PHASE 3 PROJECT DESIGN PHASE DOCUMENT

Streamlining Ticket Assignment for Efficient Support Operations

TEAM ID	**NM2025TMID08979**
Team Leader	Yogesh S
Team Member	Selvin Joshva A
Team Member	Naveen P
Team Member	Ravin Akash S

Introduction:

In modern support systems, manual ticket assignment often results in inefficient resource utilization, longer response times, and reduced customer satisfaction. This **design phase** focuses on translating the project's planning and ideation into a **detailed technical and architectural blueprint**, defining the structure, components, data flow, and integration methods of the system.

The goal is to create a scalable, modular, and AI-driven system that can automate ticket categorization and assignment while being flexible enough to integrate with existing ticketing tools.

System Overview:

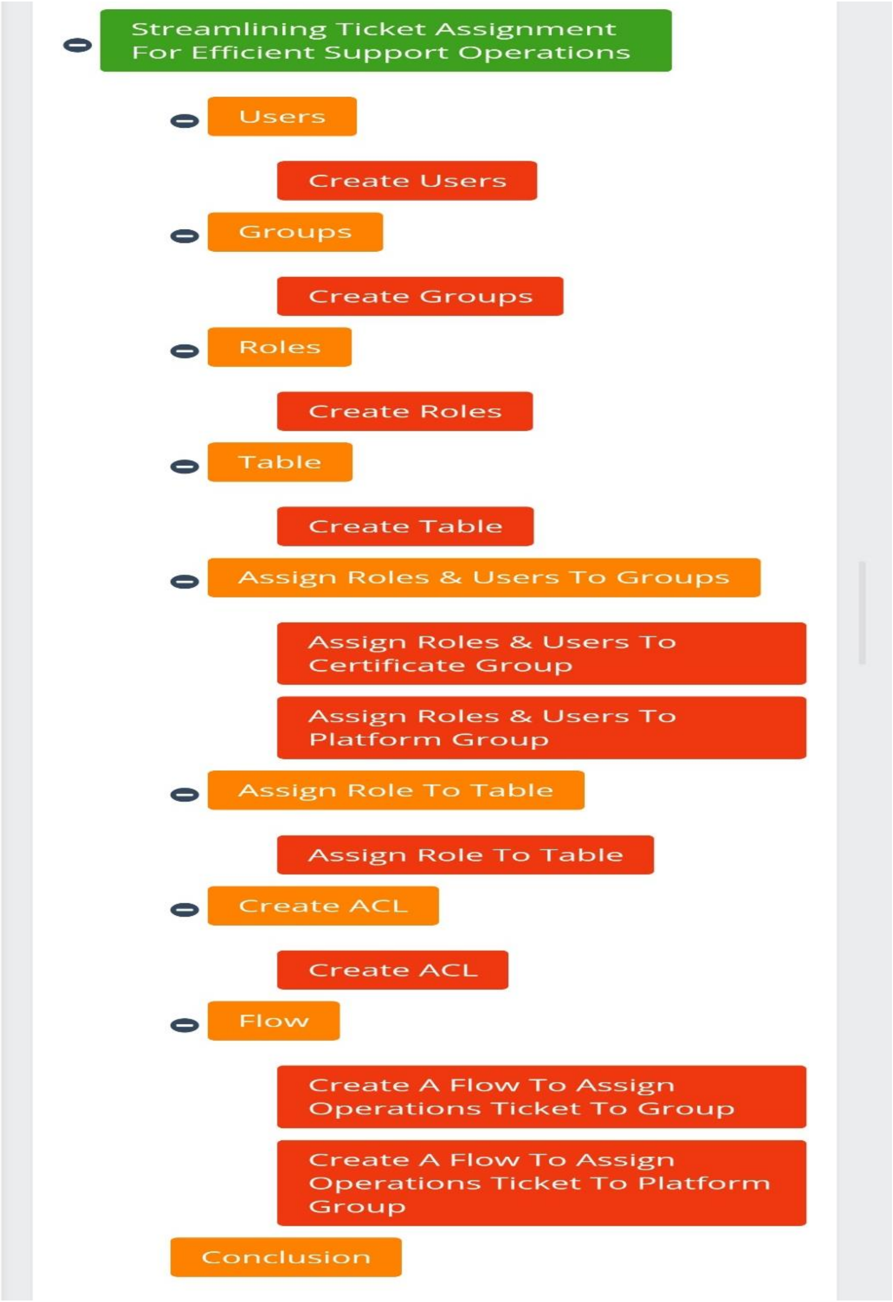
The **AI-powered ticket assignment system** automatically categorizes and assigns support tickets to the most appropriate agent based on:

- Ticket **type** (technical, billing, account-related, etc.)
- Agent **skillset**

- Current **workload** and **availability**
- Ticket **priority** and **customer sentiment**

A web-based dashboard provides **real-time monitoring** of ticket distribution, performance metrics, and SLA compliance.

