated reply suggestions. This layer also supports chatbot conversations for both customers and agents.
- **Database and Ticketing Backend:** Built on SQLite for simplicity during the hackathon, this component handles persistent storage of user data, tickets, chat history, and system configuration.

2.2 User Flow

1. A customer visits the platform, initiates a support request, or chats with the AI chatbot to describe their issue.
2. The system logs the ticket, assigns a priority (using AI-based categorization), and routes it to an available agent.
3. Agents use the dashboard to view tickets, assisted by AI that recommends replies and related documentation.
4. The admin oversees platform activity and can extract insights from dashboards for decision-making and optimization.

2.3 Deployment Stack

- **Frontend:** Streamlit-based web interface for rapid prototyping and user interaction.
- **Backend:** Python (Flask for API services), SQLite for data storage.
- **AI Models:** Integrated via APIs or in-code logic for chatbot conversation, classification, and ranking tasks.
- **Hosting:** Streamlit Community Cloud for live deployment and demonstration.

This modular and extensible architecture enables future integration with external APIs, more advanced AI models, and scalable database solutions.

3 Key Features

The platform is designed with usability, scalability, and intelligence at its core. Below are the key features implemented during the hackathon:

3.1 1. Multi-Tenant Onboarding and User Roles

The platform supports multiple businesses (tenants), allowing each to manage their support infrastructure independently. Role-based access control enables three primary user types:

- **Customer:** Can raise tickets and chat with the customer-facing chatbot.
- **Agent:** Manages assigned tickets and uses the AI assistant for faster resolutions.
- **Admin:** Configures the support environment and monitors operational insights.

