TEAM 2025104

**Table of Contents**

[**1. Introduction………………………………………………………………………**](#_xosgfzxh56rn)**…01**

[1.1 Statement of Purpose](#_6m9g5d5qow01)

[1.2 Product Scope](#_lvthzkk7m1ru)

**2.** [**System Overview and Technical Blueprint**](#_z3jymcuke42o)**……………………………………….02**

[2.1 System Architecture](#_ofjsob22hftg)

[2.3 Entity Relation Diagram](#_8tje3evagx8v)

[**3. Intelligent Employee Assessment Framework**](#_yoimrdizq3wi)**……………………………………03**

[3.1 Unified Data Pipeline and Feature Engineering](#_wlqhjzswae6x)

[3.2 Targeted Employee Detection and Engagement Logic](#_rsni5u1amkc5)

[3.3 Question Bank Development](#_ryk57n1gzqcs)

[**4. Artificial Intelligence Models and Supporting Toolchains………………………**](#_ppozl360us57)**05**

[4.1 AI Model Architecture and Fine-Tuning Strategy](#_7q4ohmoppxmt)

[4.2 Intelligent Conversation Orchestration](#_9kbo7nn5fvaq)

[4.3 HR Analytics Conversation Processing Pipeline](#_vp0kxvu4mjs)

[**5. Deployment, Integration, and Runtime Environment…………………………**](#_w9klgc3tj5kf)**…09**

[5.1 Step-by-Step Deployment Instructions](#_rptsxwp928wt)

[5.2 Environment Configuration and Optimization](#_fnd8id6vxhup)

[5.3 CI/CD Pipeline Overview](#_l9viecf6kcov)

[**6. Daily Report Generation and Analysis……………………………………………**](#_ukd37xbr1821)**10**

[6.1 Report Structure and Content](#_804iolrzfvit)

[**7. Source Code Architecture and Explanation………………………………………**](#_u6kd7zfwrom5)**11**

[7.1 Backend Implementation](#_i6w3by2in5vs)

[7.2 Frontend Architecture](#_fayu0h80nnq0)

[7.3 Database Schema and Integration](#_9auh0nn9o2tn)

[**9.Requirement**](#_8z5m2b4mx06d)**s………………………………………………………………………...15**

### **1. Introduction**

### 1.1 Statement of Purpose

In today’s dynamic work environment, understanding employee sentiment is vital. While Deloitte’s Vibemeter captures extensive mood data, analyzing it alongside HR records (leave, performance, activities, recognition) remains manual and inefficient.

**Elara** bridges this gap with an AI-powered platform that correlates mood data with HR metrics to surface critical trends and identify at-risk individuals. Its adaptive chatbot initiates personalized, empathetic conversations, uncovering drivers of sentiment shifts using a dynamic Q&A bank and escalating concerns when needed.

Integrated into HR systems, Elara delivers daily sentiment intelligence, streamlines employee support, and fosters a connected, high-performing workforce through automation and AI.

### 1.2 Product Scope

This solution automated sentiment analysis by combining Vibemeter inputs with HR data to generate real-time insights, enhance engagement, and drive timely intervention.

