Community Connect – Final Report

Jason Gregory

Introduction/Problem Definition

Introduction:

The goal of this project is to develop a prototype database-driven web application, "Community Connect," that serves as a single point of interaction, streamlining the process of connecting volunteers with relevant community organisations and events. This will involve creating a robust relational database and a Python Flask web interface to manage user data, event information, and volunteer registrations, demonstrating a comprehensive understanding of SQL queries, data management and integrity.

Problem Definition:

The "Community Connect" project addresses a problem faced by both community organisations and volunteers. Organisations, particularly non-profits and local groups, often lack an efficient system to manage their volunteer needs for various events, such as fundraisers, clean-up drives, or festivals. Potential volunteers additionally often find it difficult to discover opportunities that align with their interests, skills, and availability. This disconnect leads to missed opportunities for community engagement and inefficient resource allocation.

Investigation

Summaries of two articles discussing the challenges of volunteer management and the importance of community engagement

1. Challenges of Volunteer Management

Article: "The 5 Biggest Volunteer Management Challenges You Might Face"

- **Retention:** Keeping volunteers engaged and committed over time is a significant struggle for many organisations. Lack of recognition, insufficient communication, or unclear roles can lead to high turnover.
- **Undesirable Behaviour:** Volunteer managers often face challenges enforcing rules when volunteers display disruptive behaviour. The risk is that enforcing policies might cause volunteers to withdraw, yet ignoring behaviour can harm team morale. Key solutions involve fostering perceived organisational support and fulfilling psychological contracts—ensuring volunteers feel valued and understood.

- **Resource Constraints:** Lack of administrative support, training, and recognition programs can hinder efforts to recruit and retain volunteers effectively.
- **Sustainability:** Volunteer burnout, lack of professional management, and insufficient support systems can impact the sustainability of volunteer programs. Effective management practices are essential to address these issues.

2. Importance of Community Engagement

Article: "Why Community Engagement Matters" (Penn State University)

- **Trust and Communication:** Engaging communities increases trust in organisations and governance. Inclusive partnerships can address issues more effectively and foster mutual understanding.
- **Effective Solutions:** Involving community members in projects leads to practical, widely accepted outcomes since diverse local knowledge is utilized.
- **Empowerment and Skills:** Community engagement empowers individuals, integrates people from varied backgrounds, and helps citizens develop problem-solving and communication abilities.
- **Networks and Prevention:** It builds local social networks and offers multiple opportunities for people to discuss concerns, allowing early identification and resolution of issues before they escalate.
- **Sustained Success:** Ongoing engagement supports the development of local leadership, strengthens accountability, and enhances the sustainability of community-led projects.

1. https://www.rosterfy.com/blog/the-5-biggest-volunteer-management-challenges-you-might-face
2. https://aese.psu.edu/research/centers/cecd/engagement-toolbox/engagement/why-community-engagement-matters

Project Plan/Timeline

Week 1: Investigation & Design (Aug 18 – Aug 24, 2025)

- Research volunteer platforms and summarise findings.
- Identify entities, attributes, and relationships.
- Develop ERD (Crow's Foot notation, minimum 5 tables, including an M:N relationship).
- Perform normalisation to 3NF and produce relational notation.

- Create a full data dictionary.
- Begin research into ethical and legal considerations (Australian Privacy Principles).

Deliverables: Project management plan, research summary, ERD, normalisation documentation, data dictionary.
Milestone: Database design finalised.

Week 2: Database Implementation & Basic Application (Aug 25 – Aug 31, 2025)

- Build SQLite database with CREATE TABLE and constraints.
- Populate with 15–20 realistic records.
- Set up Flask web app and connect database.
- Implement CRUD functionality (Create volunteer, Read organisations, Update volunteer contact, Delete event).
- Write journal entry on ACID properties.

Deliverables: SQL script, SQLite database file, Flask app files, CRUD screenshots, journal entry.
Milestone: Working prototype with CRUD operations.

Week 3: Advanced Queries & Final Report (Sept 1 – Sept 6, 2025)

- Implement advanced queries in Flask (search by skill, volunteer sign-ups per event, statistics with aggregates, formatted volunteer list with age).
- Write a short discussion on data quality and cleaning.
- Compile final report including design docs, SQL, screenshots, evaluation, and ethical/legal/security issues (APP5, APP10, APP11, APP12).

Deliverables: Final Flask app, SQLite database, comprehensive report.
Milestone: Fully functional system with advanced features and documentation submitted.

6th September 2025 – final application and report due.

Deconstruction

Core Entities	Attributes
Volunteers	**VolunteerID** (Primary Key) - FirstName - LastName - Email - Password - Phone - Address - DateOfBirth - Availability (True/False) - ProfilePhoto - EmergencyContact
Organisations	**OrganisationID** (Primary Key) - Name - Description - ContactPerson - Email - Password - Phone - Address - Website (optional) - Logo (optional)
Events	**EventID** (Primary Key) - Name - Description - Date - StartTime - EndTime - Location - Status (e.g., Upcoming, Ongoing, Completed, Cancelled)
Roles	**RoleID** (Primary Key) - Name - Description - Status (e.g., pending, accepted, denied)
Skills	**SkillID** (Primary Key) - Name - Description (optional)
Entity Relationships	Volunteer -> Skill (M:N) Volunteer -> Event (M:N) Event -> Organisation (M:1) Event -> Skills (M:N) Volunteer -> Roles (M:N) Events -> Roles (M:N)

Normalisation

Note:
Underline indicates Primary Key (PK) (e.g. SkillID)
Italicised + Underline indicates Foreign Key (*FK*) (e.g. *SkillID* FK Skills)

0NF Starting Point

(**VolunteerID**, FirstName, LastName, Password, Email, Phone, Address, DateOfBirth, Availability, ProfilePhoto, EmergencyContact, **SkillID**, SkillName, SkillDescription, **OrganisationID**, OrganisationName, OrganisationDescription, OrganisationContactPerson, OrganisationEmail, OrganisationPassword, OrganisationPhone, OrganisationAddress, OrganisationWebsite, OrganisationLogo, **EventID**, EventName, EventDescription, EventDate, StartTime, EndTime, EventLocation, EventStatus, Roles, RolesDescription)

1NF Normalisation

Table does not satisfy requirements for 1NF normalisation as 'Roles' stores non-atomic values. This can be resolved to 1NF by ensuring all attributes store atomic values. (i.e. 'Roles' -> Role using individual rows for each skill attribute)

Resolved 1NF Relational Notation:

(**VolunteerID**, FirstName, LastName, Password, Email, Phone, Address, DateOfBirth, Availability, ProfilePhoto, EmergencyContact, **SkillID**, SkillName, SkillDescription, OrganisationID, OrganisationName, OrganisationDescription, OrganisationContactPerson, OrganisationEmail, OrganisationPassword, OrganisationPhone, OrganisationAddress, OrganisationWebsite, OrganisationLogo, **EventID**, EventName, EventDescription, EventDate, StartTime, EndTime, EventLocation, EventStatus, **RoleID**, Role, RoleDescription, RoleStatus)

2NF Normalisation

Table does not satisfy requirements for 2NF normalisation due to the presence of partial dependencies; there is a quadruple composite key (VolunteerID, SkillID, EventID, RoleID) (OrganisationID not required as EventID -> OrganisationID is M:1)

VolunteerID alone identifies a volunteer, but a volunteer can have many skills and participate in many events associated with different organisations.
SkillID alone identifies a skill, but many volunteers can have many different skills across many events.
EventID identifies the event, but many events can have many skills and volunteers.

RoleID identifies the role of a volunteer, but many volunteers can have many roles in many events.

(OrganisationID depends on **EventID** and thus all attributes depending on OrganisationID have transitive dependencies.)

To resolve this, all attributes depending only on 1 part of the composite key must be separated into their own respective tables. This requires creation of multiple junction tables resolving M:N relationships to ensure no partial dependencies.

