Assignment2: FRONT-End Development

**--Group7:** **Sulton Abiola Adedokun, Sabrina Yang, Kun Wang**

To implement the front-end of the non-profit volunteer management web application, we will first design the structure and behavior of the system in UML, and then implement these components using HTML and CSS.

# 1. The structure and behavior of the system

## 1.1 Use Case Diagram

The use case diagram shows the functional requirements of a system and the interactions between users (volunteers and administrators) and the system.

Step1: Identify roles: Identify different user roles in the system, such as volunteers and administrators.

Step2: Identify use cases: Identify use cases that each role can perform, such as login, registration, managing events, and so on.

Step3: Draw diagrams: Use UML to draw use case diagrams that represent the relationship between user roles and use cases.

The code in PlantUML as following:

@startuml

left to right direction

actor Volunteer

actor Administrator

actor Visitor

Volunteer --> (Login)

Volunteer --> (Register)

Volunteer --> (Update Profile)

Volunteer --> (View Events)

Volunteer --> (Sign Up for Events)

Volunteer --> (Receive Notifications)

Volunteer --> (View Participation History)

Administrator --> (Login)

Administrator --> (Register)

Administrator --> (Manage Events)

Administrator --> (Match Volunteers)

Administrator --> (Send Notifications)

Administrator --> (View participation History)

Visitor --> (View Events)

@enduml

The output we get shows in Figure 1.1.

A diagram of a person's work flow

Description automatically generated

Figure1.1: Use Case Diagram

## 1.2 Class Diagram

The class diagram shows the structure of the components and their relationships.

Step1: Identification classes: Identify the main classes in the system, such as users, events, notifications, etc.

Step2: Define properties and methods: Define properties and methods for each class.

Step3: Draw diagrams: Use UML to draw class diagrams that represent classes and their properties, methods, and relationships between classes.

The code in PlantUML as following:

@startuml

class User {

+String Email

+String Password

+String FullName

+String Address1

+String Address2

+String City

+String State

+String Zipcode

+List<String> Skills

+String Preferences

+List<Date> Availability

+void login()

+void register()

+void updateProfile()

+void viewEvents()

+void signUpForEvent(Event event)

+void receiveNotification(Notification notification)

+List<ParticipationHistory> viewParticipationHistory()

}

class Volunteer extends User {

+void updateAvailability(List<Date> availability)

+void updateSkills(List<String> skills)

}

class Administrator extends User {

+void manageEvents(Event event)

+void matchVolunteers(Event event)

+void sendNotifications(List<Notification> notifications)

}

class Event {

+String EventName

+String EventDescription

+String Location

+List<String> RequiredSkills

+String Urgency

+Date EventDate

+void CreateEvent()

+void UpdateEvent()

+List<Volunteer> matchedVolunteers()

}

class Notification {

+String Message

+Date Date

+User Recipient

+void sendNotification()

}

class ParticipationHistory {

+Event Event

+User Volunteer

+String Status

+void updateStatus(String status)

}

User "1" -- "many" ParticipationHistory : participates

Event "1" -- "many" ParticipationHistory : involves

User "1" -- "many" Notification : receives

Event "1" -- "many" Volunteer : matched

@enduml

The output we get shows in Figure 1.2.

A screenshot of a computer

Description automatically generated

Figure1.2 Class Diagram

## 1.3 Sequence Diagram

The sequence diagram shows the interactions between the front-end and the back-end services.

Step1: Identify interactions: Determine the interactions between different components in the system.

Step2: Draw a diagram: Use the UNL to draw a sequence diagram that represents the sequence of interactions and the participating components.

The code of the Volunteer Sequence Diagram in PlantUML as following:

@startuml

actor Volunteer

participant FrontEnd

participant BackEnd

participant Database

== Registering ==

Volunteer -> FrontEnd: Register

FrontEnd -> BackEnd: POST /register

BackEnd -> Database: Save User Data

Database --> BackEnd: User Saved

BackEnd --> FrontEnd: Registration Success

FrontEnd --> Volunteer: Show Success Message

== Logining ==

Volunteer -> FrontEnd: Login

FrontEnd -> BackEnd: POST /login

BackEnd -> Database: Verify Credentials

Database --> BackEnd: Credentials Verified

BackEnd --> FrontEnd: Login Success

FrontEnd --> Volunteer: Redirect to Dashboard

== Updating profile ==

Volunteer -> FrontEnd: Update Profile

FrontEnd -> BackEnd: Put/Update Profile

BackEnd -> Database: Update User Data

Database -> BackEnd: User Data Updated

BackEnd -> FrontEnd: Profile Update Success

FrontEnd -> Volunteer: Show Success Message

== Viewing Events ==

Volunteer -> FrontEnd: View Events

FrontEnd -> BackEnd: GET /events

BackEnd -> Database: Fetch Events

Database --> BackEnd: Events List

BackEnd --> FrontEnd: Events List

FrontEnd --> Volunteer: Display Events

== Signing Up for Event ==

Volunteer -> FrontEnd: Sign Up for Event

FrontEnd -> BackEnd: POST /signUpEvent

BackEnd -> Database: Save Participation

Database --> BackEnd: Participation Saved

BackEnd --> FrontEnd: Sign Up Success

FrontEnd --> Volunteer: Show Success Message

@enduml

The output we get shows in Figure 1.3.