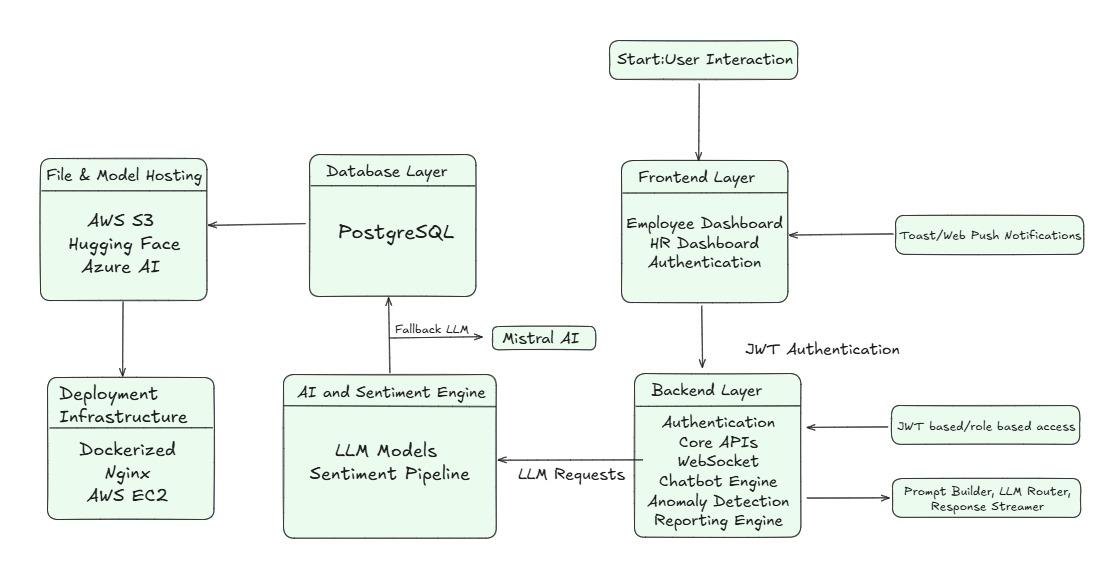
Core Features:

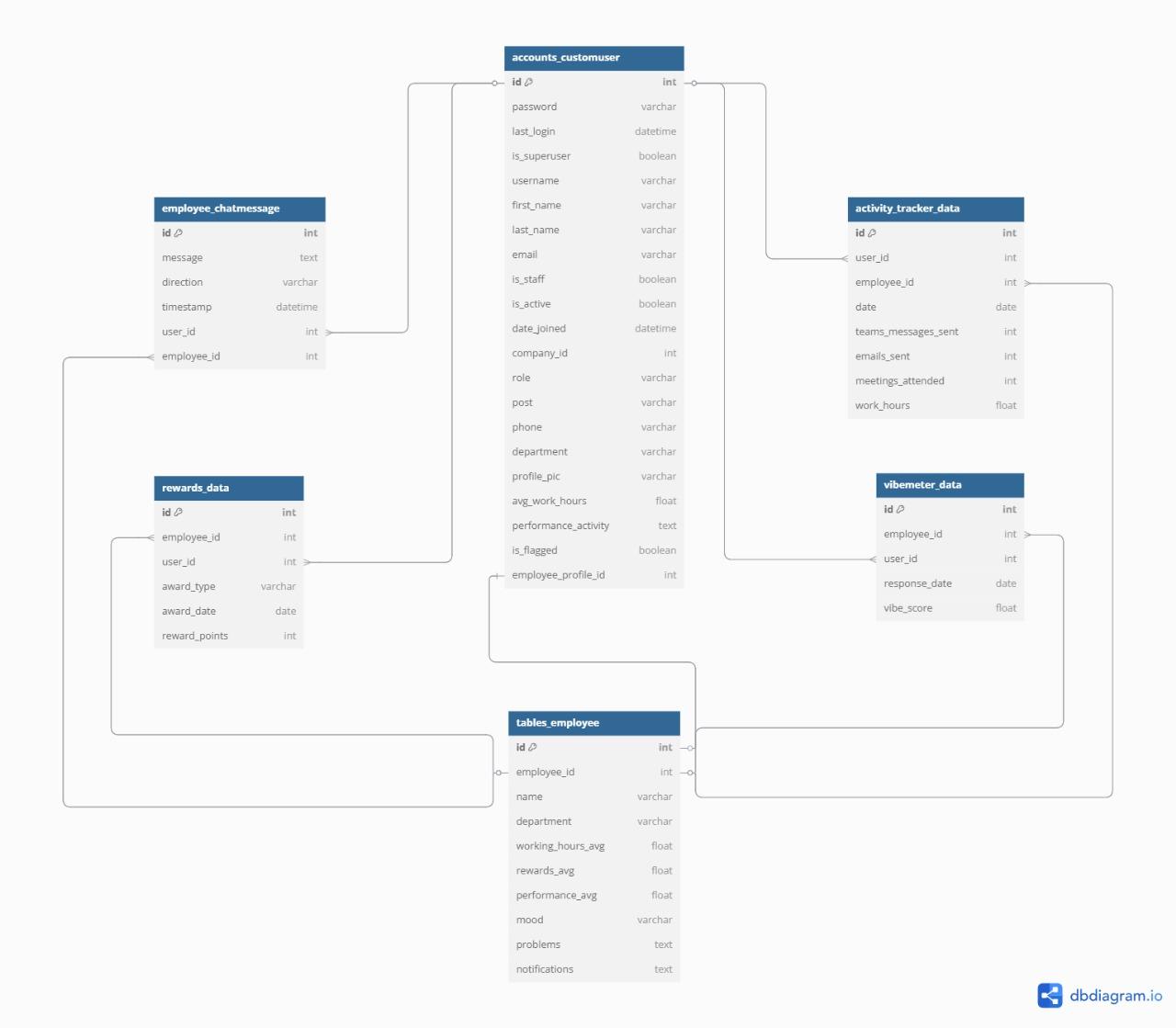
* Role-based dashboards for employees and HR
* Sentiment and engagement analytics
* *JWT*-secured access control
* AI chatbot with real-time, contextual conversations
* Daily automated reports for HR with action recommendations
* Integrated chat and notification system