Workflow Diagram:


Streamlining Ticket Assignment For Efficient Support Operations
Users
Create Users
Groups
Create Groups
Roles
Create Roles
Table
Create Table
Assign Roles & Users To Groups
Assign Roles & Users To Certificate Group
Assign Roles & Users To Platform Group
Assign Role To Table
Assign Role To Table
Create ACL
Create ACL
Flow
Create A Flow To Assign Operations Ticket To Group
Create A Flow To Assign Operations Ticket To Platform Group
Conclusion

Module Design:

Module Name	Description	Technology Used	Output
Ticket Intake Module	Collects tickets from forms or APIs	Flask, REST API	Stores ticket details
NLP Classification Module	Analyzes ticket text and predicts category	Python, SpaCy / BERT	Category & priority
Assignment Engine	Matches tickets to agents based on skill and workload	Python, Scikit-learn	Assigned agent ID
Database Module	Stores ticket, agent, and performance data	PostgreSQL / MongoDB	Persistent data store
Dashboard Module	Displays tickets, workload, and analytics	React / HTML / CSS	Real-time visualization
Notification Module	Sends alerts to assigned agents or supervisors	Flask-Mail / Webhooks	Email or dashboard alert

Data Flow Diagram (DFD):

Level 0 – Context Diagram

```
[User/Customer] → [Ticket Intake System] → [Ticket Assignment System] →
[Support Agent]
```

Level 1 – Detailed Flow

```
User → Submit Ticket → API → NLP Engine → Classify Ticket
                                        ↓
           Assignment Engine
                                        ↓
           Database Storage
```

```
                                   ↓
             Dashboard Display
                                   ↓
              Agent Notification
```

Database Design:

Entity Relationship (ER) Diagram (Conceptual):

```
+-------------+     +------------------+     +------------------+
|  Ticket    |-------|   Assignment    |-------|     Agent       |
+-------------+     +------------------+     +------------------+
| Ticket_ID   |      | Assignment_ID   |      | Agent_ID        |
| Description |      | Ticket_ID       |      | Name            |
| Category    |      | Agent_ID        |      | Skillset        |
| Priority    |      | Assigned_Date   |      | Availability    |
| Status      |      | Status          |      | Current_Load    |
+-------------+     +------------------+     +------------------+
```

Tables Overview:

Table	Description	Key Fields
Ticket	Stores ticket details	Ticket_ID, Description, Category, Priority
Agent	Stores agent data	Agent_ID, Name, Skillset, Workload
Assignment	Tracks who handled which ticket	Assignment_ID, Ticket_ID, Agent_ID, Date
Analytics	Tracks metrics and performance	Agent_ID, Avg_Resolution_Time, SLA_Compliance

Algorithm / Logic Flow:

Ticket Assignment Algorithm:

1. Receive new ticket input.
2. Preprocess text (remove stopwords, tokenize, lemmatize).
3. Use NLP model to predict **category** and **urgency level**.
4. Query database for agents with matching skillset.
5. Calculate workload and availability scores.
6. Assign ticket to the **best-matched agent** (lowest load, highest relevance).
7. Update database and send notification.
8. Display on dashboard.

User Interface Design:

Key Screens (Wireframe Overview):

Screen	Description	Main Elements
Login Page	Secure login for support team	Username, Password
Dashboard	Overview of tickets and workload	Graphs, Metrics, Status Summary
Ticket View	View and manage tickets	Ticket details, Assigned agent
Agent View	View agent status	Availability, Performance
Admin Panel	Manage users, monitor system	Settings, Reports

Technology Stack:

Layer	Technology	Purpose
Frontend	React.js / HTML / CSS	UI for users and admins
Backend	Flask (Python)	Business logic and API handling
Database	PostgreSQL / MongoDB	Data management
AI/NLP	SpaCy / BERT / TensorFlow	Ticket classification
Deployment	Render / Netlify	Hosting & CI/CD
Version Control	GitHub	Collaboration & code management

Security and Privacy Design:

- Role-based authentication (Admin / Agent).
- Encrypted communication using HTTPS.
- Secure database access with credentials protection.
- Data anonymization for training datasets.
- Regular audit logs for ticket modifications.

Integration Design:

- RESTful APIs for seamless communication between front and backend.
- JSON data exchange format.
- Integration-ready endpoints for tools like **Jira**, **Zendesk**, or **ServiceNow**.

Performance Considerations:

- Use of **asynchronous task handling** for ticket processing.
- Implement caching for frequent queries.
- Scalable deployment through containerization (Docker).
- Optimize NLP models for faster inference (use distilled models like *DistilBERT*).

Expected Design Outcomes:

- A **modular, scalable architecture** ready for implementation.
- Clearly defined data flow and system interaction points.
- Blueprint for seamless integration and deployment.
- User-friendly interface mockups for real-time monitoring.

Conclusion:

The design phase provides a detailed technical roadmap for implementing the **Streamlining Ticket Assignment System**. With modular components, a robust architecture, and AI integration, the project ensures efficient, intelligent, and scalable ticket management across support operations.