## Project Documentation: XYZ Non-Profit Event Management System

## 1. Project Overview

Client: XYZ Non-Profit Organization Team Members:

* Jonathan - Team Manager
* Juan - Back-end Developer
* Grayson - Front-end Developer
* Harrison - Front-end Developer
* Blake - Back-end Developer

## 2. Scope & Features

The system is a web-based application designed to help users register for events, view event details, and receive notifications about upcoming events. It includes the following core features:

### Key Functionalities:

* User Authentication: Users can sign up, log in, and manage their accounts securely.
* Event Management: Admins can create, update, and remove events; users can register for events.
* Notifications: Users receive automated notifications/reminders regarding upcoming events.

## 3. Purpose & Goals

This system aims to:

* Simplify the event registration process for XYZ Non-Profit Organization.
* Ensure users stay informed about event details effortlessly.
* Assist the organization in managing event logistics more efficiently.

## 4. Technology Stack

### Frontend:

* HTML, CSS, JavaScript
* Frameworks: React or Vue.js

### Backend:

* Python (Flask/Django)
* Database: MySQL or PostgreSQL

### Communication Channels:

We will utilize:

* Discord for real-time discussions
* Phone calls when needed for urgent matters

## 5. System Design & Code Overview

### User Class:

Handles user information and authentication.

python

class User:

def \_\_init\_\_(self, username, email):

self.username = username

self.email = email

### Event Class:

Stores event details and allows user registration.

python

class Event:

def \_\_init\_\_(self, name, date, location):

self.name = name

self.date = date

self.location = location

self.attendees = []

def register\_user(self, user):

self.attendees.append(user)

return f"{user.username} registered for {self.name}."

### Notification Class:

Handles reminders for upcoming events.

python

class Notification:

def send\_reminder(self, user, event):

return f"Reminder sent to {user.username} for event: {event.name}."

### GUI-Based Event Registration:

Using Tkinter, users can register via a simple interface.

python

import tkinter as tk

from tkinter import messagebox

class EventApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("XYZ Non-Profit Event Manager")

self.events = {

"Fundraiser Gala": Event("Fundraiser Gala", "2025-05-10", "Community Center"),

"Volunteer Meetup": Event("Volunteer Meetup", "2025-06-15", "Local Park"),

"Awareness Drive": Event("Awareness Drive", "2025-07-20", "City Hall")

}

tk.Label(root, text="Enter Your Name:").grid(row=0, column=0)

self.username\_entry = tk.Entry(root)

self.username\_entry.grid(row=0, column=1)

tk.Label(root, text="Enter Your Email:").grid(row=1, column=0)

self.email\_entry = tk.Entry(root)

self.email\_entry.grid(row=1, column=1)

tk.Label(root, text="Select Event:").grid(row=2, column=0)

self.event\_var = tk.StringVar(root)

self.event\_var.set("Fundraiser Gala")

tk.OptionMenu(root, self.event\_var, \*self.events.keys()).grid(row=2, column=1)

tk.Button(root, text="Register", command=self.register\_event).grid(row=3, column=0, columnspan=2)

def register\_event(self):

username = self.username\_entry.get()

email = self.email\_entry.get()

event\_name = self.event\_var.get()

if username and email:

user = User(username, email)

event = self.events[event\_name]

messagebox.showinfo("Registration", event.register\_user(user))

else:

messagebox.showwarning("Input Error", "Please enter both name and email.")

root = tk.Tk()

app = EventApp(root)

root.mainloop()

## 6. Next Steps & Improvements

* Implement role-based access control for event organizers.
* Enhance notifications with email integration.
* Extend the system to support multi-day event scheduling.