iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

SERVICE DESK MANAGEMENT SYSTEM

Pon Rajalakshmi N
Register No: 23105818 | Mobile: 9566713811
Bachelor of Computer Applications (BCA) - In Progress,
Department of Computer Applications, India

ABSTRACT
The Service Desk Management System is a web-based application designed to streamline the process of handling user queries, complaints, and service requests within an organization. In many institutions, managing issues manually leads to delays, lack of transparency, and inefficient communication between users and administrators. This project provides an automated and centralized platform where users can raise tickets and administrators can manage, track, and resolve them. The application is developed using the Django framework with SQLite database and provides an intuitive interface with features such as ticket creation, status tracking, user management, and report generation.

Keywords:
Service Desk, Ticket Management, Django, Web Application, SQLite, Issue Tracking, Admin Panel, User Authentication

.

INTRODUCTION

In today's digital environment, organizations receive a large number of service requests, complaints, and technical issues from users. Managing these requests manually through phone calls, emails, or paper-based systems often leads to confusion, delays, and lack of proper tracking. The Service Desk Management System provides a centralized platform where users can submit their issues and administrators can monitor, manage, and resolve them in a structured manner. The system is developed using the Django framework — a high-level Python web framework — and SQLite as the backend database, targeting colleges, universities, IT organizations, and any institution that handles user service requests.

EXISTING SYSTEM

Before automated service desk systems, organizations relied on manual methods such as phone calls, emails, and paper records. These approaches lacked proper structure, led to delayed responses, data loss, lack of transparency, and high manual workload. Commercial tools like Zendesk, Freshdesk, and ServiceNow offer advanced features but are costly, complex to set up, and not suitable for small-scale or customized deployments. This creates a need for a simple, cost-effective, and customizable solution tailored to organizational needs.

PROPOSED SYSTEM

The proposed system is a web-based application following a three-tier architecture: a Presentation Layer (HTML, CSS, JavaScript), an Application Layer (Python/Django), and a Database Layer (SQLite). The User Module supports registration, login, and ticket creation. The Admin Module enables viewing, updating, and resolving tickets. The Ticket Management Module generates unique ticket IDs and stores issue details. Key features include easy ticket creation, real-time status tracking, centralized data storage, user-friendly interface, and secure authentication.

METHODOLOGY

The system follows the Software Development Life Cycle (SDLC). Requirement Analysis identified the needs of users and administrators. System Design planned the UI, database schema (User and Ticket tables), and system workflow. Implementation used Django for backend, HTML/CSS for frontend, and SQLite for data storage. Testing covered unit, integration, functional, usability, and boundary testing to ensure all features work correctly. Deployment was done on a local server, and the system is maintained regularly for updates and security improvements.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

SYSTEM IMPLEMENTATION

The development environment is summarized in Table 1 below.

Component	Technology Used
Programming Language	Python
Framework	Django
Frontend	HTML, CSS, JavaScript
Database	SQLite
IDE	Visual Studio Code

Table 1: Development Environment

The system is organized into modules: User Module (registration, login, ticket creation), Ticket Module (unique ticket ID, title, description, date, status), and Admin Module (view all tickets, update statuses, manage users). The database stores user records (user_id, name, email, password) and ticket records (ticket_id, user_id, issue, status, date).

SYSTEM TESTING

Testing was performed to verify the system meets all requirements. Unit Testing checked individual modules, Integration Testing verified module interactions, Functional Testing confirmed all features work correctly, Usability Testing ensured user-friendliness, and Boundary Testing covered edge cases. The testing environment used Windows 10/11, Google Chrome, Python, and SQLite.

Test cases are shown in Table 2 below.

Test ID	Description	Input	Expected Result	Status
TC001	User Login	Valid credentials	Login successful	Pass
TC002	Invalid Login	Wrong credentials	Error message	Pass
TC003	Ticket Creation	Issue details	Ticket created	Pass
TC004	Empty Fields	No input	Validation shown	Pass
TC005	Status Update	Change to Closed	Status updated	Pass

Table 2: Sample Test Cases

RESULTS AND DISCUSSION

The system was successfully implemented and tested. The Login Page enables authenticated access. The User Dashboard shows ticket counts by status (Pending, In Progress, Resolved). The Ticket Submission Page allows users to raise complaints with subject, department, priority, and description. The Admin Dashboard provides a complete overview of all tickets. The Admin Manage Tickets page enables status updates and priority assignment. The Ticket Status Tracking page provides real-time visibility to users. All modules performed as expected with minimal resource usage, fast response times, and no major errors during testing.

CONCLUSION

The Service Desk Management System was successfully designed and implemented to streamline user complaint handling and service request management. The system provides an efficient, centralized platform that overcomes the limitations of traditional manual systems by improving transparency, reducing response time, and ensuring proper record maintenance. The Django framework and SQLite database make the system reliable, secure, and easy to maintain. In future, enhancements such as mobile application support, AI-based chatbot integration, Email/SMS notifications, advanced analytics dashboard, role-based access control, cloud deployment, and file attachment support can be added.

REFERENCES

[1] Django Software Foundation. (2023). Django Documentation. Retrieved from https://docs.djangoproject.com/
[2] Python Software Foundation. (2023). Python Language Reference. Retrieved from https://www.python.org/
[3] SQLite Consortium. (2023). SQLite Documentation. Retrieved from https://www.sqlite.org/docs.html

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

[4] Turban, E., Volonino, L., & Wood, G. R. (2015). Information Technology for Management. John Wiley & Sons.
[5] Sommerville, I. (2016). Software Engineering (10th ed.). Pearson Education.
[6] Zendesk Inc. (2023). Zendesk Help Center Platform Overview. Retrieved from https://www.zendesk.com/
[7] Mozilla Developer Network. (2023). Web Technologies Documentation. Retrieved from https://developer.mozilla.org/
[8] Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education.