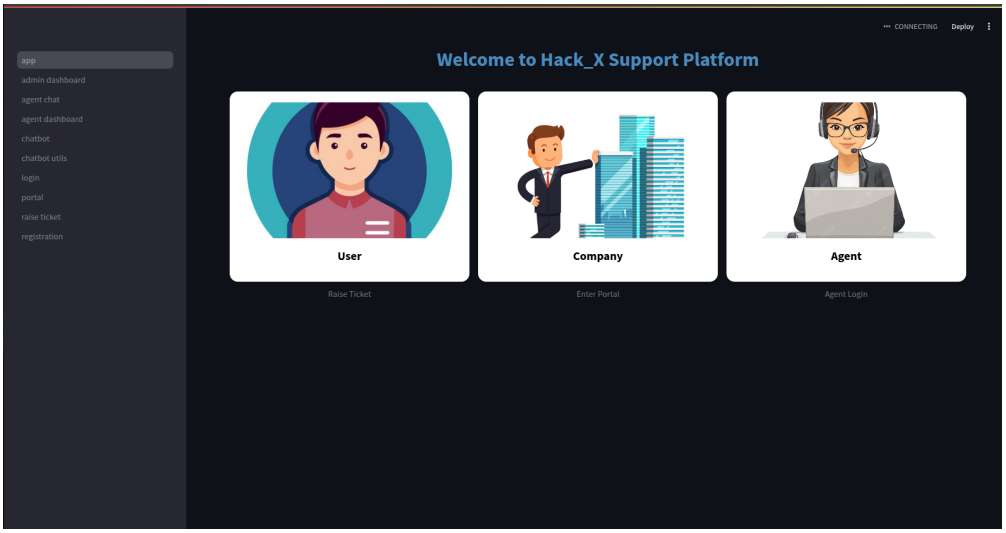

Figure 1: Login Page Interface

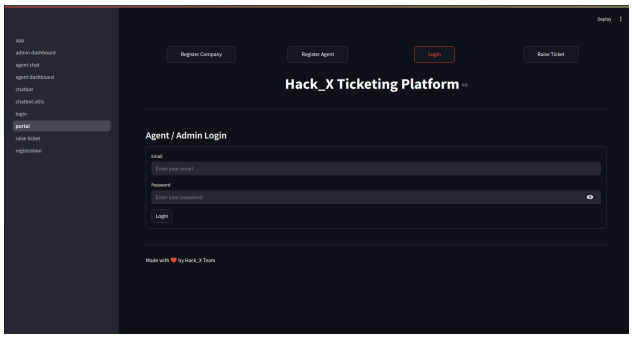

Figure 2: Admin/agent login

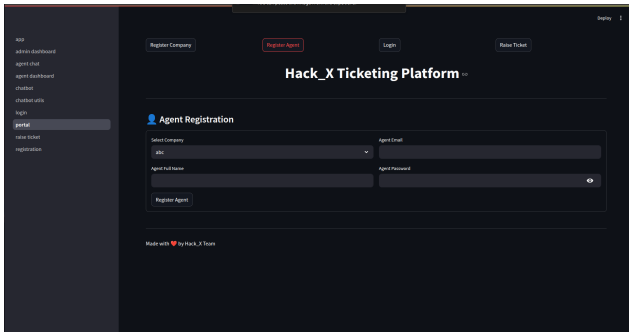

Figure 3: Agent Registration

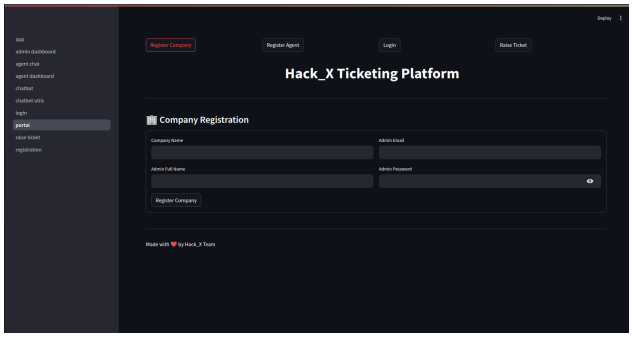

Figure 4: Company Registration

3.2 2. AI-Powered Customer Chatbot

Customers interact with a conversational AI bot that understands their queries, provides answers to FAQs, and helps route them to ticket creation if needed. The bot maintains context and streamlines the issue-reporting process.

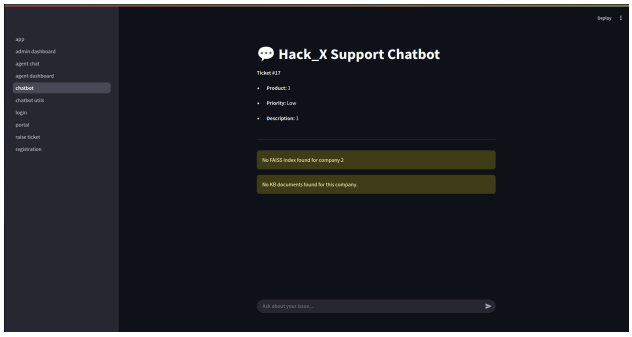

Figure 5: User Chatbot

3.3 3. Smart Ticket Creation and Management

Users can submit support requests via a structured form. The system automatically tags, categorizes, and prioritizes incoming tickets using AI.