M:N Data Relationships:

Volunteers -> Skills (M:N)
Events -> Skills (M:N)
Events -> Volunteers (M:N)

And thus **partial dependencies** are as follows:

FirstName, LastName, Email, Password, Phone, Address, DateOfBirth, Availability, ProfilePhoto, EmergencyContact -> **VolunteerID**

EventName, EventDescription, EventDate, StartTime, EndTime, EventLocation, EventStatus, OrganisationID -> **EventID**

SkillName, SkillDescription -> **SkillID**

RoleName, RoleDescription -> **RoleID**

Resolved 2NF Relational Notation:

Volunteers (**VolunteerID**, FirstName, LastName, Email, Password, Phone, Address, DateOfBirth, Availability, ProfilePhoto, EmergencyContact)

VolunteerSkills(***VolunteerID*** **FK Volunteers**, ***SkillID*** **FK Skills**)

Skills (**SkillID**, Name, Description)

Organisations (**OrganisationID**, Name, Email, Password, Description, ContactPerson, Email, Phone, Address, Website, Logo)

EventSkills(***EventID*** **FK Events**, ***SkillID*** **FK Skills**)

Signups(**ID,** ***EventID*** **FK Events**, ***VolunteerID*** **FK Volunteers**, ***RoleID*** **FK Roles**, Status)

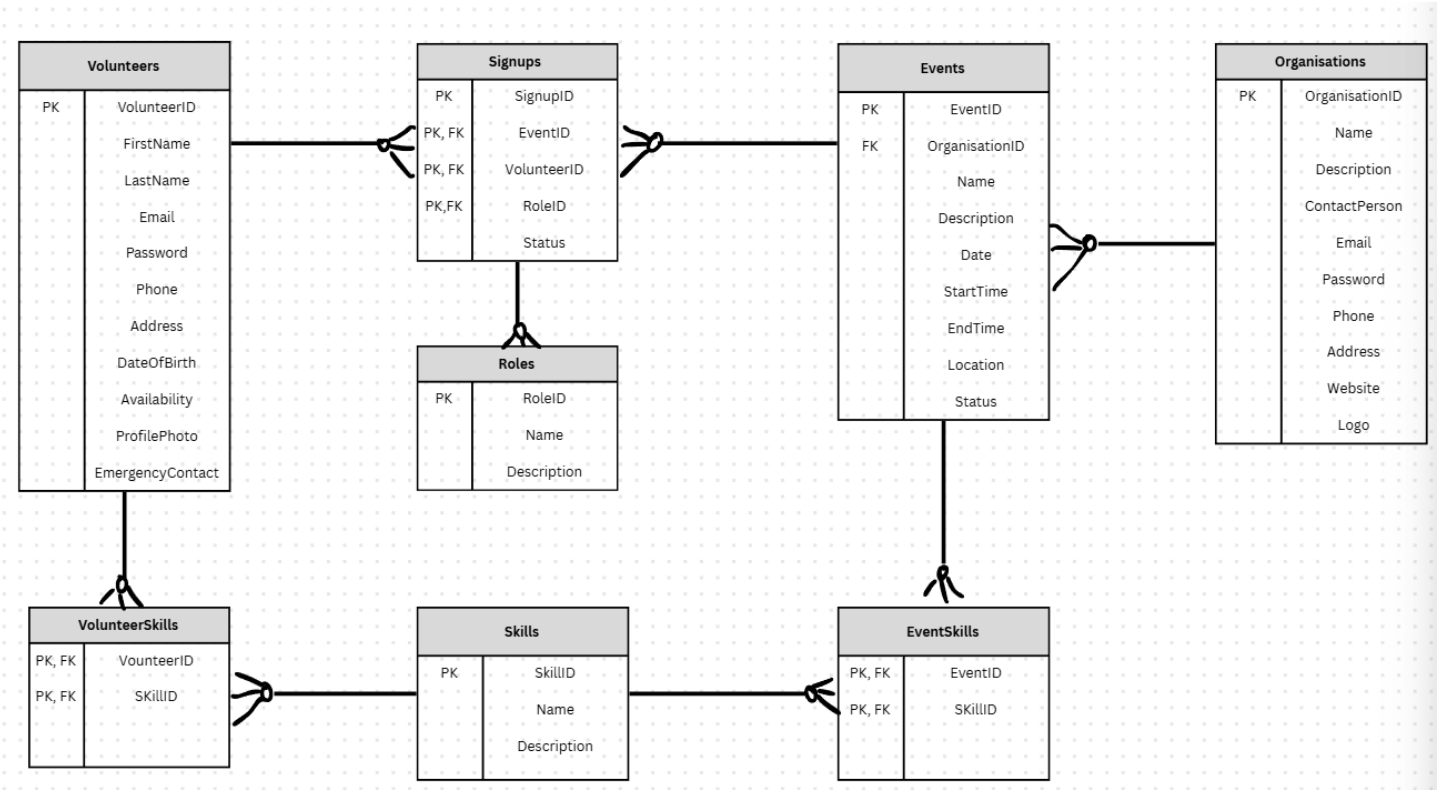
Roles(**RoleID**, Name, Description)

Events (**EventID**, ***OrganisationID*** **FK Organisations**, Name, Description, Date, StartTime, EndTime, Location, Status)

3NF Normalisation

For a database to satisfy 3NF normalisation, it must already be in 2NF, as well as have no transitive dependencies. All transitive dependencies have been eliminated, and thus the schema is already in 3NF. (Organisation attributes -> OrganisationID -> **EventID** transitive dependency has been resolved)

ERD



Data Dictionaries

Volunteers

ElementName	DataType	Size	Description	Constraints
VolunteerID	Integer		Unique Identifier for Volunteer	PK, Unique, Not Null
FirstName	Text	20	First Name of Volunteer	Not Null
LastName	Text	20	Last Name of Volunteer	Not Null
Email	Text	320	Email Address	Unique, Not Null
Password	Text	128	Hashed Password	Not Null
Phone	Text	32	Phone Number	
Address	Text	255	Home Address	
DateOfBirth	Date		Date of Birth	
Availability	Boolean		Availability Status	
ProfilePhoto	BLOB		Profile Photo	
EmergencyContact	Text	32	Emergency Contact Number	

Events

ElementName	DataType	Size	Description	Constraints
EventID	Integer		Unique identifier for Event	PK, Unique, Not Null
OrganisationID	Integer		Unique identifier for the Organising Organisation	FK → Organisations
Name	Text	100	Name of Event	Not Null
Description	Text	255	Description of Event	
Date	Date		Date of Event	
StartTime	Time		Start Time	
EndTime	Time		End Time	
Location	Text	255	Location of Event	
Status	Text	20	Status (Upcoming, etc.)	

Organisations

ElementName	DataType	Size	Description	Constraints
OrganisationID	Integer		Unique Identifier for Organisation	PK, Unique, Not Null
Name	Text	100	Organisation Name	Not Null
Description	Text	255	Organisation Description	
ContactPerson	Text	50	Contact Person	
Email	Text	320	Contact Email	Unique, Not Null
Password	Text	128	Hashed Password	Not Null
Phone	Text	32	Contact Phone	
Address	Text	255	Address	
Website	Text	255	Website URL	
Logo	BLOB		Organisation Logo	

Skills

ElementName	DataType	Size	Description	Constraints
SkillID	Integer		Unique identifier for Skill	PK, Unique, Not Null
Name	Text	50	Name of Skill	Not Null
Description	Text	255	Description of Skill	

Roles

ElementName	DataType	Size	Description	Constraints
RoleID	Integer		Unique identifier for Role	PK, Unique, Not Null
Name	Text	50	Name of Role	Not Null
Description	Text	255	Description of Role	

Signups

ElementName	DataType	Size	Description	Constraints
SignupID	Integer		Unique identifier for Signup	PK, Unique, Not Null
EventID	Integer		Unique identifier for Event being signed up for	FK → Events, Not Null
VolunteerID	Integer		Volunteer signing up	FK → Volunteers, Not Null
RoleID	Integer		Role assigned	FK → Roles, Nullable
Status	Text	20	Status (Pending, Confirmed, etc.)	