Powered by *Django*, *React.js,* and *PostgreSQL*, and containerized for scalable deployment.

## **2. System Overview and Technical Blueprint**

### 2.1 System Architecture

****2.2 Entity Relation Diagram



**3. Intelligent Employee Assessment Framework**

### 3.1 Unified Data Pipeline and Feature Engineering

The data cleaning pipeline consolidates the six employee datasets into a unified master dataset for downstream analysis:

1. **Data Sources**: There are six datasets: *Activity metrics, Leave records, Onboarding feedback, Performance reviews, Reward history, Vibermeter responses*
2. **Key Processing Steps**
   * Temporal Decay Modeling: Applies recency-weighted decay factors to historical data (leave days, rewards, onboarding feedback)
   * Feature Engineering:
     + Aggregates activity metrics (sum/mean/median/std)
     + Encodes qualitative feedback (e.g., "Good" → numerical score)
     + Maps emotion zones to numerical values
   * Temporal Alignment: Standardizes date formats across all datasets.
3. **Output**
   * A 39-feature master dataset with:
     + Employee communication patterns
     + Workload intensity metrics
     + Historical performance trends
     + Decay-adjusted engagement scores

## **System Integration**

* Serves as input to Section 4.2 (*Employee Detection and Engagement Logic*) for anomaly detection and correlation analysis
* Provides foundation for daily report generation (Section 6)

### 3.2 Targeted Employee Detection and Engagement Logic

This module leverages ***Isolation Forest*** and ***SHAP explainability*** to identify potentially distressed employees based on behavioral, engagement, and performance signals. It facilitates proactive HR intervention through a transparent, tunable anomaly detection pipeline.

**Overview** The system is designed to:

* Detect anomalous behavior using *Isolation Forest*
* Interpret results using *SHAP values* to highlight stress-related features
* Calibrate detection sensitivity via a threshold factor
* Log flagged profiles in a centralized summary table (*af\_db*)

**Architecture Components**

* *Anomaly Detection*
  + Isolation Forest assigns an anomaly score to each employee.
  + More negative scores imply higher deviation from normal behavioral patterns.
* *Explainability Layer*
  + SHAP (SHapley Additive exPlanations) identifies key features influencing the anomaly score.
  + Emphasis is on *negative SHAP values*, signaling potential stressors (pushing towards the anomalous side of the detector).
* *Sensitivity Factor*
  + Configurable threshold to control detection depth:
    - Higher → broader flagging, early intervention
    - Lower → stricter filtering for severe cases
* *Output Summary (af\_db)*Stores structured insights for each flagged employee: *employee\_id, anomaly\_score, problems, other\_problems, average\_work\_hours, reward\_factor, performance\_rating, vibe\_factor.*

**Data Pipeline**

* **Input Features (n = 39):** Derived from the master dataset (Section 4.1)
* **Preprocessing:**
  + Encodes qualitative feedback
  + Imputes missing values (median)
  + Scales inputs via StandardScaler
* **Model Execution:**
  + Isolation Forest computes anomaly scores
  + SHAP TreeExplainer extracts top contributing features
  + Results logged in af\_db