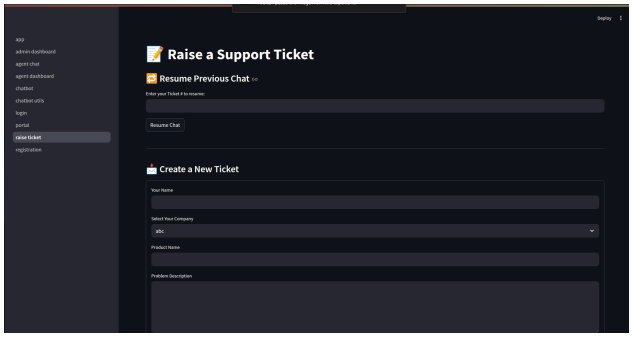

Figure 6: Resume Conversation

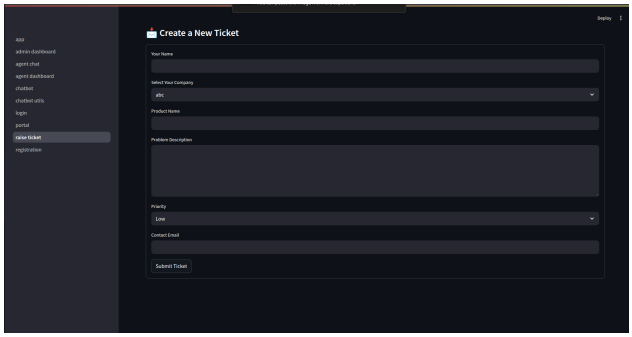

Figure 7: Raise Ticket

Agents can view ticket details, add responses, and update the status directly from their dashboard.

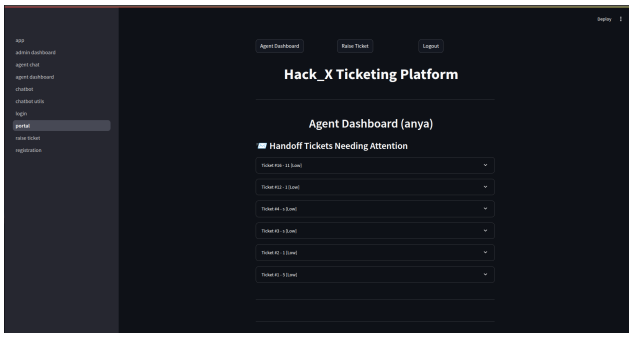

Figure 8: Agent Dashboard

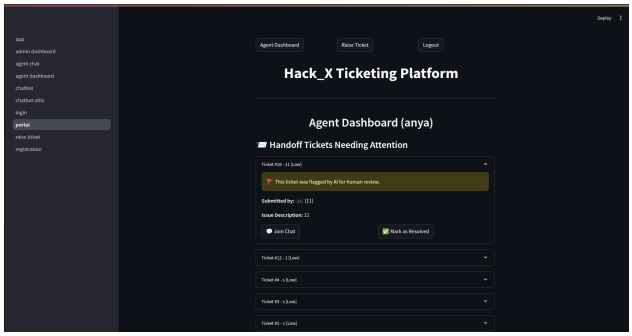

Figure 9: Agent Portal

3.4 4. Agent Assistant (AI Copilot)

Support agents are equipped with a built-in AI assistant that helps:

- Summarize customer issues.
- Recommend possible resolutions or documentation.

- Draft context-aware responses.

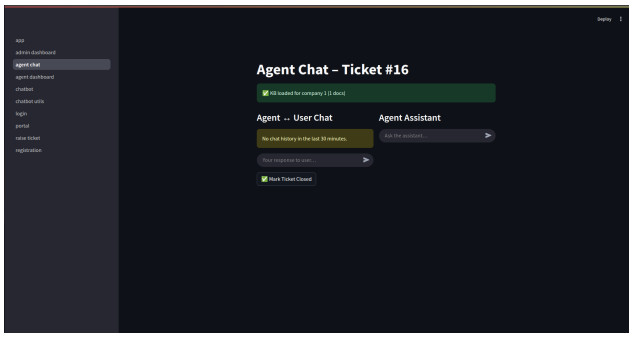

Figure 10: Agent Chatbot

3.5 5. Admin Dashboard and Analytics

Admins can view global ticket metrics, agent performance, customer satisfaction trends, and system usage. These insights help in team planning, SLA monitoring, and CX optimization.