A diagram of a software project

Description automatically generated with medium confidence

Figure 1.3 Volunteer Sequence Diagram

The code of Administrator Sequence Diagram in PlantUML as following:

@startuml

actor Administrator

participant FrontEnd

participant BackEnd

participant Database

== Creating Event ==

Administrator -> FrontEnd: Fill event form

FrontEnd -> BackEnd: POST /createEvent

BackEnd -> Database: Save Event Data

Database --> BackEnd: Event Saved

BackEnd --> FrontEnd: Event Creation Success

FrontEnd --> Administrator: Show success message

== Matching Volunteer ==

Administrator -> FrontEnd: Request volunteer matching

FrontEnd -> BackEnd: GET /matchVolunteers

BackEnd -> Database: Retrieve Event and Volunteer Data

Database --> BackEnd: Event and Volunteer Data Retrieved

BackEnd -> BackEnd: Match Volunteers to Events

BackEnd --> FrontEnd: Matching Results

FrontEnd --> Administrator: Show matching results

== Allocating Event ==

Administrator -> FrontEnd: Assign volunteers to event

FrontEnd -> BackEnd: POST /assignVolunteers

BackEnd -> Database: Update Volunteer Event-participated

Database --> BackEnd: Event-participated Updated

BackEnd --> FrontEnd: Event-participated Successful

FrontEnd --> Administrator: Show success message

@enduml

The output we get shows in Figure 1.4.

A diagram of a program

Description automatically generated with medium confidence

Figure 1.4 Administrator Sequence Diagram

## 1.4 Activity Diagram

The activity diagram represents the workflow of user registration and profile management.

Step1: Identify activities: Identify the main activities involved in the user registration and data management process.

Step2: Draw diagrams: Use UML to chart activities, representing the sequence of activities and decision points.

The code of Volunteer Registration Activity Diagram in PlantUML as following:

@startuml

start

:Volunteer accesses registration page;

:Volunteer enters email and password;

:Submit registration form;

-> Registration Successful;

:Send verification email;

:Volunteer verifies email;

:Account activation;

stop

@enduml

The output we get shows in Figure 1.5.

A diagram of a program

Description automatically generated

Figure 1.5 Volunteer Registration Activity Diagram

The code of Volunteer Updating Profile Activity Diagram in PlantUML as following:

@startuml

start

:Volunteer logs in;

:Volunteer accesses profile page;

:Volunteer updates profile information;

if (Is all required information valid?) then (yes)

:Submit profile update form;

:Update profile information in database;

-> Update Successful;

else (no)

:Show error message;

endif

stop

@enduml

The output we get shows in Figure 1.6.

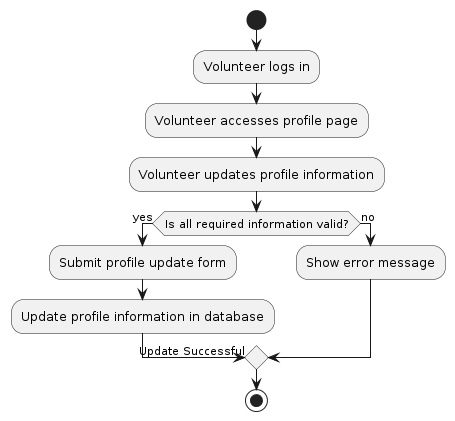


Figure 1.6 Volunteer Updating Profile Activity Diagram

The code of Administrator Creating Events Activity Diagram in PlantUML as following:

@startuml

start

:Administrator logs in;

:Administrator accesses event management page;

:Administrator enters event details;

if (Are all required fields filled?) then (yes)

:Submit event creation form;

:Save event information in database;

-> Event Creation Successful;

else (no)

:Show error message;

endif

stop

@enduml

The output we get shows in Figure 1.7.

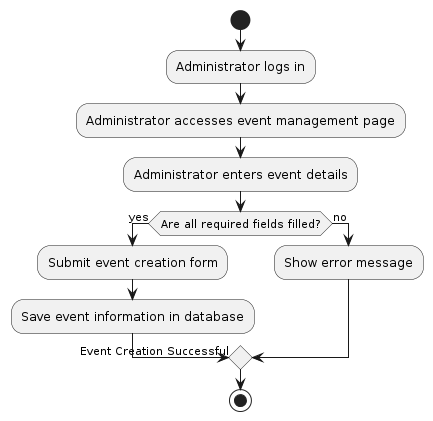


Figure 1.7 Administrator Creating Events Activity Diagram

The code of Administrator Matching Volunteer Activity Diagram as following:

@startuml

start

:Administrator logs in;

:Administrator accesses volunteer matching page;

:Administrator selects an event;

:System retrieves event details and volunteer profiles;

:Administrator initiates volunteer matching;

:Match volunteers based on skills, availability, and location;

:Show matching results;

if (Is administrator satisfied with matches?) then (yes)

:Assign volunteers to event;

:Update volunteer assignments in database;

-> Assignment Successful;

else (no)

:Adjust matching criteria and retry;

endif

stop

@enduml

The output we get shows in Figure 1.8.

A flowchart of a program

Description automatically generated

Figure 1.8 Administrator Matching Volunteer Activity Diagram