EventSkills

ElementName	DataType	Size	Description	Constraints
EventID	Integer		Unique identifier for Event needing the skill	PK, FK → Events, Not Null
SkillID	Integer		Unique identifier for Required skill for the event	PK, FK → Skills, Not Null

(Composite PK: EventID + SkillID)

VolunteerSkills

ElementName	DataType	Size	Description	Constraints
VolunteerID	Integer		Unique identifier for Volunteer with this skill	PK, FK → Volunteers, Not Null
SkillID	Integer		Unique identifier for Skill of the volunteer	PK, FK → Skills, Not Null

(Composite PK: VolunteerID + SkillID)

SQL Scripts from week 2

Table Creation:

```
-- Drop tables if exist (reverse order of dependencies)
DROP TABLE IF EXISTS Signups;
DROP TABLE IF EXISTS EventSkills;
DROP TABLE IF EXISTS VolunteerSkills;
DROP TABLE IF EXISTS Events;
DROP TABLE IF EXISTS Organisations;
DROP TABLE IF EXISTS Volunteers;
DROP TABLE IF EXISTS Roles;
DROP TABLE IF EXISTS Skills;

-- Organisations Table
CREATE TABLE Organisations (
  OrganisationID INTEGER PRIMARY KEY AUTOINCREMENT,
  Name TEXT(100) NOT NULL,
  Description TEXT(255),
  ContactPerson TEXT(50),
  Email TEXT(320) UNIQUE NOT NULL,
  Password TEXT(128) NOT NULL,
  Phone TEXT(32),
  Address TEXT(255),
  Website TEXT(255),
  Logo BLOB
);

-- Volunteers Table
CREATE TABLE Volunteers (
```

```
    VolunteerID INTEGER PRIMARY KEY AUTOINCREMENT,
    FirstName TEXT(20) NOT NULL,
    LastName TEXT(20) NOT NULL,
    Email TEXT(320) UNIQUE NOT NULL,
    Password TEXT(128) NOT NULL,
    Phone TEXT(32),
    Address TEXT(255),
    DateOfBirth DATE,
    Availability BOOLEAN,
    ProfilePhoto BLOB,
    EmergencyContact TEXT(32)
);

-- Roles Table
CREATE TABLE Roles (
    RoleID INTEGER PRIMARY KEY AUTOINCREMENT,
    Name TEXT(50) NOT NULL,
    Description TEXT(255)
);

-- Skills Table
CREATE TABLE Skills (
    SkillID INTEGER PRIMARY KEY AUTOINCREMENT,
    Name TEXT(50) NOT NULL,
    Description TEXT(255)
);

-- Events Table
```

```
CREATE TABLE Events (
  EventID INTEGER PRIMARY KEY AUTOINCREMENT,
  OrganisationID INTEGER NOT NULL,
  Name TEXT(100) NOT NULL,
  Description TEXT(255),
  Date DATE,
  StartTime TIME,
  EndTime TIME,
  Location TEXT(255),
  Status TEXT(20),
  FOREIGN KEY (OrganisationID) REFERENCES Organisations(OrganisationID),
  CONSTRAINT chk_end_after_start CHECK (EndTime > StartTime)
);

-- VolunteerSkills (Many-to-Many Volunteers <> Skills)
CREATE TABLE VolunteerSkills (
  VolunteerID INTEGER NOT NULL,
  SkillID INTEGER NOT NULL,
  PRIMARY KEY (VolunteerID, SkillID),
  FOREIGN KEY (VolunteerID) REFERENCES Volunteers(VolunteerID) ON DELETE
CASCADE,
  FOREIGN KEY (SkillID) REFERENCES Skills(SkillID) ON DELETE CASCADE
);

-- EventSkills (Many-to-Many Events <> Skills)
CREATE TABLE EventSkills (
  EventID INTEGER NOT NULL,
  SkillID INTEGER NOT NULL,
```

```
  PRIMARY KEY (EventID, SkillID),
  FOREIGN KEY (EventID) REFERENCES Events(EventID) ON DELETE CASCADE,
  FOREIGN KEY (SkillID) REFERENCES Skills(SkillID) ON DELETE CASCADE
);

-- Signups Table (Many-to-Many Volunteers <> Events with Roles)
CREATE TABLE Signups (
  SignupID INTEGER PRIMARY KEY AUTOINCREMENT,
  EventID INTEGER NOT NULL,
  VolunteerID INTEGER NOT NULL,
  RoleID INTEGER,
  Status TEXT(20),
  FOREIGN KEY (EventID) REFERENCES Events(EventID) ON DELETE CASCADE,
  FOREIGN KEY (VolunteerID) REFERENCES Volunteers(VolunteerID) ON DELETE CASCADE,
  FOREIGN KEY (RoleID) REFERENCES Roles(RoleID) ON DELETE SET NULL
);
```

Data Population:

```
-- Clear existing data (respecting FK order)
DELETE FROM Signups;
DELETE FROM EventSkills;
DELETE FROM VolunteerSkills;
DELETE FROM Events;
DELETE FROM Organisations;
DELETE FROM Volunteers;
DELETE FROM Roles;
DELETE FROM Skills;
```

-- Insert Organisations

INSERT INTO Organisations (Name, Description, ContactPerson, Email, Password, Phone, Address, Website, Logo)

VALUES

('Helping Hands', 'Non-profit focused on community outreach and events.', 'Alice Johnson', 'contact@helpinghands.org', 'hashed_pw1', '555-1234', '123 Main St', 'https://helpinghands.org', NULL),

('Green Earth', 'Organisation dedicated to environmental conservation.', 'Bob Smith', 'info@greenearth.org', 'hashed_pw2', '555-5678', '456 Forest Rd', 'https://greenearth.org', NULL),

('Food For All', 'Charity focused on providing meals to the homeless.', 'Maria Lopez', 'support@foodforall.org', 'hashed_pw3', '555-8765', '22 Market St', 'https://foodforall.org', NULL),

('Tech Volunteers', 'Organisation connecting volunteers with IT skills to NGOs.', 'David Chen', 'admin@techvolunteers.org', 'hashed_pw4', '555-1212', '900 Tech Ave', 'https://techvolunteers.org', NULL);

-- Insert Volunteers

INSERT INTO Volunteers (FirstName, LastName, Email, Password, Phone, Address, DateOfBirth, Availability, ProfilePhoto, EmergencyContact)

VALUES

('John', 'Doe', 'john.doe@email.com', 'hashed_pw5', '555-7890', '789 Pine St', '1990-05-14', 1, NULL, '555-1111'),

('Emma', 'Brown', 'emma.brown@email.com', 'hashed_pw6', '555-2222', '321 Oak Ave', '1995-09-20', 0, NULL, '555-3333'),

('Liam', 'Nguyen', 'liam.nguyen@email.com', 'hashed_pw7', '555-4444', '987 Cedar Blvd', '1988-12-02', 1, NULL, '555-5555'),

('Sophia', 'Khan', 'sophia.khan@email.com', 'hashed_pw8', '555-6666', '12 River St', '1993-03-08', 1, NULL, '555-7777'),

('Ethan', 'Wong', 'ethan.wong@email.com', 'hashed_pw9', '555-8888', '33 Ocean Dr', '2000-11-15', 1, NULL, '555-9999');

Community Connect Project; Final Report – Jason Gregory

```
-- Insert Roles
INSERT INTO Roles (Name, Description)
VALUES
('Team Leader', 'Responsible for managing a group of volunteers.'),
('Helper', 'General assistance role.'),
('Medical Aid', 'Provides first aid and health-related support.'),
('Cook', 'Prepares and serves meals.'),
('Technical Support', 'Helps set up and troubleshoot technical systems.');

-- Insert Skills
INSERT INTO Skills (Name, Description)
VALUES
('First Aid', 'Basic first aid and CPR knowledge.'),
('Cooking', 'Ability to prepare and serve meals.'),
('Event Management', 'Organising and coordinating events.'),
('IT Support', 'Technical troubleshooting and setup.'),
('Logistics', 'Managing supplies, transport, and resources.');

-- Insert Events
INSERT INTO Events (OrganisationID, Name, Description, Date, StartTime, EndTime, Location, Status)
VALUES
(1, 'Community Clean-up', 'Neighborhood clean-up event.', '2025-09-15', '09:00', '12:00', 'Central Park', 'Upcoming'),
(2, 'Tree Planting Drive', 'Planting trees to promote sustainability.', '2025-10-01', '08:30', '11:30', 'Riverside Grounds', 'Upcoming'),
(3, 'Soup Kitchen', 'Serving hot meals to the homeless.', '2025-09-20', '11:00', '14:00', 'Downtown Shelter', 'Upcoming'),
```

```
(4, 'NGO Tech Fair', 'Tech workshops for non-profits.', '2025-11-05', '10:00', '16:00', 'Tech Hub', 'Upcoming'),

(1, 'Fundraising Gala', 'Annual dinner to raise funds.', '2025-12-10', '18:00', '22:00', 'City Hall', 'Planned');


-- Insert VolunteerSkills

INSERT INTO VolunteerSkills (VolunteerID, SkillID)

VALUES

(1, 1), (1, 3),      -- John: First Aid, Event Management

(2, 2),              -- Emma: Cooking

(3, 1), (3, 2),      -- Liam: First Aid, Cooking

(4, 4), (4, 5),      -- Sophia: IT Support, Logistics

(5, 2), (5, 5);      -- Ethan: Cooking, Logistics


-- Insert EventSkills

INSERT INTO EventSkills (EventID, SkillID)

VALUES

(1, 1), (1, 3),      -- Clean-up: First Aid + Event Mgmt

(2, 2), (2, 3),      -- Tree Planting: Cooking + Event Mgmt

(3, 2), (3, 5),      -- Soup Kitchen: Cooking + Logistics

(4, 4), (4, 3),      -- Tech Fair: IT Support + Event Mgmt

(5, 3), (5, 5);      -- Gala: Event Mgmt + Logistics


-- Insert Signups

INSERT INTO Signups (EventID, VolunteerID, RoleID, Status)

VALUES

(1, 1, 1, 'Confirmed'),   -- John = Team Leader for Clean-up

(1, 2, 2, 'Pending'),     -- Emma = Helper for Clean-up

(2, 3, 3, 'Confirmed'),   -- Liam = Medical Aid for Tree Planting
```

