Capstone Project 1

Management-Reminder app

August 2022

Overview

This app will model any sort of management scenario where there's an institution that has someone in charge(e.g. a manager) and people who are under that person's charge(e.g. staff). The institution has certain tasks that need to get done and the members of the institution may have certain responsibilities that need to be performed. The basic goal of the app is to perform 2 functions: 1. To send reminders to users about upcoming tasks. 2. To provide at least a basic tracking of whether those tasks were done or not. The app will have to keep track of time to allow for these functions. The app will have two classes of users: regular and admin. Admins will have full CRUD support for basically every piece of data in the app. Regular users will have limited crud for their own data.

Basic Data Classes/SQL Tables

- **TASKS:** They would include a boolean value of 'completed' for if they're completed or not, a reference to the user they are assigned to(for a task it could be many, for a responsibility it's always one to a specific user), a value for when they should be completed by, a value for when they can first be completed(if applicable), a value for when the task/responsibility was first created, a value for when the job was decommissioned, , a value for the admin that created the job, and possibly a value for priority level of the job.

- **USERS:** Could be regular or admin. Attributes would include a first and last name, username, password, email, phone number, the date they were first added to the institution, the date they left(if they did), a reference to the tasks they have, a reference to the responsibilities they have, a reference to reminders that apply to them

- **ASSIGNMENTS:** A class that would connect users to jobs. It would have a user that's being reminded, a task or responsibility that it's reminding about, a message, a option for when it first reminds. The reminder would involve the external api that would send sms texts as the reminder.
- **ADMINS:** Could be a subclass that inherits from the user class but adds on some new attributes or methods. The simple alternative would be to have an 'is_admin" attribute on every user.

Stretch Goal Classes(for a later version of the app):

- **Organization:** Possible super class for the entire app. It could have all of the jobs, users and reminders whether they are active or not.
- **ROLES:** A class that would bundle a pattern of tasks and/or responsibilities. Many people in an organization might have nearly identical tasks or responsibilities, so a role would be a bundle of jobs, and would have either a special init method or a convenience method that takes a user and automatically adds all of the jobs to that user.

External API

An sms api that would provide a host phone #, and would provide a service for sending automated texts to phone #'s provided.

Rough draft of the User Flow:

- There would be a signup page that by default creates the new user as a non-admin user. The status of that could be changed by an actual admin directly in the database.
- Once a task is created, the admin could assign the task to the user(s) that the admin wants.

- Once a user-task connection has been made, the admin can set up a reminder. Alternatively, there could be preset functionality that automatically instantiates a reminder once a task or responsibility has been assigned to a user.
- The user could sign in to see a list of his/her tasks/responsibilities and reminders that are coming up.
- An admin could access a list of the tasks, and update or delete them.
- An admin could access a list of users, an individual user, or a user's responsibilities and edit or delete any of that information.

Potential Issues:

- Figuring out how to keep track of time
- Figuring out how best to tether jobs/reminders to specific times
- Limiting the amount of api calls, both to my server and to the external api. (For example, instead of sending 10 api calls to the external server to remind 10 users about the same task, it would be better to send 1 call to remind 10 users).
- Limiting the amount of db querying
- Formatting (because it's always an issue)

Stretch Goals:

- Allowing users to modify reminders that are required

- Allowing users to create their own reminders, and then have full crud over them
- Allowing the user to check off for themself, that they've completed a certain job
- Defining different patterns of trust that would determine when a job is considered complete.
- Allowing the admin to make more options for a job than just complete or incomplete
- Having an analytics page that admin could see