### 3.3 Question Bank Development

To create a robust Question Bank for fine-tuning the conversational AI model, we conducted a systematic analysis of employee data and designed empathetic engagement strategies tailored to individual concerns. The process involved the following steps:

**Data Analysis**

Using the `*master\_df*` (32 employees), we manually analyzed key metrics such as anomaly scores, top 5 problems, and other concerns associated with each employee. Basic statistical techniques were applied to identify recurring patterns and prioritize issues that required immediate attention.

**Question Design**

For each identified problem, we crafted empathetic questions aimed at eliciting meaningful responses from employees. These questions were designed to:

* Address specific concerns directly.
* Encourage open communication.
* Maintain a professional yet supportive tone.

**Response Modeling**

To simulate realistic interactions, we developed seven possible employee responses for each key question. These responses reflected a range of emotional states and perspectives, ensuring the AI model could handle diverse conversational scenarios.

**Follow-Up Questions**

Based on the modeled responses, follow-up questions were created to deepen engagement and gather actionable insights. These follow-ups were designed to:

- Clarify ambiguous responses.

- Explore underlying causes of distress or satisfaction.

- Provide opportunities for employees to share additional feedback.

**Integration into Fine-Tuning**

The finalized Question Bank was passed to the AI model during fine-tuning. This ensured the chatbot could generate empathetic and contextually relevant conversations while addressing employee-specific concerns effectively.

## **4. Artificial Intelligence Models and Supporting Toolchains**

### 4.1 AI Model Architecture and Fine-Tuning Strategy

For our conversational bot solution, we implemented a sophisticated fine-tuning approach using **Mistral-7B-Instruct-v0.2** as our foundation model combined with **Quantized Low-Rank Adaptation (QLoRA)** techniques to create an empathetic HR assistant capable of nuanced employee engagement.

#### **Base Model Selection and Quantization**

We selected **Mistral-7B-Instruct** as our base model due to its strong performance in understanding context and generating human-like responses while being relatively efficient compared to larger models. To optimize computational resources, we implemented **8-bit quantization**, reducing memory footprint while preserving model quality. This enables deployment on standard enterprise hardware without specialized infrastructure.

#### **QLoRA Implementation**

Our QLoRA implementation specifically targets the attention mechanism components (*query, key, value, and output projections*) with the following parameters:

* **Rank (r):** 8
* **LoRA alpha:** 16
* **Dropout:** 0.05
* **Target modules:** "q\_proj", "k\_proj", "v\_proj", "o\_proj"

This approach allows us to fine-tune only **0.1–1%** of the model parameters while maintaining performance, significantly reducing training time and computational requirements.

**HR-Specific Prompt Engineering**

We developed a specialized **prompt formatting function** that structures conversations in a system-user-assistant format, enabling the model to maintain contextual awareness of its role in HR support scenarios. Each prompt contains:

* **System context:** HR policies, employee data insights
* **User input:** Employee queries or responses
* **Assistant response:** Empathetic guidance based on Vibemeter and related data

The training process **masks the labels** for system and user portions, ensuring the model learns to generate appropriate HR responses while maintaining the context of the provided employee data.

**Training Configuration**

The model was trained with carefully optimized hyperparameters:

* **Learning rate:** 2e-4
* **Training epochs:** 3
* **Gradient accumulation steps:** 2
* **Mixed precision training:** FP16

These parameters balance training efficiency with model quality, allowing the bot to quickly learn employee engagement patterns without overfitting to the training data.

The resulting fine-tuned model can effectively correlate **employee Vibemeter data** with other metrics (leave patterns, performance ratings, activity levels) to generate personalized, empathetic responses that encourage meaningful employee disclosure while maintaining professional boundaries.

### 4.2 Intelligent Conversation Orchestration

Our implementation builds upon the fine-tuned Mistral-7B-Instruct model through a sophisticated conversation orchestration system designed to deliver empathetic and personalized employee interactions. The system incorporates advanced reasoning techniques and dynamic context management to ensure meaningful engagement.