(2, 5, 2, 'Confirmed'), -- Ethan = Helper for Tree Planting

(3, 2, 4, 'Confirmed'), -- Emma = Cook for Soup Kitchen

(3, 4, NULL, 'Pending'), -- Sophia = signed up, role TBD (edge case: nullable RoleID)

(4, 4, 5, 'Confirmed'), -- Sophia = Tech Support at Tech Fair

(5, 1, 1, 'Confirmed'), -- John = Team Leader for Gala

(5, 3, 2, 'Confirmed'); -- Liam = Helper for Gala

Basic CRUD Functionality:

Create:

INSERT INTO Volunteers (FirstName, LastName, Email, Password, Phone, Address, DateOfBirth, Availability, EmergencyContact) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)

Read:

SELECT * FROM Organisations

Update:

UPDATE Volunteers SET Phone = ? WHERE VolunteerID = ?

Delete:

DELETE FROM Events WHERE EventID = ?

Screen Captures showing all basic + advanced features

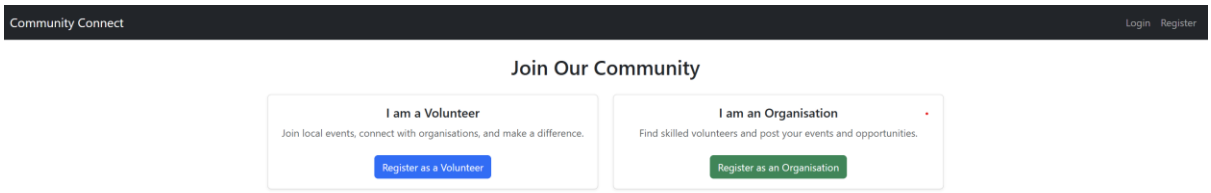


Figure 1 Register Select

Figure 2 Volunteer Registration

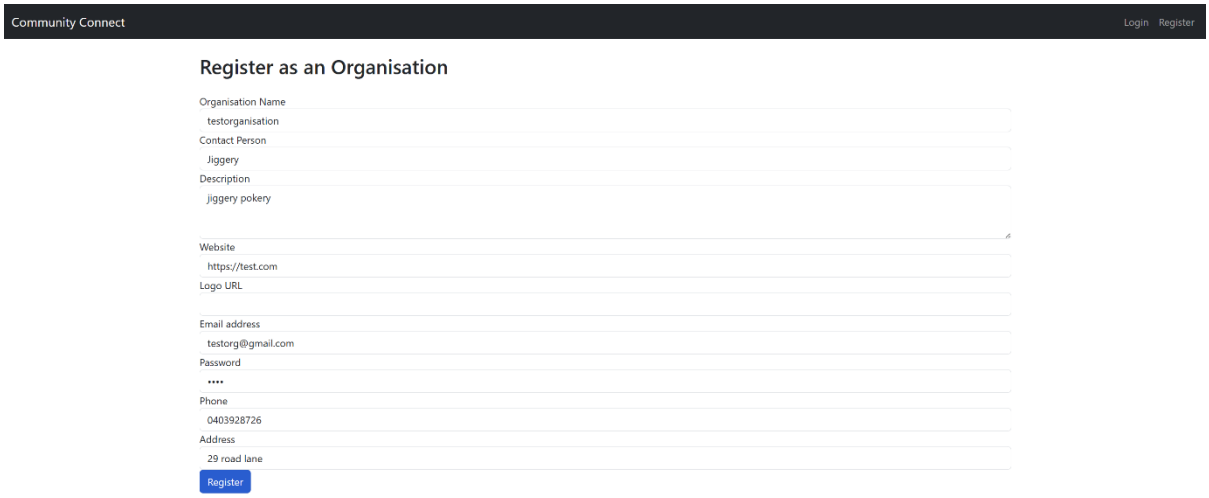


Figure 3 Organisation Registration

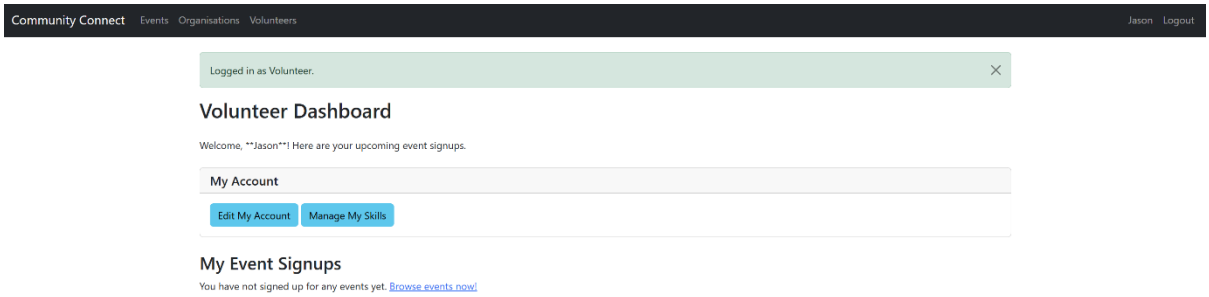


Figure 4 Volunteer Dashboard

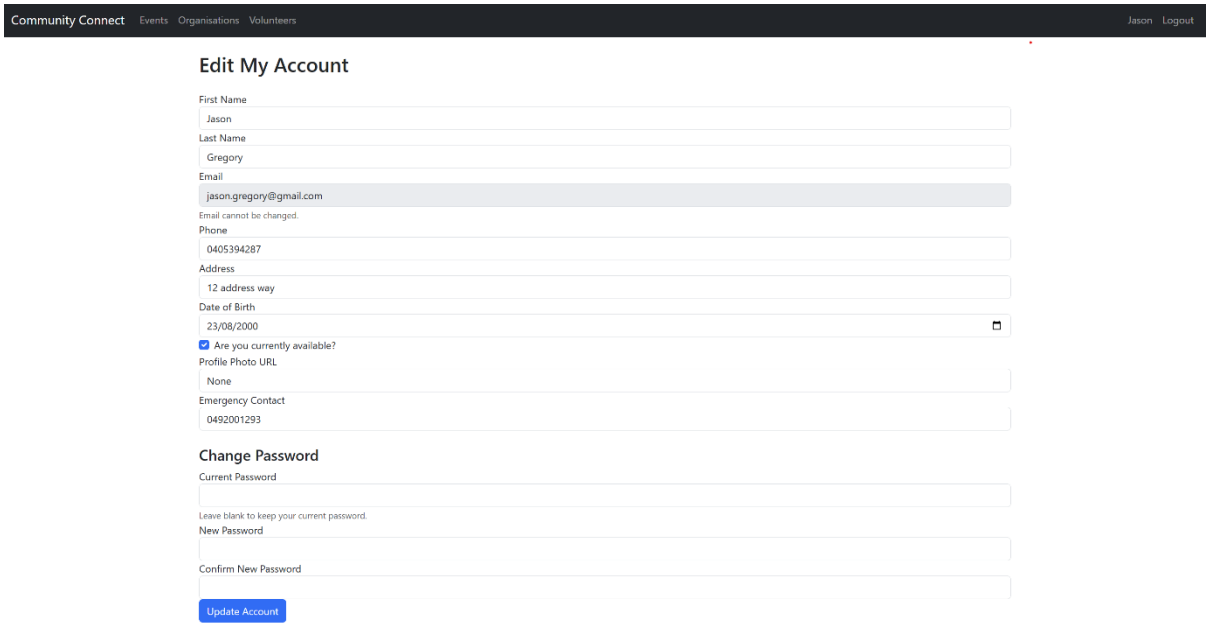


Figure 5 Volunteer Edit/Update Account

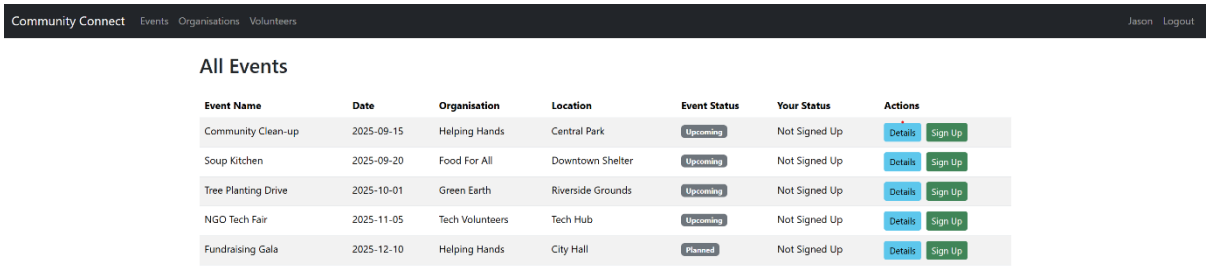


Figure 6 View Events – Volunteer View

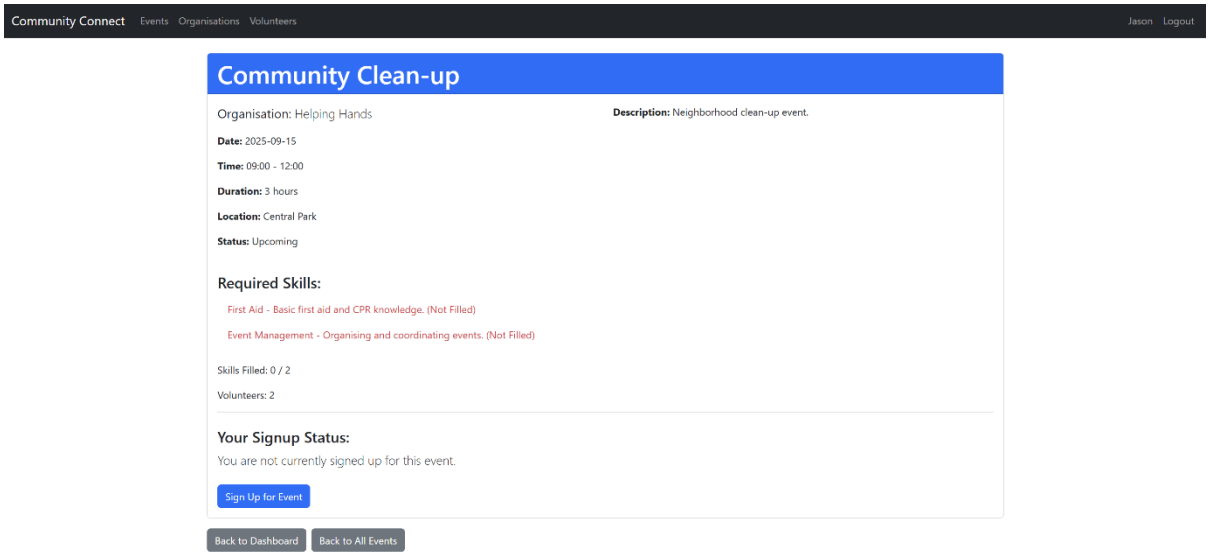


Figure 7 View Event – Volunteer View/Signup Page

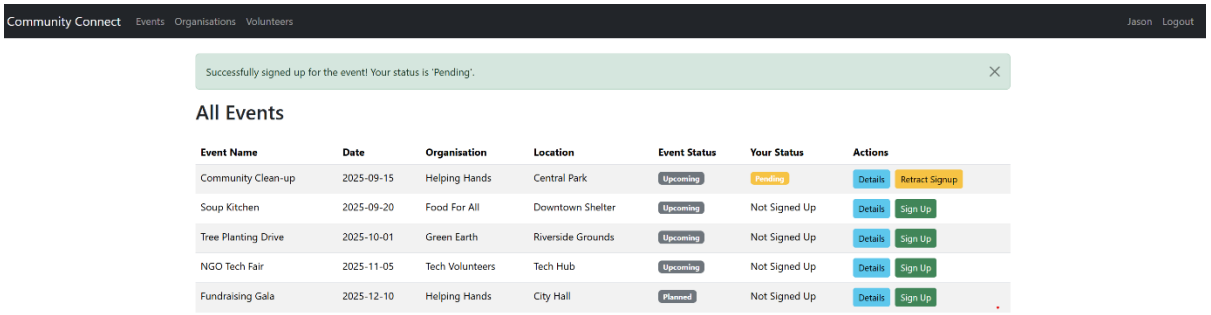


Figure 8 Event List Volunteer View; Events Signup Page

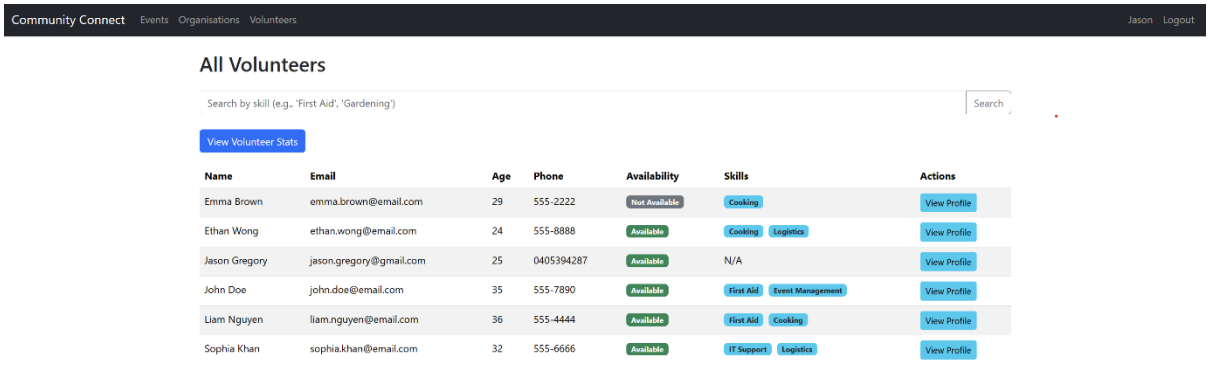


Figure 9 View Volunteers

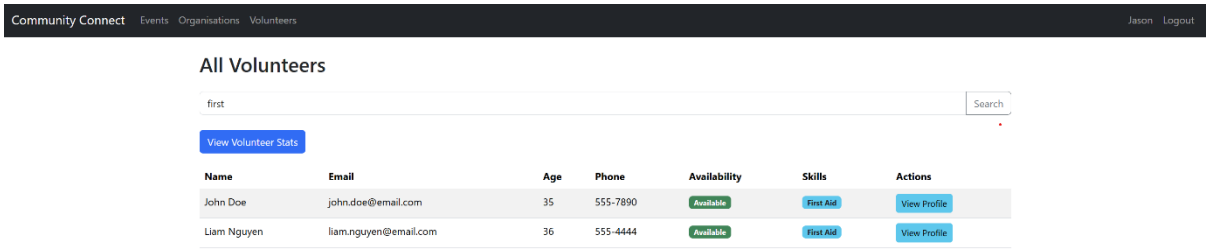


Figure 10 View Volunteers; Search Volunteers By Skill (e.g. First Aid)

