Call Center Ticket Assignment System – Frontend Challenge

Overview

This project is a smart ticket assignment system for a call center. The backend has already been implemented and exposes a RESTful API (see backend code attached in the email). Your task is to build the **frontend application** that interacts with this backend and provides a clean, intuitive UI for agents and operations.

You'll build a dashboard that allows users to view:

- System status (active calls, messages, completed tasks)
- Current tasks
- Agent overall workloads
- Active agents and their current task distribution

A sample UI wireframe is provided below for reference and inspiration.

The Task

Build a frontend application that implements the following functionality:

Core Functionality

1. **Dashboard View**

- Displays:
  - Number of active calls, messages, and completed tasks
  - Total tickets in queue
  - Agent workload and task distribution
- *Hint: check the debug endpoint*

2. **Tasks View**

- Segregate task list by type: voice and text
- Display empty state if no tasks are in queue
- Allow the user to create a task from the view

3. **Agent View**

- Display each agent’s current task load and capacity usage
- Show voice call/task indicators and capacity bar
- Allow the user to create new agents and assign/edit agent's skills

4. **Active Agents Section (on Dashboard)**

- Show agent name, number of supported languages, and task capacity
- View agent task status (calls and messages count)

5. **Actions**

- Reset system (`POST /reset`)
- Assign ticket (`POST /tickets/assign`)
- Complete task (`POST /tasks/complete`)
- Register new agent (`POST /agents`)

API Endpoints Summary

Attached you will find the backend server which you can test and run locally. The code includes a detailed readme file explaining its functionality and endpoints. Your frontend application should use it as a server.

> The backend server must be running locally at: `http://localhost:8000`

You can access the API documentation here: http://localhost:8000/docs

---

Requirements

- Use **TypeScript**
- Use **React** (or a framework like Next.js)
- Implement a **responsive** layout
- Build modular, maintainable components
- Interact with the provided REST API
- Cleanly organize the app into routes or pages

Nice to Have

- End-to-End Tests using **Playwright** or **Cypress**
- Use of component libraries (e.g., Radix UI, ShadCN)
- Type-safe API integration
- **Implement dynamic reassignment mechanism**: Queued tasks (e.g. Korean call) should be immediately assigned if a matching agent is added or gains the necessary language skill. At the moment, this functionality is not available in the backend code.

Submission Instructions

Please submit a GitHub repository with:

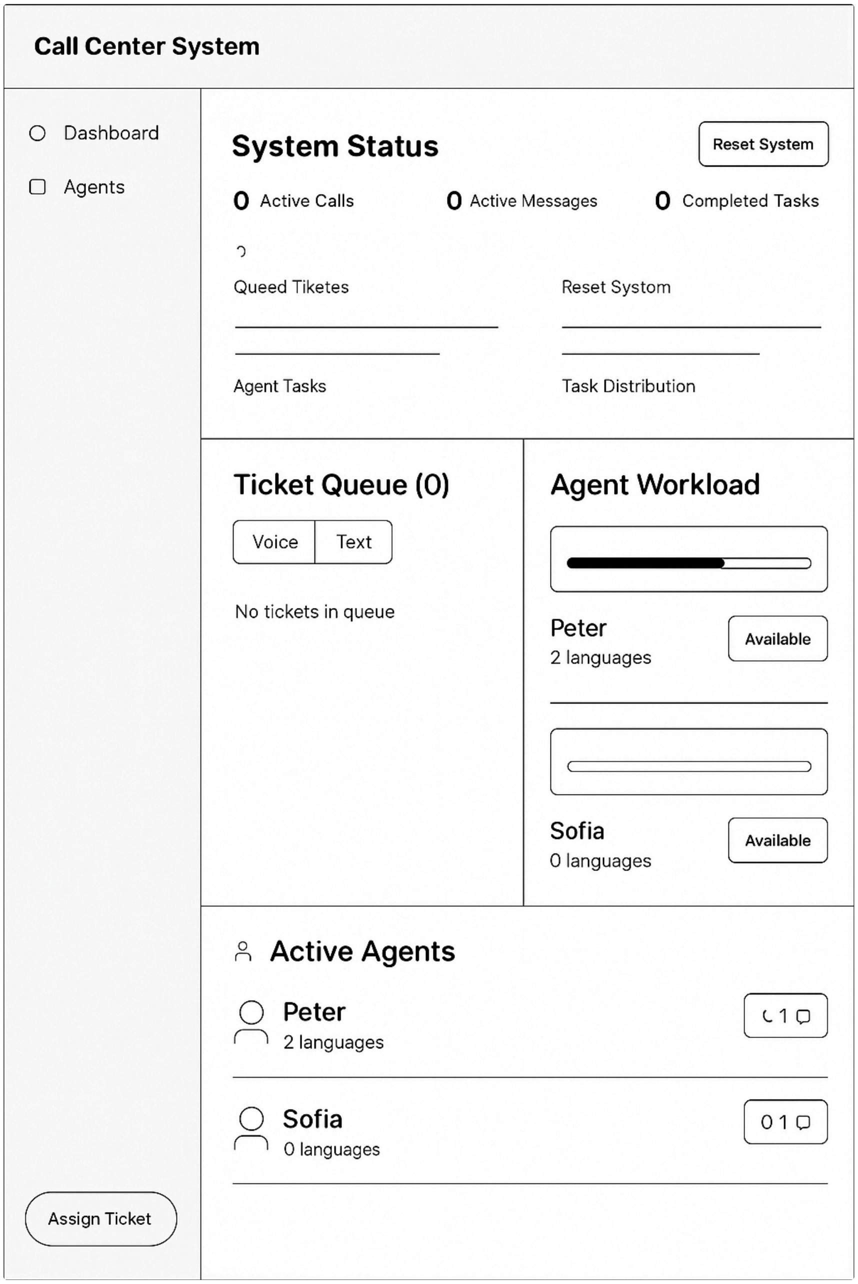
- A `README.md` file with instructions on how to run the application and personal notes explaining parts of the code (as you see necessary)
- All source code
- Any test results or setup scripts

Bonus Tips

- Think like a user — what would you want to see first on the dashboard?
- Use visual cues like icons, badges, and progress bars to improve usability
- Your design doesn’t need to be pixel-perfect, but it should be clear, tidy, and functional

Wireframe

Please use this wireframe as inspiration and rough guidance.



System Status

0 Active Calls

0 Active Messages

0 Completed Tasks

Queed Tiketes

Reset Systom

Agent Tasks

Task Distribution

Ticket Queue (0)

No tickets in queue

Agent Workload

Peter
2 languages

Sofia
0 languages

Active Agents

Peter
2 languages
1

Sofia
0 languages
0 1