**Employee Context Generation**

The system employs a structured data extraction pipeline that transforms employee metrics into conversational contexts using:

* Severity-based problem categorization (severe, significant, moderate)
* Performance rating interpretation with contextual significance analysis
* Work hour pattern detection for overwork identification
* Social dynamics assessment through vibe factor analysis

These factors are synthesized into a comprehensive employee profile that guides the model's conversational approach while maintaining privacy and ethical boundaries.

**Tree-of-Thought Reasoning Framework**

The core innovation is our multi-perspective reasoning system that simulates a council of experts:

* Expert Council Simulation: The system implements a virtual roundtable of domain specialists including psychologists, workload analysts, social dynamics experts, and HR policy experts
* Deliberative Analysis: Each expert independently analyzes employee data before engaging in a simulated debate that surfaces competing hypotheses
* Consensus Building: The system synthesizes multiple perspectives into cohesive responses that reflect nuanced understanding of employee situations

This approach enables the model to consider multiple interpretations simultaneously, significantly enhancing its contextual awareness without revealing the underlying reasoning mechanism to employees.

**Dynamic Conversation Management**

To maintain consistent performance during extended interactions, we implemented:

* Context-aware token management that selectively prunes conversation history while preserving critical information
* Hierarchical importance weighting that prioritizes recent turns and system instructions
* Multi-stage response extraction with perspective validation to prevent role confusion
* Fallback response generation for edge cases to ensure conversation continuity

**Response Validation and Quality Control**

The system incorporates robust validation mechanisms that ensure responses maintain appropriate professional boundaries:

* Perspective contamination detection that identifies and filters responses adopting employee viewpoints
* Coherence assessment that ensures logical flow between conversation turns
* Length optimization balancing completeness with conversational efficiency

This comprehensive approach enables deep, empathetic engagement with employees while maintaining professional standards and generating valuable insights for the HR department.

### 4.3 HR Analytics Conversation Processing Pipeline

This report outlines our NLP pipeline for analyzing employee-chatbot conversations and generating structured HR reports.

## **System Architecture**

Our solution leverages a fine-tuned Mistral-7B model to process conversational data and employee metrics, producing actionable insights in a standardized format.

## **Key Technical Components**

## **Data Processing**

* Robust CSV and JSON parsing with error recovery
* AST-based deserialization for complex nested structures
* Format-agnostic conversation extraction supporting multiple schema types

## **Model Infrastructure**

* Hardware-aware deployment (CUDA/CPU detection)
* 4-bit quantization for memory optimization
* Dynamic precision selection based on available hardware

## **Inference Pipeline**

* Structured prompt engineering with format specification
* Parameter-optimized generation (temperature: 0.7, top-p: 0.9)
* Multi-strategy response extraction with fallback mechanisms

## **Output Processing**

* State-machine-based markdown parsing
* Fuzzy matching for section identification
* Hierarchical JSON structuring with normalization
* Empty section pruning for consistency

## **System Integration**

* Command-line interface with argument validation
* Flexible I/O handling for batch processing
* Structured JSON output for downstream system integration

## **Performance Optimization**

The system incorporates quantization techniques, efficient prompt construction, and dynamic hardware utilization to ensure optimal performance across different computing environments.

This pipeline provides a production-ready solution for automatically analyzing employee conversations and generating actionable HR reports with minimal human intervention.

## **5. Deployment, Integration, and Runtime Environment**

### 5.1 Step-by-Step Deployment Instructions

The backend is hosted on *Render* with auto-deploy enabled via *GitHub* integration. On push to main, Render pulls the latest code, installs dependencies, applies migrations, and restarts the app using *Gunicorn*. Django Channels, and PostgreSQL run as managed services. Render handles load balancing, rate limiting, and health checks, with built-in horizontal scaling and fallback mechanisms.

For auxiliary services and GPU workloads, an **AWS EC2 g5.2xlarge and g4dn instance** (Ubuntu AMI) is provisioned. Applications are containerized using a **multi-stage Dockerfile** and orchestrated via **Docker Compose**. Containers are exposed via secure ports and can be optionally routed through *NGINX* or *Traefik*.