Figure 11 View Volunteer Skill Distribution Stats/Counts

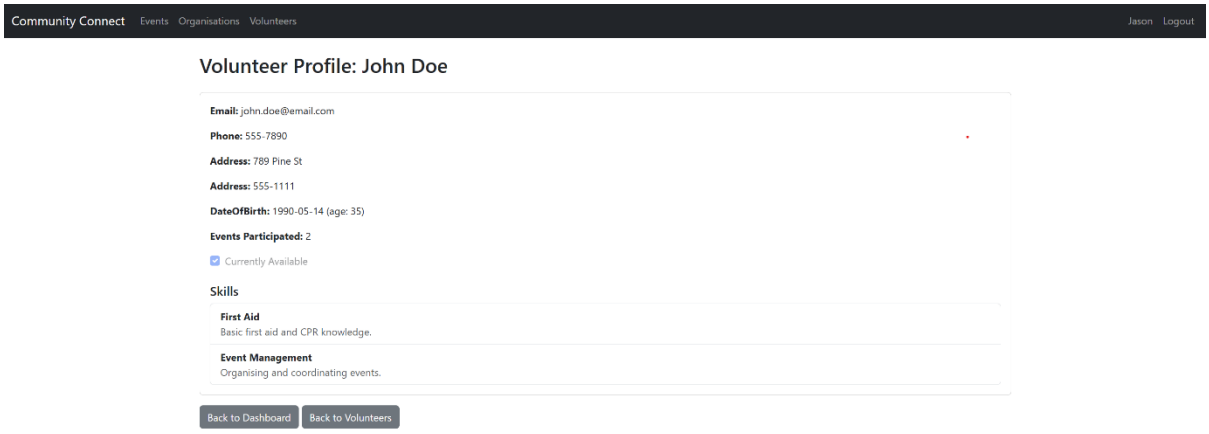


Figure 12 View Volunteer Profile

Figure 13 Edit (own) Volunteer Account Skills

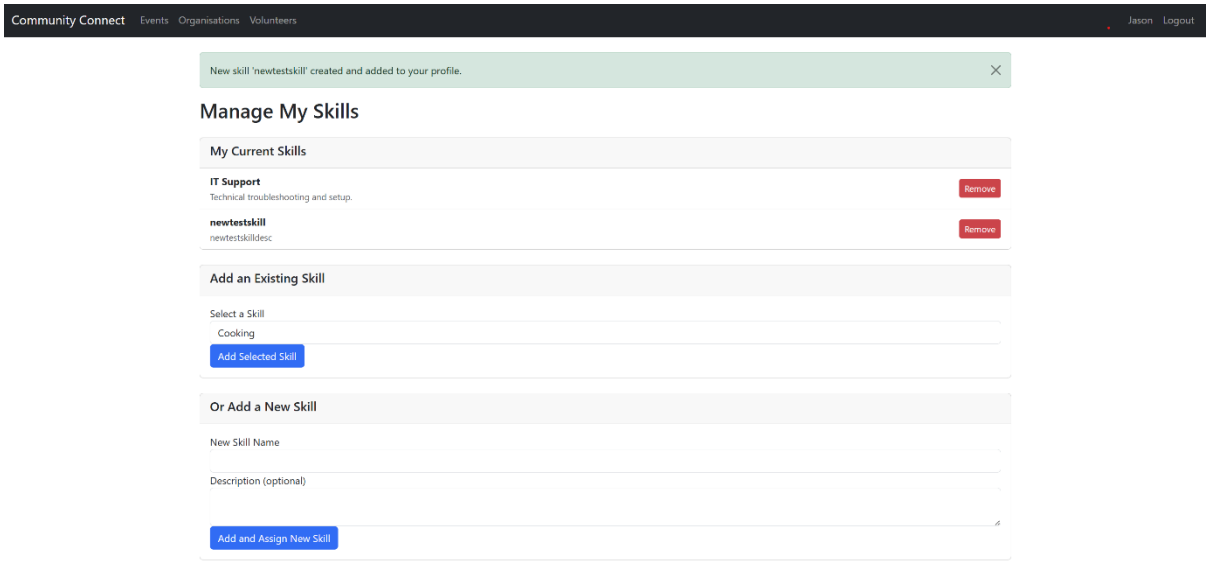


Figure 14 Edit (own) Volunteer Account Skills; Skills Updated

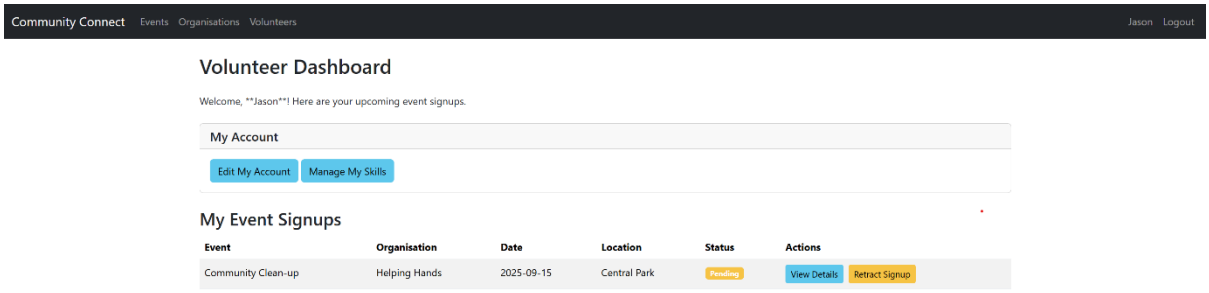


Figure 15 Volunteer Dashboard After signing up for event

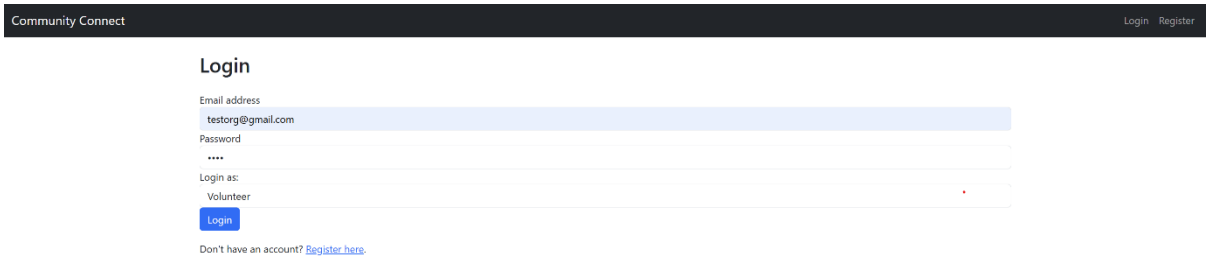


Figure 16 Organisation Login

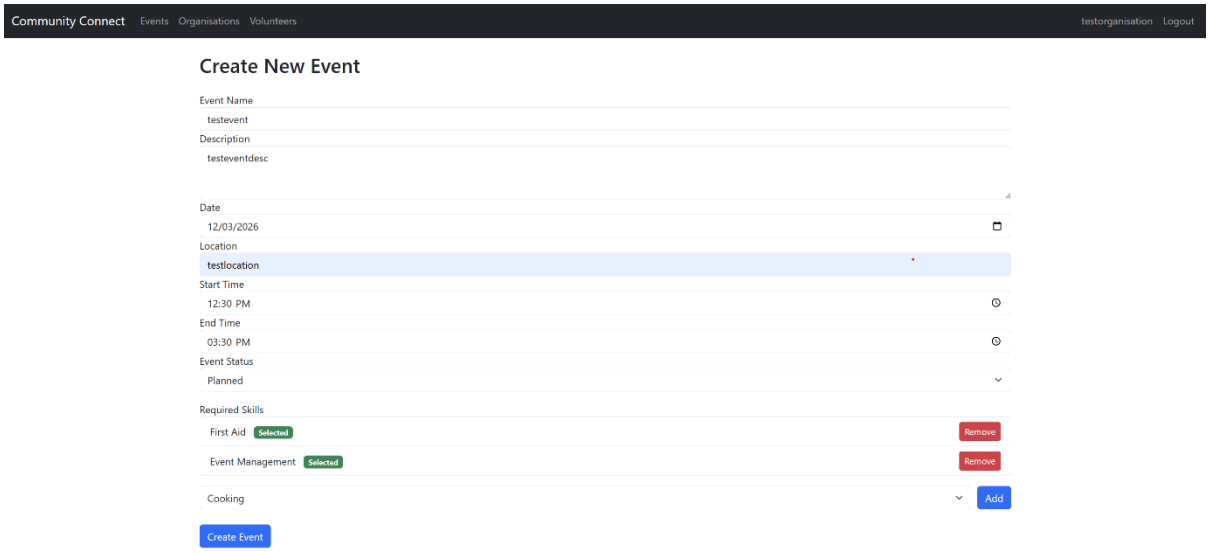


Figure 17 Create New Event

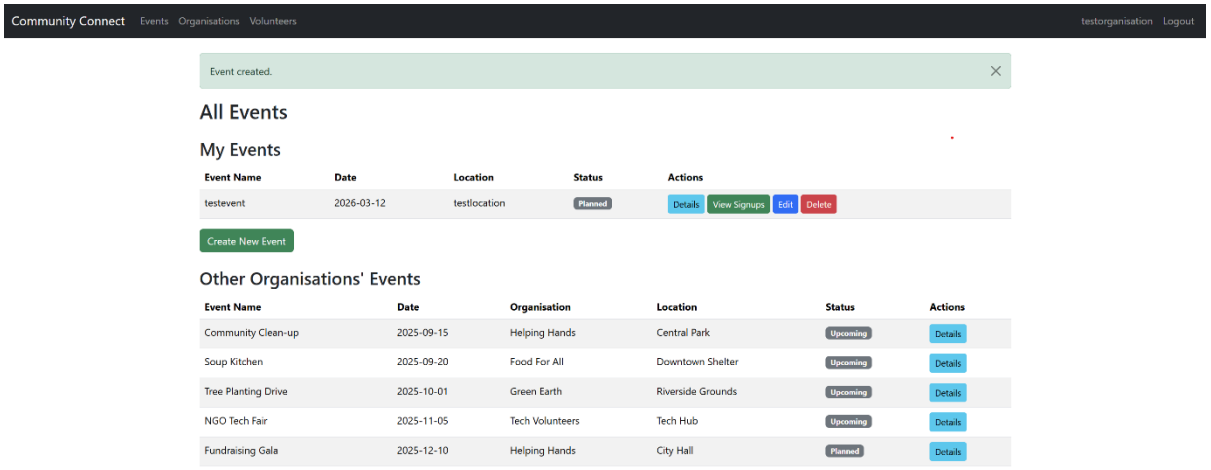


Figure 18 View Events; Organisation View

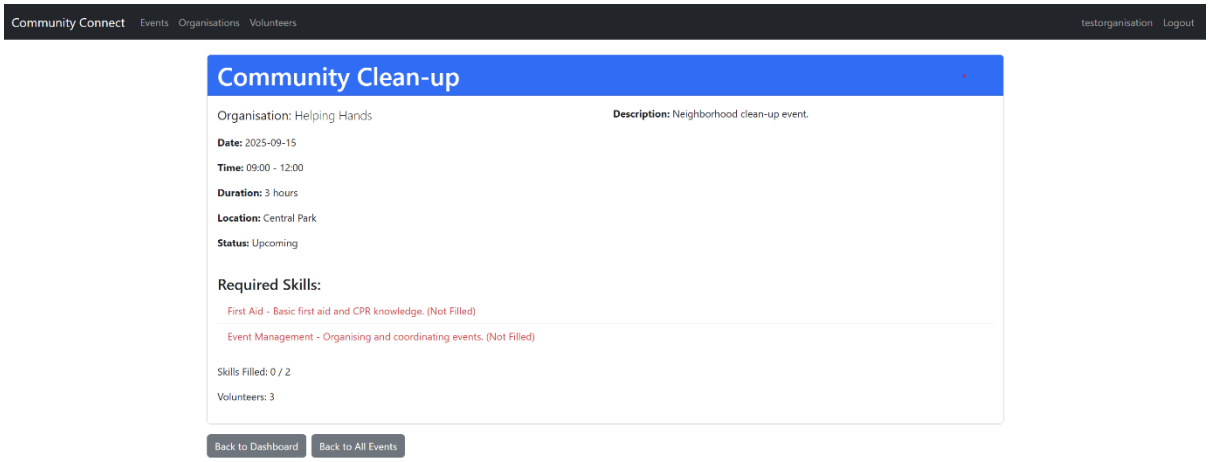


Figure 19 View Event; Organisation View

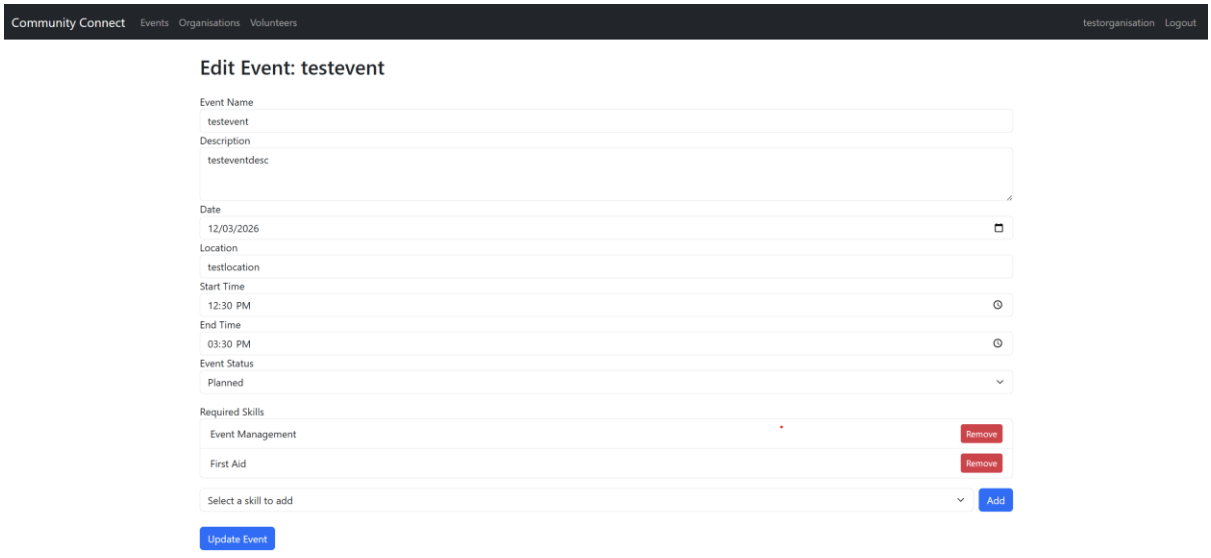


Figure 20 Edit Event - Organisation View

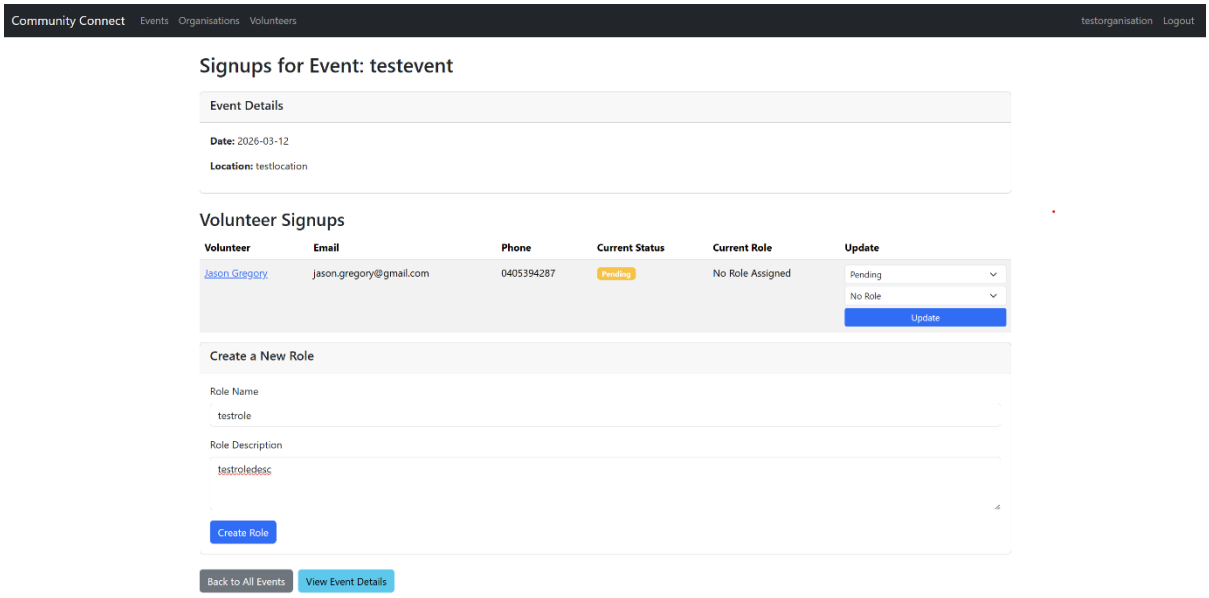


Figure 21 Signups For Event - Organisation View

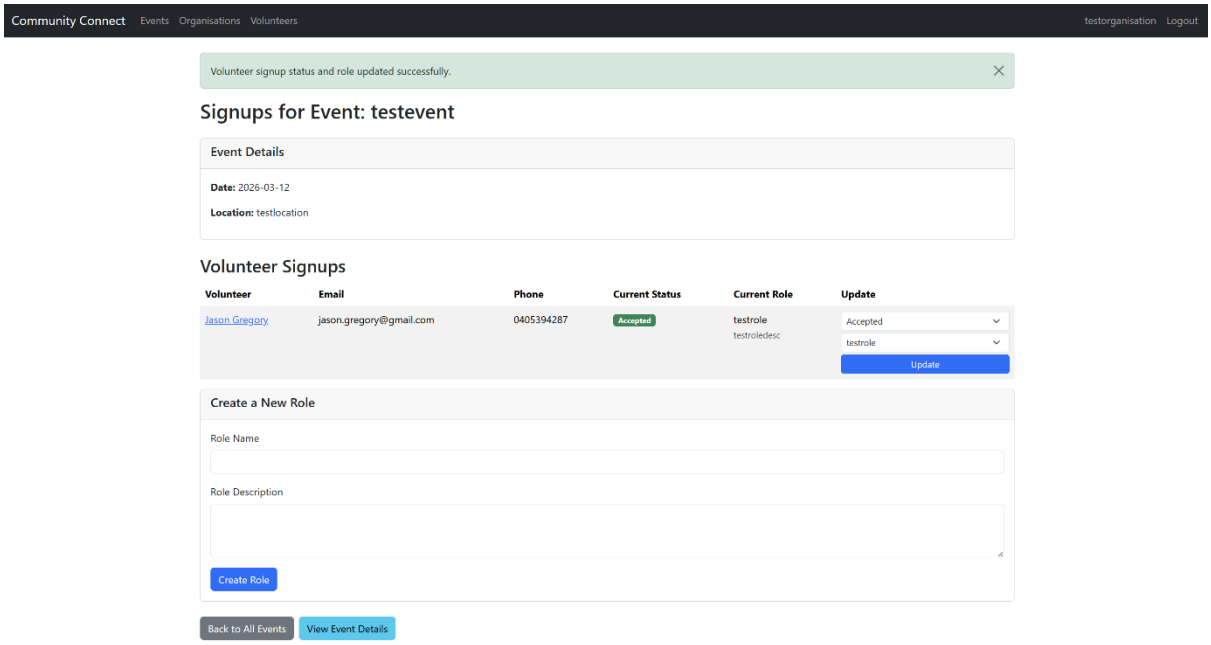


Figure 22 Signup Updated

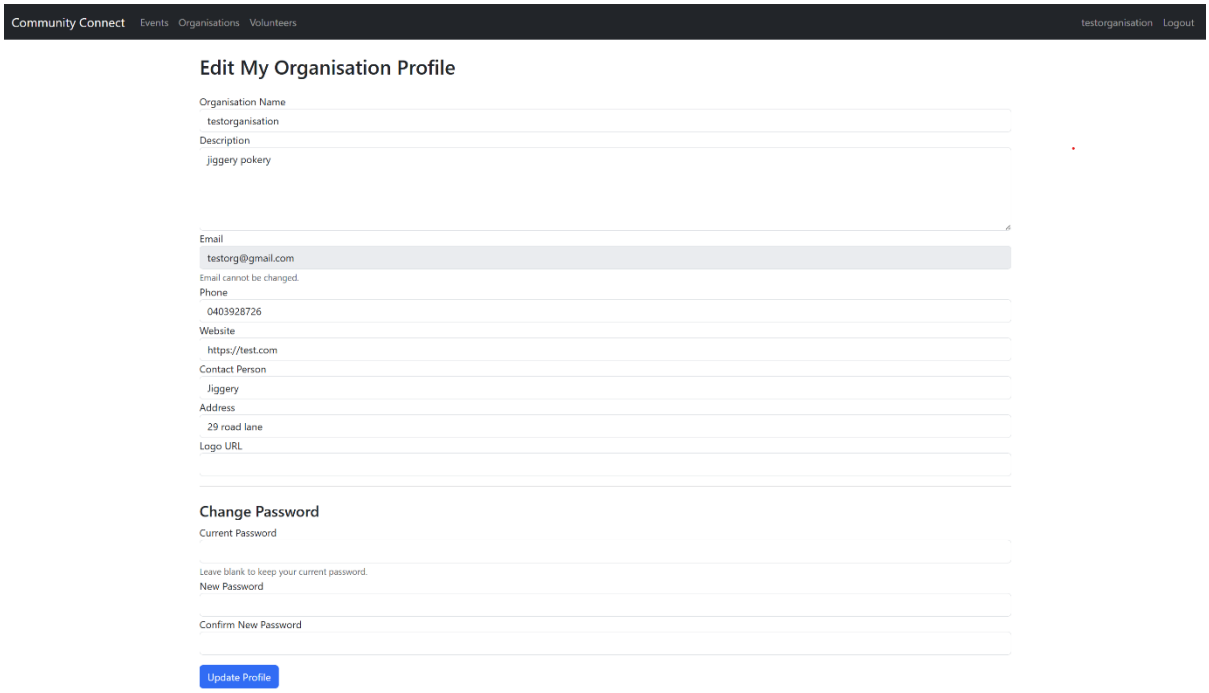


Figure 23 Edit Organisation Profile

Evaluation (final product vs initial requirements – limitations and potential future improvements)

The Community Connect flask application successfully meets the initial requirements of a prototype system for volunteer coordination. The relational database design, including the ER diagram and normalized schema, provides a solid foundation for managing volunteer and organisation data, event listings, and registrations. The implemented SQL queries (CREATE, INSERT, SELECT, UPDATE, DELETE) demonstrate the ability to perform basic data manipulation, which is crucial for the application's functionality. The Python Flask web application provides a functional user interface that allows organisations to post events and volunteers to sign up for them. The system effectively stores and retrieves data, ensuring that volunteer skills can be matched with event requirements, and that organisations can view who has signed up for their events. This therefore appropriately solves the core problem identified in the project's introduction. Current limitations include a lack of a more advanced and intuitive user interface, lack of scalability (due to the nature of an SQLite database), missing security protocols and lack of more advanced features/views. Potential improvements may include a notification system, feedback and rating system, real-time chat, and user authentication and authorisation.

