Project Name: I.T Support Ticketing System

Date: September 14, 2024

Document Version: 1.0

Prepared By: Mr. Brown, Thomas

# Introduction

The IT Support Ticketing System is a platform designed to facilitate efficient communication between customers and IT staff. The system allows customers to submit support requests (tickets), while IT staff manage, respond to, and resolve these tickets. This document outlines the detailed functional requirements needed to build the system. These requirements will guide the development team through the system’s design and implementation, ensuring that all functionalities meet the business objectives.

# Functional Overview

## The system comprises two primary components:

* **Customer Dashboard**: Used by customers to create and manage support tickets.
* **IT Admin Dashboard**: Used by IT staff to view, manage, and resolve tickets.

Both components will integrate with a backend system that handles real-time notifications, ticket updates, and secure user authentication.

# Functional Requirements

## Customer Dashboard

### Ticket Creation

* **Description**: Customers must be able to create a support ticket by entering the issue type, description, priority, and attaching relevant files.
* **Inputs:** Issue type (dropdown), description (text area), priority (dropdown), file attachment (upload option).
* **Process:** Upon submission, the system saves the ticket in the database and assigns an initial status of “Open.”
* **Outputs:** A confirmation message and the ticket status.

### Ticket Status Tracking

* **Description:** Customers must be able to view the status of their submitted tickets, categorized as Open, In Progress, or Resolved**.**
* **Inputs**: Customer selects a specific ticket.
* **Process**: The system retrieves the status from the database and displays the ticket details.
* **Outputs**: Ticket details, including current status and responses from IT staff.

### File Attachments

* **Description**: Customers can attach files (e.g., screenshots or logs) to their support tickets.
* **Inputs**: File upload input.
* **Process**: The system saves the uploaded file to the server and links it to the respective ticket.
* **Outputs**: Confirmation of file upload and its availability for IT staff.

### Real-Time Notifications

* **Description**: Customers must receive real-time notifications for ticket status changes or responses from IT staff.
* **Inputs**: None (event-driven).
* **Process**: The system uses WebSocket or Pusher to push notifications to the customer dashboard
* **Outputs**: A notification showing the updated status or response.

## IT Admin Dashboard

### View and Manage Tickets

* **Description**: IT staff can view all submitted tickets and filter them based on criteria such as priority, status, and customer.
* **Inputs**: Filter options (e.g., dropdowns for status, priority).
* **Process**: The system retrieves tickets from the database based on the selected filters.
* **Outputs**: A list of tickets that match the selected filters, along with their details..

### Ticket Assignment

* **Description**: IT staff must be able to assign tickets to other team members for resolution.
* **Inputs**: Selection of a team member from a dropdown list.
* **Process**: The system updates the ticket in the database with the assigned team member’s details.
* **Outputs**: Confirmation of ticket assignment.

### Real-Time Updates

* **Description**: IT staff must receive real-time updates whenever new tickets are submitted or ticket statuses change.
* **Inputs**: None (event-driven).
* **Process**: The system pushes updates using WebSocket or Pusher.
* **Outputs**: Real-time notifications.

### Ticket Resolution

* **Description**: IT staff must be able to update tickets with a resolution and close them once the issue is resolved.
* **Inputs**: Resolution description (text area), status change (dropdown to set as "Resolved").
* **Process**: The system updates the ticket with the resolution details and marks it as "Resolved."
* **Outputs**: Confirmation of ticket resolution.

### Audit Log

* **Description**: All actions taken on a ticket (e.g., status changes, assignments) must be logged for future reference.
* **Inputs**: Actions performed on tickets.
* **Process**: The system creates log entries for every action related to a ticket.
* **Outputs**: Audit logs viewable by IT staff.

## IT Admin Dashboard

### JWT-Based Authentication

* **Description**: Users must authenticate with a JWT-based token upon login. The system will validate the token for each request.
* **Inputs**: Username, password.
* **Process**: The system generates a JWT upon successful login and stores it client-side for subsequent requests.
* **Outputs**: JWT token, or an error message if login fails.

### Role-Based Access Control

* **Description**: The system must enforce role-based access control to differentiate between customer and IT admin access.
* **Inputs**: User role (Customer, IT Admin).
* **Process**: Based on the role, the system grants or restricts access to certain features.
* **Outputs**: Access granted or denied based on the user's role.

## Real-Time Communication

### WebSocket Integration

* **Description**: The system must use WebSocket or Pusher to handle real-time communication between the frontend and backend for ticket updates and notifications.
* **Inputs**: None (event-driven).
* **Process**: The WebSocket connection is maintained, and updates are pushed in real-time.
* **Outputs**: Notifications for ticket updates, new tickets, and status changes.

# System Requirements

## Frontend

* **Framework**: React
* **Real-Time Communication**: WebSocket or Pusher for real-time updates.
* **UI Library**: Ant Design or Material UI for UI components.

## Backend

* **Framework**: Django with Django REST Framework (DRF).
* **Authentication**: JWT-based using Django rest framework-simple jwt.
* **Real-Time Communication**: Django Channels for WebSocket integration.
* **Database**: PostgreSQL for relational data, Redis for session caching (optional).
* **Deployment**: Cloudways or other cloud platforms supporting Django and PostgreSQL.

# Non-Functional Requirements

## Performance

* **Response Time:** API response times should be under 200ms for standard requests.
* **Real-Time Latency**: Notifications and updates should be delivered in real-time with minimal delay (under 2 seconds).

## Security

* **Data Security**: Sensitive data must be encrypted during transit and at rest. Passwords must be securely hashed.
* **Authentication**: Use JWT for secure token-based authentication.

## Scalability

* The system must support concurrent users and real-time updates without performance degradation.

# Future Enhancements

## Mobile Application

* **Description:** Future versions of the system will include mobile apps for customers and IT staff, allowing them to submit and manage tickets on the go.

## Advanced Analytics

* **Description:** Reporting features to track ticket trends, resolution times, and IT staff performance.

## Chat Feature

* **Description:** Implement a live chat feature between customers and IT staff to facilitate faster issue resolution.

# Assumptions

* The system will have internet access for real-time WebSocket communication.
* The users will be familiar with basic IT ticketing systems.
* Cloudways or similar cloud service will be used for hosting.

# Constraints

* Deployment may be delayed due to budgetary limitations for additional features like mobile app integration and advanced analytics.

# Dependencies

* Django Channels: Required for real-time communication.
* PostgreSQL: For storing user, ticket, and audit log data.
* Redis: For optional session caching.

# Approvals

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Signature** |
| Mr. Thomas | Project Owner | [Signature] |
| IT Support Team | End-Users | [Signature] |
| Development Team | Implementation | [Signature] |

This document outlines the necessary functional requirements for the IT Support Ticketing System to ensure that both customers and IT staff have a smooth and efficient experience. It serves as the blueprint for the development and testing teams to build a system that meets business objectives and performance standards.