https://github.com/CabrrHernan/4353-Group-17

Initial Thoughts

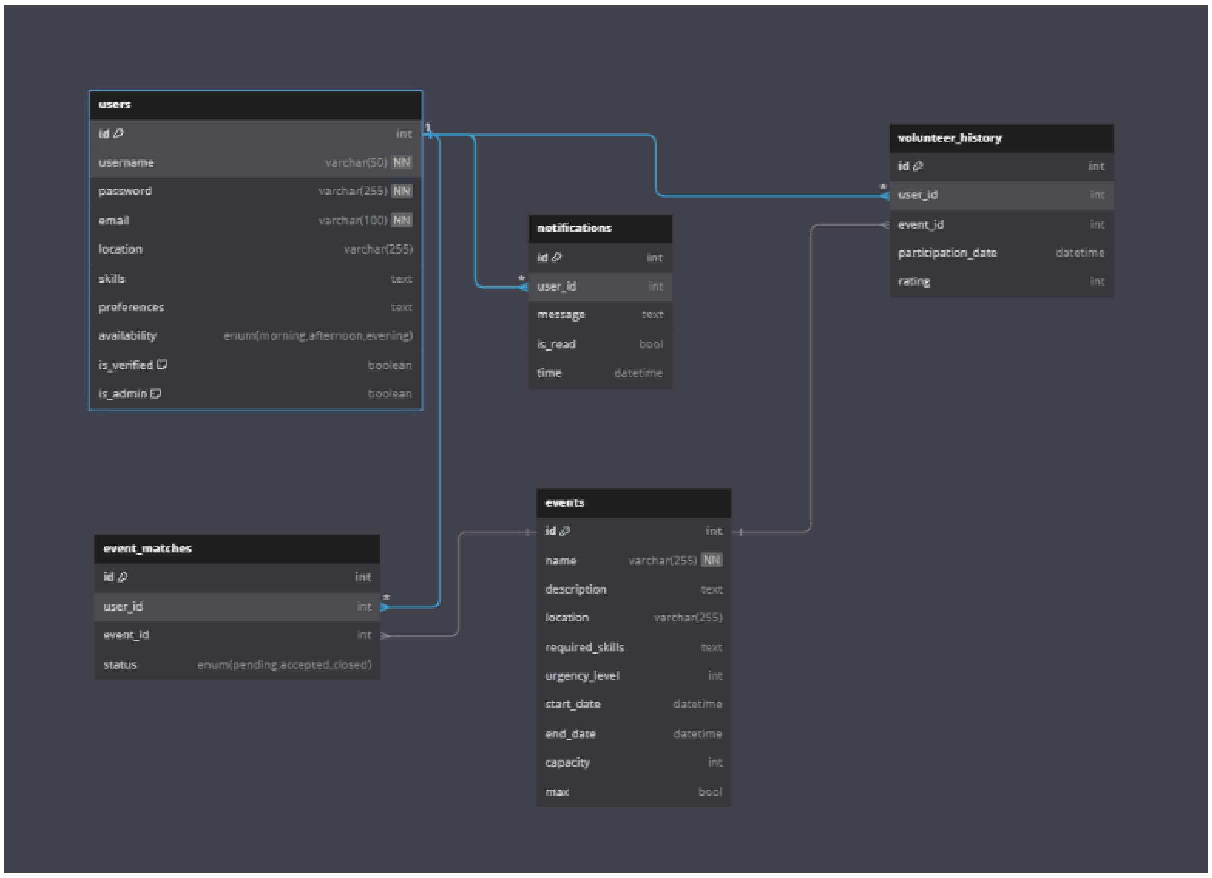
- Consider user experience: How will users (volunteers and administrators) interact with the application?
    - Both Volunteers and Administrators will have accounts associated with the website
    - Volunteers will have a more basic user page letting them sign up for events
    - Administrators will have same ability but with additional tools to create,edit, delete events
- Identify the key functionalities: What are the essential features the application must have?
    - User login, User Registration,User Profile Management,Event Management,Volunteer Matching,Notification System,Volunteer History
- Technology stack: What technologies might you use for front-end, back-end, database, and other components?
    - FrontEnd: React, HTML/CSS
    - Backend: Flask
    - Database: PostgreSQl
    - Server: Microsoft Azure

Development Methodology

- Explain why you would choose a particular development methodology (e.g., Agile, Waterfall, DevOps).
    - We are going to use the Agile methodology. Our limited experience in software development and the time constraints hinder our ability to create a solid and in depth blueprint. Methods such as the waterfall method, where planning is a crucial element, are not ideal for our project. The Agile method allows for greater flexibility, so our plan can evolve as our project takes shape.
- Discuss how this methodology will help manage the project effectively.
    - Since the requirements and constraints of the project are pretty straightforward, the Agile method affords us the opportunity to start building our project with just the basic outline provided. This allows for us to begin construction as soon as possible so that we can gain a better understanding of the absolute necessities and later hone in on the minute details. Since communication is a crucial element of this method, we have set aside time every week to meet and discuss our thoughts and findings.

High-Level Design / Architecture

- Create a diagram to illustrate the overall structure of your application.



-
- Identify the main components (e.g., front-end, back-end, database).
  - FrontEnd: React, HTML/CSS
  - Backend: Flask
  - Database: PostgreSQl
  - Server: Microsoft Azure
- Describe how these components will interact with each other.
  - Front-End (React, HTML/CSS):
    - the user interface of the application. React will handle the UI elements, while HTML/CSS manage the structure and styling
    - React will make HTTP requests to the Flask to update or delete data when someone interacts with the UI
  - Back-End (Flask):
    - Flask is the web framework written in Python that handles the server-side logic of the application.
    - It processes incoming requests from the front-end, performs necessary business logic, and interacts with the PostgreSQL database to retrieve or modify data. Flask then sends the

        processed data back to the front-end, typically in JSON format, which React uses to update the UI.
    - Database (PostgreSQL):
        - PostgreSQL is the database to store the application's data. The Flask back-end communicates with PostgreSQL to perform CRUD (Create, Read, Update, Delete) operations.
        - Flask uses an ORM or direct SQL queries to interact with the database.
    - Server (Microsoft Azure):
        - Microsoft Azure hosts the application, providing the infrastructure for the front-end, back-end, and database.
        - Azure runs the Flask application and manages the database server where PostgreSQL is hosted.

- Mention any third-party services or APIs you plan to integrate.
    - We will use an API to send email verifications to the user

Member Name	Contributions	Notes
Edward Khalil	Thoughts and ideas	
Hernan Cabrera	Made the groupchat, Created the Github, typed the google doc, thoughts and ideas	
Richard Ayala	Created the diagram, thoughts and ideas	
Charlie Oblock	Minor google doc edits, scheduled meeting, thoughts and ideas	