Development issues (ethical, legal, security)

Ethical

Privacy Concerns: The project handles personal information, including names, contact details, and potentially sensitive information like skills and availability. It is crucial to be transparent about how this data is collected, stored, and used. Users should have clear visibility into what data is being shared with organizations and what is kept private.

Appropriate Use of Data (Relevancy): The data collected should only be used for its intended purpose: connecting volunteers with organizations. It should not be used for marketing unrelated products, sold to third parties, or used for any purpose beyond the scope of the "Community Connect" platform.

Data Mining Potential: The database contains valuable information about volunteer behaviour, skill sets, and organizational needs. There is a potential ethical risk if this data is mined for insights that could be used for commercial or non-community purposes without user consent. The system design and usage policy must strictly prohibit such activities.

Legal

The Australian Privacy Principles (APPs) outline how Australian government agencies and organizations must handle personal information. The "Community Connect" design must comply with these principles, specifically:

- **APP 5 (Notification of Collection):** APP 5 requires that the entity collects personal information about an individual to take reasonable steps either to notify the individual of certain matters or to ensure the individual is aware of those matters.
  This is achieved in my app via the clarity of input fields at every data collection point, allowing users to know clearly when their data is being collected/stored (e.g. when updating profile a flash message informs the user their data is being updated). A clear privacy policy at each data collection field would improve compliance to this APP.
- **APP 10 (Quality of Personal Information):** The system must take reasonable steps to ensure that the personal information it collects, uses, and discloses is accurate, up-to-date, and complete.