Figure 11: Admin Dashboard Interface

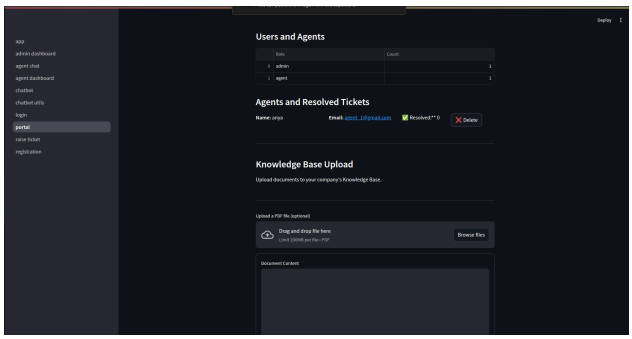

Figure 12: Admin Portal

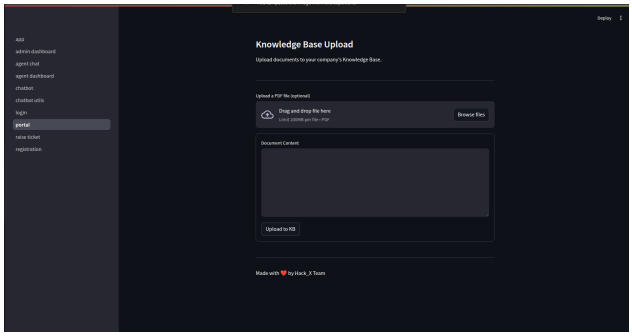

Figure 13: Knowledge Based

3.6 6. Streamlit-Powered Interactive UI

All features are accessible via an intuitive web interface built using Streamlit. Its dynamic rendering enables real-time updates and a smooth user experience across roles.

3.7 7. Lightweight and Portable Backend

The backend leverages Python and SQLite, ensuring ease of deployment and portability. The platform can be easily containerized or migrated to cloud-native services for scale.

4 AI Integration Strategy

The AI-powered customer support system integrates a large language model (LLM), knowledge base (KB) retrieval, and ticket metadata to generate intelligent and context-aware responses for both customers and support agents.

4.1 LLM Selection and Prompting

We use Google's Gemini 2.0 Flash model as the core assistant. The model is accessed via the Gemini API endpoint:

`https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-flash:generateCo`

The prompt provided to the model is dynamically constructed using:

- **Ticket Details**: Product, priority, description, company ID, and contact name.
- **Knowledge Base Context**: Top matching chunks retrieved using FAISS.
- **Conversation History**: The last 10 dialogue turns.
- **User Input**: The latest customer query.

The final prompt enforces a structure that encourages factual, empathetic, and concise replies. It also embeds instructions for special cases like handoff detection using the tag `[HANDOFF_REQUIRED]`.

4.2 Knowledge Base Search using FAISS

To enhance response accuracy, relevant content is retrieved from a company-specific knowledge base using FAISS (Facebook AI Similarity Search). The retrieval process includes:

1. Loading or building a FAISS index from KB articles stored in the SQLite database.
2. Encoding the user's query using a Sentence-BERT model (`all-MiniLM-L6-v2`).
3. Fetching the top-k (default: 3) similar chunks as context.

Each company has its own FAISS index folder (`faiss_index_company_{company_id}`), ensuring data isolation and retrieval efficiency.

4.3 Chatbot Workflow

The chatbot follows the workflow below:

1. Validate active session and ticket ID.
2. Fetch ticket metadata and chat history.
3. Check for ticket closure or escalation status.
4. Retrieve relevant KB content using the FAISS index.
5. Generate a structured prompt with all available context.
6. Query the Gemini model and display the response.
7. Log all interactions in the `chat_sessions` table.
8. Detect ticket resolution or handoff triggers based on keywords or LLM response flags.

4.4 Session Inactivity and Auto-Closure

A session is monitored for inactivity using timestamps in the chat history. If no message is received within 30 minutes, the ticket is automatically marked as closed. This is implemented using:

```
timestamp >= datetime.now() - timedelta(minutes=30)
```

4.5 Escalation and Human Handoff

If the chatbot is unable to resolve an issue or the user requests human help, the response includes the marker `[HANDOFF_REQUIRED]`. The system updates the ticket status to `Handoff`, and the agent-side dashboard auto-refreshes to receive new messages.

4.6 Agent-Side Integration

An agent dashboard is provided in `agent_chat.py`, which connects to the same database and displays real-time incoming user queries for escalated tickets. Embedding-based retrieval (via ChromaDB) is also available to assist human agents in resolving queries faster.