### 5.2 Environment Configuration and Optimization

Environment variables are managed securely via .env files (Render dashboard or EC2 shell). *Gunicorn* handles WSGI/ASGI with high concurrency support. *PostgreSQL* pooling is handled via *psycopg2*. Static/media files are served via WhiteNoise or AWS S3.. Docker Compose enables isolated service environments and shared volumes. *Custom DRF* throttling ensures API rate limits. WebSocket performance is optimized through *Daphne/Uvicorn* for scalable bi-directional communication.

### 5.3 CI/CD Pipeline Overview

**Render CI/CD** triggers builds on every push. The pipeline installs dependencies, runs collectstatic, applies migrations, and restarts the service with rollback on failure. Blue/green deployments ensure zero downtime. Optionally, **GitHub Actions** may run tests (*pytest*), linters (*flake8, black*), security checks (*bandit, safety*), and coverage reports prior to deployment.

### **6. Daily Report Generation and Analysis**

### 6.1 Report Structure and Content

The system generates two distinct report types to support HR decision-making:

**Individual Employee Reports**

Each personalized report follows a comprehensive structure:

1. *Cover Page*: Corporate branding with "Deloitte ELARA" header
2. *Employee Summary*: Basic identification and organizational placement
3. *Working Hours Analysis*: Total hours, regular/overtime distribution, and attendance metrics
4. *Mood Trend Visualization*: Four-week emotional trajectory chart
5. *Key Insights*: Bullet-point summary of performance indicators and patterns
6. *Risk Assessment*: Stratified analysis of concerning signals with severity indicators
7. *Recommended Actions*: Tailored intervention suggestions based on identified patterns
8. *Conversation Summary*: Recent AI-employee interactions with key discussion points

**Collective Reports**

Aggregated daily summaries provide organization-wide insights:

1. Department-level mood distribution
2. Risk category distribution
3. Trending concerns across employee segments
4. Intervention effectiveness metrics

**Technical Implementation**

The report generation pipeline leverages ReportLab's PDF generation capabilities with a sophisticated multi-layered architecture:

* *Document Structure:* BaseDocTemplate with custom PageTemplates implementing dedicated frames for cover and content sections
* *Visual Styling:* Corporate design system implementation through ParagraphStyle inheritance and color standardization
* *Data Visualization:* Matplotlib integration for dynamic chart generation with trend visualization
* *Content Assembly:* Platypus flowable objects structured in sequential rendering pipeline
* *Data Integration:* Employee profile extraction combined with AI-summarized conversation insights
* *Header/Footer Management:* Canvas manipulation for consistent branding and pagination

Reports are generated on-demand via Django views, with PDF output stream as FileResponse objects. The modular implementation supports both individual and collective report generation through inheritance-based customization of the base generator class.

### **7 . Source Code Architecture and Explanation**

### 7.1 Backend Implementation

**The Elara backend** is a modular and scalable system built with Django and Django REST Framework. It powers Elara’s intelligent HR chatbot, handling authentication, sentiment analysis, HR data ingestion, reporting, and real-time communication. It integrates PostgreSQL for data storage and Django Channels for WebSocket support.

**7.1.1 System Design and Modules**

The system follows a **modular architecture**, with each component responsible for a specific functional domain:

* **Accounts Module**: Manages authentication and authorization using JWT tokens. Supports role-based access for HR personnel and employees.
* **Employee Module**: Encapsulates employee data, performance metrics, mood scores, and chat interactions.
* **Datasets Module**: Handles ingestion, validation, and merging of HR datasets, including activity logs and sentiment scores.
* **Chat Module**: Enables real-time, AI-driven communication via WebSockets, powered by Django Channels.

Each module is decoupled and adheres to single-responsibility principles, ensuring high cohesion and maintainability.

## **7.1.2 Key Features**

### **Authentication & Role Management**

* JWT-based secure login system.
* Role-based routing for employee and HR dashboards.
* Permissions enforced through Django’s built-in decorators and custom middleware.

