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 web-based platform designed to streamline the process of submitting, managing, and resolving IT support tickets. This document provides a detailed description of the technology stack, database design, and third-party services used in the system.

# Technology Stack

## Frontend

* **Framework**: React.js
* **Description**: React is used for building the user interfaces for both the Customer Dashboard and IT Admin Dashboard. It allows for dynamic and interactive components.
* **State Management**: Context API (or Redux if more complex state management is needed).
* **UI Libraries**: Ant Design or Material UI for pre-built, responsive components.
* **Real-Time Communication**: WebSocket (for live notifications and ticket updates)
* **Description**: WebSocket is used to provide real-time updates between the frontend and backend. It ensures that ticket updates and notifications are instantly reflected on both customer and IT admin dashboards without page reloads.
* **Routing**: React Router
* **Description**: React Router is used for client-side routing, enabling navigation between the various sections of the platform (e.g., Ticket Submission, Ticket Management).

## Backend

* **Framework:** Django (Python)
* **Description:** Django is used as the backend framework for handling business logic, API requests, and connecting to the database.
* **Django REST Framework (DRF):** Used to build the API endpoints that the frontend communicates with. This handles requests for ticket submission, updates, status tracking, and user authentication.
* **Real-Time Communication: Django Channels**
* **Description:** Django Channels allows for WebSocket integration in Django, enabling real-time communication between the frontend and backend for live ticket updates and notifications.
* **Authentication:** JWT (JSON Web Token)
  + **Description**: JWT is used for token-based authentication. When users log in, they receive a secure token that is used to authenticate their requests without needing to store session data on the server.
  + **Library**: djangorestframework-simplejwt
* **Task Scheduling**: Celery (optional)
  + **Description**: Celery can be used for asynchronous task management (e.g., sending email notifications, running background jobs).

## Database

* **Database Management System (DBMS):** PostgreSQL
* **Description:** PostgreSQL is the relational database used to store all system data, including users, tickets, comments, audit logs, and attachments.

**Database Tables**

1. **Users Table**:
   * **Columns**: user\_id (PK), name, email, password (hashed), role, created\_at, updated\_at.
   * **Description**: Stores details about customers and IT admins, including hashed passwords and role-based access control.
2. **Tickets Table**:
   * **Columns**: ticket\_id (PK), user\_id (FK to Users), title, description, priority, status, assigned\_to (FK to Users), resolution, created\_at, updated\_at.
   * **Description**: Stores all the data related to support tickets.
3. **Ticket Comments Table**:
   * **Columns**: comment\_id (PK), ticket\_id (FK to Tickets), user\_id (FK to Users), comment\_text, created\_at, updated\_at.
   * **Description**: Stores the comments added by users and IT admins on each ticket.
4. **Audit Logs Table**:
   * **Columns**: log\_id (PK), user\_id (FK to Users), action, ticket\_id (FK to Tickets), description, created\_at.
   * **Description**: Records all critical actions performed on tickets for accountability.
5. **Attachments Table**:
   * **Columns**: attachment\_id (PK), ticket\_id (FK to Tickets), file\_path, created\_at.
   * **Description**: Stores file attachments uploaded by customers or IT admins for each ticket.

* **Relationships**:
  + One-to-Many: A user can have multiple tickets.
  + One-to-Many: A ticket can have multiple comments.
  + One-to-Many: A ticket can have multiple attachments.
  + One-to-One: A ticket is assigned to a single IT admin.
* **Caching**: Redis (Optional)
  + **Description**: Redis can be used for session caching or improving the performance of frequent queries.

## APIs and Endpoints

**Authentication API**

* **POST /api/auth/login**: Login using email and password, returns a JWT token.
* **POST /api/auth/register**: Register a new user (Customer or IT Admin).
* **POST /api/auth/logout**: Logs the user out and invalidates the JWT token.

**Ticket Management API**

* **GET /api/tickets/**: Retrieve a list of all tickets (filtered by status, priority, etc.).
* **POST /api/tickets/**: Create a new ticket (for customers).
* **GET /api/tickets/{ticket\_id}/**: Retrieve details of a specific ticket.
* **PUT /api/tickets/{ticket\_id}/**: Update ticket details (admin only).
* **DELETE /api/tickets/{ticket\_id}/**: Delete a ticket (admin only).

**Comment API**

* **POST /api/tickets/{ticket\_id}/comments/**: Add a comment to a specific ticket.
* **GET /api/tickets/{ticket\_id}/comments/**: Retrieve all comments for a ticket.

**Real-Time WebSocket API**

* **/ws/tickets/{ticket\_id}/updates**: WebSocket connection for real-time ticket updates (notifications for both customers and IT admins).

## Third-Party Services

**Cloud Hosting: Cloudways**

* **Description**: Cloudways will be used for hosting the application. It supports Django and PostgreSQL, and offers scalability and performance optimization.
* **Features**:
  + Managed hosting environment.
  + Integrated Git deployment for easy updates.
  + Automated backups and server monitoring.

**Email Service: SendGrid (Optional)**

* **Description**: SendGrid can be used to send system-generated emails, such as ticket updates, password resets, or new ticket notifications.

# Security and Data Protection

## J WT Authentication

* **Description**: JWT is used for token-based authentication to ensure secure, stateless communication between the frontend and backend.
* **Token Expiry**: Tokens are configured to expire after a set time (e.g., 1 hour), and refresh tokens can be used for re-authentication

## Data Encryption

* **In Transit**: All communication between the frontend and backend will be encrypted using HTTPS (SSL/TLS).
* **At Rest**: Sensitive data such as passwords will be hashed using bcrypt, and important data will be encrypted in the database.

## Role-Based Access Control

* **Description**: The system implements role-based access, ensuring that customers can only access their own tickets, while IT admins have access to all tickets based on their assignments.

# Scalability and Performance

## Horizontal Scaling

* **Description**: The system will be designed to scale horizontally, meaning additional servers can be added as the user base grows, without significantly degrading performance.

## Caching

* **Redis**: Redis can be implemented to cache frequently accessed data and reduce the load on the database.

## Load Balancing

* **Description**: Load balancing can be introduced to distribute traffic across multiple server instances, ensuring high availability and reliability during peak times.

# Monitoring and Maintenance

## Monitoring

* **Description**: Cloudways offers server monitoring tools to track resource usage (CPU, memory, bandwidth) and detect potential bottlenecks.
* **Error Tracking**: Error logs will be tracked, and real-time alerts will be configured to notify admins of critical issues.

## Backups

* **Description:** Automated daily backups will be configured on Cloudways to ensure data recovery in case of system failure.

# Conclusion

This technical specifications document provides an overview of the technologies, database design, and third-party services that will be used to develop, host, and maintain the IT Support Ticketing System. The chosen technologies ensure the system is scalable, secure, and performant, providing a seamless experience for both customers and IT admins.