5 Technical Architecture

5.1 Tech Stack

Layer	Technologies & Tools	Description
Frontend	Streamlit	Rapid, interactive web app interface built with Python, enabling quick UI development.
Backend	Python, SQLite, Passlib	Handles business logic, user authentication, password hashing, and database interaction.
Database	SQLite	Lightweight relational database for persisting companies, users, tickets, and other data.
Authentication	Passlib	Secure password hashing and verification using bcrypt.

Table 1: Technology stack used in the Hack_X Ticketing Platform

5.2 Backend & Frontend Overview

The platform backend is implemented using Python with Streamlit serving both frontend and backend logic in a single application. Streamlit provides a simple way to build the user interface with interactive forms, buttons, and state management using `st.session_state`.

The backend handles user authentication, registration, and role-based access control. Passwords are securely hashed using the `Passlib` library's bcrypt scheme. The application uses SQLite as a persistent data store to manage company information, user credentials, roles (admin/agent), and support tickets.

Key backend modules include:

- **Registration Module:** Handles company and agent registration, ensures unique emails and company names, and hashes passwords.
- **Authentication Module:** Manages login, session state, and role assignment.
- **Dashboard Module:** Provides different views and functionalities based on user roles (admin or agent).
- **Ticketing Module:** Allows logged-in users to raise tickets, view ticket status, and communicate via the chatbot.

5.3 Database Schema

The database schema consists of the following primary tables:

- **Company:** Stores company details with unique company names.
- **User:** Stores user details including company association, name, email, hashed password, and role (admin or agent).
- **Ticket:** Stores support tickets raised by users with references to the user and company.
- **Chatbot Interaction (planned):** Stores conversation logs for AI-powered chat interactions.

The schema enforces referential integrity through foreign key constraints (e.g., users link to companies via `company_id`). Unique constraints prevent duplicate company names and user emails.

5.4 Deployment Infrastructure

The application is designed for easy deployment on cloud or on-premise servers with minimal dependencies. Key deployment characteristics include:

- **Single Python Application:** The entire platform runs as a Streamlit app, simplifying deployment and maintenance.
- **SQLite Database:** Enables lightweight, file-based storage requiring no separate database server.
- **Containerization (Optional):** The app can be containerized using Docker for consistent environments across development, testing, and production.

- **Hosting Options:** Can be deployed on any server or cloud provider supporting Python, such as AWS EC2, Google Cloud Compute Engine, Heroku, or Streamlit Community Cloud.

Security considerations during deployment include using HTTPS for encrypted communication, environment variable management for sensitive credentials, and regular backups of the SQLite database.

—

6 User Experience & Interface Design

6.1 UX Strategy

The primary goal of the Hack_X Ticketing Platform is to provide an intuitive and efficient interface tailored to different user roles: customers, agents, and administrators. The UX strategy focuses on simplicity, accessibility, and role-based navigation to streamline workflows and reduce training overhead. Navigation components such as top navigation buttons allow users to easily switch between key functions like registration, login, raising tickets, and accessing dashboards. Feedback mechanisms such as success messages and error alerts ensure users are informed of their actions' outcomes, improving trust and engagement.

6.2 Customer Portal

The customer portal serves as the primary interface for users raising support tickets. It allows new users to register companies and agents, and existing users to log in and submit tickets. The portal uses Streamlit forms to gather input data in a structured and user-friendly manner. For example, company registration requires the user to input the company name and admin details, with client-side validation to ensure all fields are completed. Passwords are securely hashed before storage. The portal dynamically adjusts the navigation bar based on authentication state, showing or hiding options like "Raise Ticket" or "Login" accordingly. The portal design emphasizes clarity and simplicity, enabling users to quickly perform key tasks without confusion.

6.3 Agent Dashboard

Agents access a dedicated dashboard post-login, providing a consolidated view of their assigned tickets and support tasks. The dashboard interface is optimized for productivity, with navigation streamlined to agent-specific functionalities. Upon successful login, the navigation bar updates to display options like "Agent Dashboard", "Raise Ticket", and "Logout", simplifying access to relevant actions. The dashboard itself supports key workflows such as ticket management, status updates, and communication with customers, fostering efficient support resolution.