### **HR Data Ingestion & Processing**

* Upload interface and API endpoint for six mandatory datasets: *Activity, Leave, Onboarding, Performance, Rewards, Vibemeter*
* Robust validation logic to ensure schema correctness.
* Consolidation logic merges inputs into a unified dataset for analysis and visualization.

### **Websocket-Driven Real-Time Communication**

* WebSocket-powered real-time chat via Django Channels.
* Conversations persist for sentiment tracking and auditability.
* Flagging mechanism identifies concerning patterns in employee behavior.

**Notification & Alert System**

* Sends scheduled notification to targeted employees for encouraging employee interaction, using AP Scheduler.
* Emails are sent along with the notifications if an employee has missed their notifications.
* HR is alerted along with email about an employee if they haven't chatted after a threshold of 3 notifications.

### **Automated Reporting**

* Performance summaries generated as downloadable PDFs using ReportLab.
* Visual trend plots rendered with Matplotlib.
* Reports include KPIs, mood trajectories, and managerial flags.

## 7.2 Frontend Architecture

This module defines the core user-facing features of the platform. Built with **React.js** and **Electron**, it supports **role-based access**, **interactive dashboards**, **real-time communication**, and **intelligent assistance** for both employees and HR teams, across web and desktop environments.

### **7.2.1 User Authentication and Role-Based Authorization**

Secure JWT-based login enables differentiated access:

* Employees are routed to a personal dashboard reflecting individual well-being and activity.
* HR personnel access an administrative interface with oversight tools and employee insights.

### **7.2.2 Interactive Dashboards and Visual Analytics**

#### **Employee Dashboard**

* *Mood History* : Time-series visualization of emotional trends.
* *Activity Log* : Day wise Work hours
* *Calendar & Task View* : Upcoming workload and deadlines.
* *Notifications* : Subtly prompts employees to engage with chatbot.
* *Chatbot Integration* : Access to Elara for personalized support.

#### **HR Dashboard**

* *Employee Focus Panel:* View flagged employees for further attention.
* *Employee Reports:* Open and download detailed PDF reports.
* *Mood Analytics:* Average mood history graph across teams.
* *Work Hours Analysis:* Graph showing average working hours department wise.
* *Performance vs Rewards:* Bar chart visualizing performance and corresponding rewards across teams.

### **7.2.3 Integrated Chatbot Assistant**

**Elara**, the AI-driven assistant, provides:

* Real-time chat via WebSockets.
* Voice-enabled interaction.
* Context-aware guidance on mood and engagement.

### **7.2.4 Messaging and Notifications**

* Secure internal messaging between HR and employees.
* A sleek notification icon in the navbar subtly prompts employees to engage with support via chat

### **7.2.5 Profile Management**

Users can view and update personal information such as name, contact details, and designation through the profile module.

### 7.3 Data Schema and Integration

The database schema is architected using*PostgreSQL* and is centered around the *accounts\_customuser* table, which serves as the relational core. All other tables are either directly or indirectly connected to this central entity, ensuring referential integrity and streamlined data access across modules.

Database Structure Overview:

* *accounts\_customuser*Serves as the principal user table, storing core authentication and role-based data (e.g., Employee, HR). It forms the basis for user identity across the system.
* *employee\_chatmessage*Captures historical chat interactions with the Elara assistant. Each record references a user via user\_id, allowing conversational analytics and engagement tracking.
* *activity\_tracker\_data*Logs day-to-day employee productivity metrics—including message counts, meeting participation, and work hours—via a foreign key employee\_id relationship to accounts\_customuser.
* *rewards\_data*Stores information related to employee recognition, such as reward points and award justifications, with foreign key linkage to employee records.
* *vibemeter\_data*Monitors and stores periodic mood scores for each employee. The data is associated with the user through employee\_id, supporting sentiment trend analysis.
* *tables\_employee*Aggregates key performance indicators, mood scores, and flagged issues per employee who is flagged. This table underpins the user dashboard and facilitates managerial insights.

This relational model ensures modular scalability, robust data normalization, and efficient query performance across various system components.

## **9. Requirements**

1. Performance Requirements

* *Scalability*: Supports concurrent employees, ensuring smooth operation during peak usage.
* *Latency*: Chatbot response time does not exceed a few seconds to maintain conversational flow.
* *Throughput*: Handles simultaneous conversations without degradation.