  This is achieved in my app via the ease of updating data in profiles and events (for organisations and volunteers). Interval prompting of users for updating of data could improve compliance.
- **APP 11 (Security):** The application must protect the personal information it holds from misuse, interference, and loss, as well as from unauthorized access,

modification, or disclosure.
My app does this by hashing passwords, only collecting relevant data (data minimization), and limiting access to data depending on the user (e.g. only specific organisation can edit only their own events). Data could be encrypted or stored more securely to improve compliance here.

- **APP 12 (Access to Personal Information):** The system must provide a mechanism for individuals to access their personal information held by the application.
  My app does this by allowing users to edit all of their own personal information at any point easily by interacting with the database via the flask frontend/app.

Security

Restricting Access: Security measures are vital to protect the integrity of the database. Access to the database should be restricted to authorized users and processes. In a full-scale application, this would involve a multi-layered security approach including secure user authentication, role-based access control, and network security measures to prevent unauthorized access to the database server. In my current application it simply extends to restricting access to specific data access filtered by account type.

Importance of Backups: Data loss can occur due to hardware failure, software corruption, or malicious attacks. A comprehensive backup strategy is essential to ensure that data can be restored. Regular, automated backups of the entire database and user-uploaded content are critical for business continuity and data recovery. In this project that could simply extend to storing a copy of the database elsewhere.

Data Ownership: Clear policies on data ownership must be established. The data created by volunteers and organizations belongs to them. The "Community Connect" platform is the custodian of this data, not the owner; the terms of service must explicitly state that users can request to have their data deleted and that the platform will not sell or misuse their information.