6.4 Admin Controls

The admin dashboard is a privileged interface accessible only to users with administrative roles. It enables comprehensive management of companies, agents, and overall platform settings. Upon login, administrators see navigation options tailored to their role, including "Admin Dashboard", "Raise Ticket", and "Logout". The admin panel facilitates user and company registrations, role assignments, and monitoring system-wide activity. Security measures are enforced via role-based access control and secure password hashing, ensuring only authorized personnel can perform critical administrative functions. The design prioritizes clarity and control, empowering administrators to maintain platform integrity and support smooth operations.

7 Future Work & Enhancements

7.1 Planned Features

To continuously improve the Hack_X Ticketing Platform, several new features are planned. These include enhanced ticket prioritization using automated categorization, integration of multi-channel support such as email and chat, and real-time notifications for agents and customers. Additionally, a comprehensive reporting module will be developed to provide detailed analytics on ticket trends, agent performance, and customer satisfaction. These features aim to improve responsiveness and decision-making for all users.

7.2 AI Advancements

Leveraging AI technologies forms a key part of the platform's future roadmap. Planned advancements include deploying machine learning models to automate ticket routing and predict ticket resolution times, thereby optimizing agent workload distribution. The integration of an AI-powered chatbot is underway to provide instant support for common queries and guide users through ticket submission. Natural language processing (NLP) capabilities will also be enhanced to analyze ticket descriptions and detect sentiment, enabling proactive support and escalation.

7.3 Enterprise Readiness

To support deployment in large-scale enterprise environments, efforts will focus on scalability, security, and compliance. This involves implementing multi-tenant architecture to isolate company data securely, enhancing authentication with multi-factor methods, and ensuring compliance with data protection regulations such as GDPR. Additionally, integration with enterprise identity providers (e.g., LDAP, SSO) will be supported. Infrastructure improvements will enable high availability and disaster recovery capabilities, ensuring reliable service for enterprise customers.

8 Conclusion

8.1 Key Learnings

Throughout the development of the Hack_X Ticketing Platform, several valuable lessons were learned. These include the importance of secure authentication mechanisms, effective role-based access control, and seamless integration between frontend and backend components. Additionally, designing a user-friendly interface that caters to different user roles was critical for adoption. The project reinforced the significance of modular, scalable architecture for future growth and maintainability.

8.2 Impact of the Platform

The platform provides a comprehensive solution to streamline customer support processes, improving communication between customers, agents, and administrators. By automating key workflows and enabling efficient ticket management, the platform enhances operational efficiency and customer satisfaction. The integration of AI-powered features promises to further reduce resolution times and improve service quality, making the platform a valuable asset for organizations of all sizes.

8.3 Summary of Innovation

This project introduces innovative features such as AI-driven ticket routing and chatbot assistance, dynamic role-based dashboards, and a robust multi-company architecture. The use of modern technologies like Streamlit for rapid interface development combined with secure backend design demonstrates a forward-thinking approach to customer support systems. These innovations position the Hack_X Ticketing Platform as a scalable and intelligent solution in the evolving landscape of support services.

Appendices

A GitHub Repository Link

The complete source code for the Hack_X Ticketing Platform is available at:
`https://github.com/AnyaGupta12/AI_ticketing_bot`

B Setup Instructions

- Clone the repository from GitHub.
- Install the required Python packages using:

```
pip install -r requirements.txt
```

- Initialize the database by running the initialization script.
- Run the app with:

```
streamlit run app.py
```

- Access the platform at `http://localhost:8501` in your browser.

C Video Presentation Link

Watch the project video presentation here:
`https://drive.google.com/file/d/1yzAd-a-oMA1GdtCRqtzfJM3mkhcr0Eh5/view?usp=drive_link`

D Team Members

- **Kshitij Sonje**
- **Anya Gupta**
- **Priyanshi Peswani**