Detailed Report: AI-Powered Ticketing System

Hybrid AI ITSM framework

1. Introduction

This report documents the design, development, and implementation of an **AI-assisted IT Ticketing System**. The system combines **automation**, **AI-powered classification and resolution**, and **human-in-the-loop escalation**, ensuring efficiency while maintaining reliability.

The primary goal is to reduce IT support workload, shorten resolution times, and build a **self-learning knowledge base** for future queries.

---

2. Planning & Design Tools

Effective planning and design ensured clarity before development:

- **Draw.io** – Used to create architecture diagrams, workflow charts, and escalation flows.
- **Microsoft OneNote** – Served as a central repository for brainstorming, documentation, and meeting notes.
- **Dribbble** – Provided UI/UX inspiration to create a clean and intuitive interface for users.

---

3. Core Technologies

The system is built on a modern, lightweight stack.

Backend

- **Python** – Primary programming language for backend logic.
- **Flask** – Web framework enabling API endpoints, routing, and business logic.
- **MySQL** – Database used to store tickets, users, feedback, and knowledge base entries.

Frontend

- **HTML5** – Provides the structural layout of all pages.
- **CSS3** – Used for styling, theming, and responsive layouts.

- **JavaScript** – Powers dynamic interactions, form validations, and dashboard interactivity.
- **Jinja2** – Templating engine used with Flask for rendering dynamic HTML content.
- **Bootstrap** – Ensures responsive design for desktops, tablets, and mobile devices.

---

4. AI & Integration

The intelligence of the system comes from **OpenAI APIs** and **AMCharts** for visualization.

OpenAI API

- **Function Calling** – Automates classification by extracting severity, urgency, category, and agent assignment from user prompts.
- **Text Completion** – Used for generating natural responses and explanations.
- **Natural Language Processing** – Detects intent and sentiment in user queries.

AMCharts

- **Interactive Data Visualization** – Charts and graphs built for real-time ticket tracking.
- **Custom Dashboards** – Allows IT managers to view workload distribution, feedback, and performance metrics.
- **Real-Time Analytics** – Helps measure AI vs human IT contributions.

---

5. Development Tools

- **Git** – Version control for tracking changes.
- **GitHub** – Central repository for collaborative code management.
- **VS Code** – Development environment for coding and debugging.
- **GitHub Copilot** – AI coding assistant that improved development speed by suggesting code snippets and reducing boilerplate effort.

---

6. Training Data

A **synthetic dataset** was generated using ChatGPT to simulate realistic tickets.

- **Ticket Categories** – 6 major categories covering IT support:
    1. Access & Authentication
    2. Networking & Connectivity
    3. Hardware / Device
    4. Software / Applications
    5. Collaboration Tools

6. Security & Compliance

- **Sample Queries** – 10 common issues per category, each with 10 variations (5 short, 5 detailed).
- **Resolution Templates** – Draft solutions generated to train AI for consistent responses.
- Each entry was tagged with **severity** and **urgency** to aid decision-making.

---

7. System Workflow

Ticket Creation

- User enters **Name, Email, Development Center, and Issue Statement**.

Classification

- Input is passed to **OpenAI Function Calling**, returning:
  - Severity
  - Urgency
  - Category
  - Suggested Agent

Assignment

- If issue is **low severity & urgency** → handled by **AI Agent**.
- If **high severity** or **complex** → assigned to **IT Team**.
- Tickets not assigned → marked as **IT TEAM (pool)**, later assignable to individuals.

Resolution Lifecycle

1. AI/Agent provides a resolution.
2. User can **close the ticket** or add **comments**.
3. If AI answers → closure option shown after 10 seconds.
4. Closed tickets are sent to **Knowledge Base** for future use.
5. If unresolved in **24 hours** → escalation triggered.

Feedback

- Upon closure, users provide **emoji-based feedback**.
- Feedback metrics logged for analytics.

---

8. Analytics Dashboard

Developed with **AMCharts**, the dashboard contains:

1. **Tasks Distribution (Bar Chart)**
    - AI Agents vs IT Team vs Not Allocated.
2. **AI vs IT Team (Pie Chart)**
    - Percentage of tickets resolved by AI vs human IT.
3. **Feedback (Bar Chart)**
    - Breakdown of emoji-based feedback scores.
4. **Resolution Time (Figures)**
    - Average resolution time per AI agent.
    - Average resolution time for IT team.

---

9. Knowledge Base

Every closed ticket (AI or IT team) is stored with:

- Original Query
- Resolution Provided
- Metadata: category, severity, urgency

This allows users to **search past resolutions** for faster self-service.

---

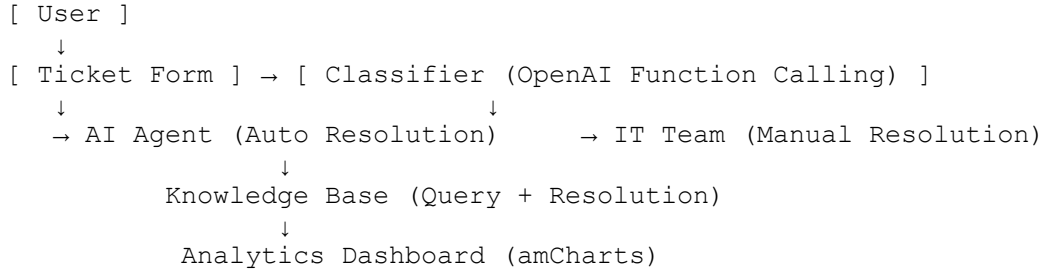
10. Future Roadmap

- **Proactive Resolution** – Suggest fixes from the knowledge base before ticket submission.
- **Advanced Sentiment Analysis** – Auto-close tickets when positive closure intent is detected.
- **Gamification of Feedback** – Reward users for giving consistent feedback.
- **Org-Level Insights** – Compare development centers to identify recurring IT bottlenecks.

---

11. Architecture Diagram (Conceptual)

(Built using Draw.io)



---

12. Conclusion

This system demonstrates how **AI + human collaboration** can drastically improve IT support efficiency. While not a fully autonomous agent yet, the use of **OpenAI function calling, GitHub Copilot, and amCharts** provides a strong foundation.

It enables:

- Faster resolution times
- Reduced IT workload
- Continuous learning via knowledge base
- Transparent analytics for decision-making
