Vibemeter: Employee Engagement Platform

Vibemeter is a comprehensive employee engagement platform that collects, analyzes, and visualizes employee sentiment data through interactive chatbot conversations and surveys. It empowers HR teams to monitor engagement trends, track employee mood, and take timely action when needed.

Features

- **Smart Chatbot:**
  AI-powered conversation analysis to understand employee sentiment.
- **HR Dashboard:**
  Comprehensive view of organization-wide engagement metrics.
- **Analytics:**
  Data visualization of employee engagement trends.
- **Sentiment Tracking:**
  Monitors employee mood and satisfaction levels.
- **Escalation System:**
  Alerts HR teams when employees need additional support.

Backend Setup

Prerequisites

- Python 3.10 or higher
- PostgreSQL
- Git

Installation

1. **Clone the Repository:**

   ```
   git clone https://github.com/LAL-BAHADUR-SHASTRI/Opensoft-2025
   cd vibemeter
   ```

2. **Create and Activate a Virtual Environment:**

   ```
   python -m venv venv
   source venv/bin/activate  # On macOS/Linux
   venv\Scripts\activate     # On Windows
   ```

3. **Install Dependencies:**

   ```
   pip install -r requirements.txt
   ```

Database Setup

Local Development Database

1. **Create a PostgreSQL Database.**
2. **Configure your environment variables** in a `.env` file in the backend directory with your database credentials and other settings.

Docker Database Setup

If using Docker, run the following command to start your PostgreSQL container along with the backend:

```
docker-compose up -d db
```

Database Initialization

Run the initialization script to create all tables and add default users:

- Creates all database tables based on SQLAlchemy models.
- Adds an HR user: **username:** `hruser`, **password:** `hruser`.
- Adds a sample employee: **username:** `emp0048`, **password:** `emp0048`.

```
python init_db.py
```

Reset Database

Warning: This deletes all data.
To completely reset the database, run the reset script provided.

```
python reset_db.py
```

Running the Application

Backend

- **Development Mode:**
  Run your application in development mode with hot reloading.
- **Production Mode:**
  The application runs with 4 worker processes to handle simultaneous connections.

Frontend Setup

Prerequisites

- Node.js 20.x or later
- npm or yarn

Installation

1. Navigate to the `Frontend` directory.
2. Install dependencies:

   ```
   npm install --force
   ```

Running the Frontend

- **Development Mode:**
  Start the development server:

  ```
  npm run dev
  ```

- **Production Build:**
  Build the frontend for production:

  ```
  npm run build
  ```

Docker Deployment

Both the frontend and backend can be deployed using Docker. Use the provided `docker-compose.yml` file to start:

- PostgreSQL database
- Backend API
- Frontend application

```
docker-compose up --build
```

API Endpoints

Authentication

- **POST** `/token`
  Login to get access token.

- **POST** `/logout`
  Logout and clear authentication cookie.
- **GET** `/users/me`
  Retrieve current user information.

Data Management

- **POST** `/upload-csv/`
  Upload CSV files (HR role required).
- **GET** `/jobs/{job_id}`
  Check status of a background job.
- **GET** `/jobs/`
  Get status of all background jobs.
- **POST** `/users/create-async/`
  Create user accounts from employee data (async).
- **GET** `/tables/`
  Get list of all tables and record counts.
- **GET** `/data/{table_name}`
  Get data from a specific table with pagination.
- **GET** `/employee/data`
  Get data for the logged-in employee.

Reporting

- **GET** `/report/collective`
  Generate a collective organizational report.
- **POST** `/report/employee`
  Generate an individual employee report.
- **POST** `/report/selective`
  Generate a report for selected employees.

Chat

- **POST** `/start_chat`
  Start a new chatbot session.
- **POST** `/chat`
  Send a message and get the next question or final analysis.
- **GET** `/chathistory`
  Get chat history for an employee.
- **GET** `/chatdates`
  Get dates when an employee had chatbot conversations.

HR Management

- **POST** `/hr/resolve-escalation/{employee_id}`
  Clear HR escalation flag for a specific employee.
- **GET** `/hr/employees/need-attention`
  Get a list of employees requiring HR attention.

CSV Data Import

The system supports importing data from CSV files.

1. **Log in as an HR user.**
2. Use the **POST** `/upload-csv/` endpoint to upload CSV files.
3. After upload, user accounts are automatically created for all employee IDs.

Supported CSV files:

- `activity_tracker_dataset.csv`
- `leave_dataset.csv`
- `onboarding_dataset.csv`
- `performance_dataset.csv`
- `rewards_dataset.csv`
- `vibemeter_dataset.csv`

Sample CSV files are available in the `data` directory.

Environment Variables

Create a `.env` file in the **Backend** directory with your configuration variables. For the **Frontend**, create a `.env` file in the Frontend directory with the required settings.

Chatbot System

The application includes an AI-powered chatbot that:

- Conducts sentiment analysis on employee responses.
- Identifies key themes and concerns.
- Provides HR with actionable insights.
- Flags employees who may need additional support.
- Maintains complete conversation history for reference.

Troubleshooting

Database Connection Issues

- Ensure PostgreSQL is running.
- Verify database credentials in `.env`.
- If using Docker, ensure the `db` service is running.

CSV Upload Issues

- Ensure the CSV headers match the expected format.
- Verify that the HR user has the correct permissions.
- Check the logs for detailed error messages.

License