## Assignment 1: Initial Thoughts and Design

## Initial Thoughts

### User Experience

* **Volunteers**: Access a responsive web app via React and Material UI. They register (RegisterPage.tsx) with email verification, log in (LoginPage.tsx), and complete profiles (ProfilePage.tsx) with location (via Google Maps), skills (e.g., medical, teaching), preferences (e.g., outdoor events), and availability (calendar input). They view matched events (MatchPage.tsx), track participation (HistoryPage.tsx), and receive notifications (NotificationPage.tsx) via email/SMS.
* **Administrators**: Log in to create events (EventsPage.tsx), specifying location, skills, and urgency. They view volunteer profiles, assign matches, and generate PDF/CSV reports (ReportPage.tsx).
* **Usability**: Material UI ensures a consistent, mobile-friendly design. Error messages guide users (e.g., invalid login), and a navbar simplifies navigation.

### Key Functionalities

* **Login/Registration**: JWT-based authentication with SendGrid email verification (auth.py).
* **User Profile Management**: Stores location (coordinates), skills, preferences, and availability (profile.py, ProfilePage.tsx).
* **Event Management**: Admins create events with location, required skills, and urgency (high/medium/low) (events.py).
* **Volunteer Matching**: Scores volunteers based on location proximity (Google Maps API), skill match, availability, and event urgency (match.py, MatchPage.tsx).
* **Notification System**: Sends event assignments and reminders via SendGrid (NotificationPage.tsx).
* **Volunteer History**: Tracks participation and performance (history.py, HistoryPage.tsx).
* **Reporting**: Exports volunteer/event analytics as PDF/CSV (report.py, ReportPage.tsx).

### Technology Stack

* **Frontend**: React (scalable UI), TypeScript (type safety), Material UI (responsive design).
* **Backend**: FastAPI (high-performance APIs), Python, Pydantic (data validation).
* **Database**: PostgreSQL via Supabase (real-time, managed hosting).
* **Testing**: pytest, pytest-cov (unit testing, coverage).
* **Version Control**: Git, GitHub (collaboration).
* **Why Chosen**: FastAPI’s async capabilities suit API needs; Supabase simplifies database and auth; React/Material UI ensures modern UI.

## Development Methodology

* **Methodology**: Agile.
* **Why**: Agile supports iterative development, enabling client feedback and feature prioritization. Two-week sprints deliver modules (e.g., auth, matching). GitHub tracks progress via branches/commits.
* **Management**: GitHub Issues for tasks, daily stand-ups for coordination.

## High-Level Design / Architecture

### Diagram

A diagram of a volunteer management application

AI-generated content may be incorrect.

### Components

* **Frontend (React, TypeScript, Material UI)**:
  + HomePage.tsx: App overview.
  + RegisterPage.tsx, LoginPage.tsx: Authentication.
  + ProfilePage.tsx: Profile management.
  + MatchPage.tsx: Matched events.
  + EventsPage.tsx: Event management (admin).
  + HistoryPage.tsx: Participation history.
  + NotificationPage.tsx: Notifications.
  + ReportPage.tsx: Analytics dashboard.
* **Backend (FastAPI, Python, Pydantic)**:
  + auth.py: JWT authentication.
  + events.py: Event CRUD.
  + history.py: Participation tracking.
  + match.py: Matching algorithm (distance, skills, urgency).
  + profile.py: Profile management.
  + report.py: Report generation.
  + supabase\_client.py: Supabase connection.
* **Database (Supabase PostgreSQL)**:
  + Tables: users (profiles), events (details), matches (assignments), history (records), notifications (messages).

### Component Interactions

* **Frontend-to-Backend**: React calls FastAPI endpoints (e.g., POST /register, GET /match). E.g., MatchPage.tsx requests /match to display events.
* **Backend-to-Database**: FastAPI uses supabase\_client.py to query Supabase (e.g., SELECT \* FROM events WHERE urgency='high').
* **External Services**: Google Maps API calculates volunteer-event distances for match.py. SendGrid sends emails for auth.py and notifications.
* **Real-Time**: Supabase WebSockets push updates to NotificationPage.tsx.

### Third-Party Services

* **Supabase**: Database, authentication, real-time.
* **SendGrid**: Email verification, notifications.
* **Google Maps API**: Location-based matching.

### Group Contributions

| Group Member Name | GitHub Username | Contribution |
| --- | --- | --- |
| Ryan Sharifi | ryan4sharifi | GitHub, Supabase, backend and Frontend setup for Login and Registration. |
| Fidel Garcia | FF-FG-FF |  |
| Nhat Tran | NhatTran222 | google maps api, front-end |
| Truc Le | FrankinstyLe | Profile management (profile.py, ProfilePage.tsx), pytest tests |

**GitHub**: https://github.com/ryan4sharifi/Group-9-App