1. Security Requirements

* *Data Privacy*: Encrypt data in transit (TLS) and at rest (AES-256).
* *Access Control*: Implement role-based access control (RBAC) for HR and employee data.
* *Authentication*: Use secure JWT-based authentication for all users.
* *Audit Logs*: Maintain detailed logs of interactions for compliance.
* *Anomaly Detection*: Monitor system usage for unusual patterns.

1. Safety Requirements

* *Ethical Safeguards*: Validate chatbot responses against pre-defined ethical guidelines to avoid harmful outputs.
* *Escalation Mechanism*: Automatically escalate flagged high-risk cases.

1. Software Quality Attributes

* *Reliability*: Ensure consistent performance under varying workloads, including peak hours.
* *Maintainability*: Modular architecture should allow updates without disrupting the entire system.
* *Usability*: Provide intuitive interfaces for employees and HR personnel with clear navigation and prompts.
* *Portability*: Ensure compatibility across Windows, Mac, and cloud platforms like AWS or Azure.

1. Additional Requirements

* *Customization*: Allow HR teams to adjust thresholds for anomaly detection and escalation.
* *Monitoring and Alerts*: Implement real-time monitoring of system health
* *Compliance*: Adhere to GDPR and relevant data protection regulations.
* *Integration Support*: Facilitate integration with third-party tools like Slac

ANNEXURE

* [**https://leena.ai/blog/chatbots-for-employee-engagement/**](https://leena.ai/blog/chatbots-for-employee-engagement/)
* [**https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2**](https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2)
* [**https://www.kaggle.com/code/younesselbrag/fine-tuning-mistral-7b-using-qlora**](https://www.kaggle.com/code/younesselbrag/fine-tuning-mistral-7b-using-qlora)
* [**https://huggingface.co/docs/transformers/perf\_infer\_gpu\_one#flashattention**](https://huggingface.co/docs/transformers/perf_infer_gpu_one#flashattention)
* [**https://medium.com/towards-data-science/explaining-anomalies-with-isolation-forest-and-shap-0d5d1224b918**](https://medium.com/towards-data-science/explaining-anomalies-with-isolation-forest-and-shap-0d5d1224b918)
* [**https://arxiv.org/abs/2305.10601**](https://arxiv.org/abs/2305.10601)
* [**https://render.com/docs/github**](https://render.com/docs/github)
* [**https://aws.amazon.com/ec2/instance-types/g4/**](https://aws.amazon.com/ec2/instance-types/g4/)
* [**https://docs.docker.com/build/building/multi-stage/**](https://docs.docker.com/build/building/multi-stage/)
* [**https://adityamangal98.medium.com/wsgi-vs-asgi-a-deep-dive-into-gunicorn-uvicorn-and-more-90747dfb2fc9**](https://adityamangal98.medium.com/wsgi-vs-asgi-a-deep-dive-into-gunicorn-uvicorn-and-more-90747dfb2fc9)
* [**https://www.django-rest-framework.org/api-guide/throttling/**](https://www.django-rest-framework.org/api-guide/throttling/)
* [**https://github.com/features/actions**](https://github.com/features/actions)
* [**https://docs.reportlab.com/reportlab/userguide/ch5\_platypus/**](https://docs.reportlab.com/reportlab/userguide/ch5_platypus/)
* [**https://channels.readthedocs.io/en/latest/**](https://channels.readthedocs.io/en/latest/)
* [**https://docs.djangoproject.com/en/5.2/**](https://docs.djangoproject.com/en/5.2/)
* [**https://docs.reportlab.com/**](https://docs.reportlab.com/)
* [**https://www.electronjs.org/docs/latest**](https://www.electronjs.org/docs/latest)
* [**https://gunicorn.org/#deployment**](https://gunicorn.org/#deployment)
* [**https://tailwindcss.com/docs/installation/using-vite**](https://tailwindcss.com/docs/installation/using-vite)
* [**https://docs.pytest.org/en/stable/**](https://docs.pytest.org/en/stable/)