Office Lunch Menu Management System

Description

The Office Lunch Menu Management System is a web application designed to streamline the process of managing daily lunch options in an office environment. Admins can easily add and manage daily lunch menus, while employees can view these menus and select their lunch preferences. This system aims to simplify the lunch ordering process and ensure that all employees' choices are recorded efficiently.

Technologies

- Backend: Node.js with Express.js
- Frontend: React.js
- Database: PostgreSQL

Features for MVP (Minimum Viable Product)

Admin Interface:

- Add Daily Menu Options: Admins can add new lunch options for specific dates.
- View Employee Choices: Admins can view which employees have chosen which lunch options.

Employee Interface:

- View Daily Menu: Employees can see the lunch options available for the current day.
- Select Lunch Choice: Employees can select their preferred lunch option from the daily menu.

Submission Instructions for Office Lunch Menu Management System

To be considered for the internship position, please follow these submission instructions carefully:

Create a GitHub Repository:

1. Create a new GitHub repository for your project.
2. Name your repository appropriately (e.g., `office-lunch-menu-management`).

Implement the Project:

Backend (Node.js with Express.js):

1. Set up a Node.js project with Express.js.
2. Implement routes and controllers for menu and user management.
3. Use PostgreSQL for the database.

Frontend (React.js):

1. Set up a React.js project.
2. Implement components for viewing the menu and selecting lunch choices.
3. Ensure both backend and frontend are properly connected.

Follow Commit Guidelines:

1. Write clear, concise commit messages.
2. Use conventional commits (e.g., `feat:`, `fix:`, `chore:`) to describe changes.
3. Make small, incremental commits instead of large, monolithic ones.

Use a Branching Strategy:

1. Use `master` for stable code.
2. Create feature branches (e.g., `feature/add-menu`) for new features.
3. Merge feature branches into `master` with pull requests.

Example of Good Commit Messages:

- `feat: add menu creation endpoint`
- `fix: correct user authentication logic`
- `chore: update README with setup instructions`

Follow Git Guidelines:

1. Follow conventional commits for commit messages.
2. Use feature branches for new features.
3. Merge feature branches into `master` with pull requests.

Update the README File:

Ensure the README file in the root directory of your repository includes:

- Technologies used
- Features
- Database schema
- Setup instructions for both backend and frontend
- Instructions on how to run the projects