**Computer Science Project Report**

**Introduction**

In today’s society, volunteering plays a crucial report in strengthening communities by enabling individuals to contribute their time, skills, and passion to meaningful causes. Though a big issue with volunteering, is the fact that non-profit organisations and community groups often struggle to attract, manage, and retain reliable volunteers for their events and initiatives. Despite the increasing demand for volunteer support, there is a noticeable gap between individuals and organisations in need, where this disconnect frequently stems from the lack of a streamlined system for matching volunteers to suitable opportunities based on their skills, availability, and interests.

Traditionally, organisations rely on word of mouth, email lists, or social media, to reach out to potential volunteers. While these methods may yield results, they are often inefficient, unstructured, and do not guarantee the recruitment of volunteers with the right qualifications. Furthermore, individuals interested in volunteering may find it difficult to discover opportunities that align with their personal strengths, location, or availability. The absence of a centralised platform creates barriers on both sides of the situation, reducing overall community engagement, and limiting the impact of charitable initiatives.

To address these challenges, the Community Connect web application was developed. The primary aim of this application is to serve as a bridge between volunteers and organisations by providing a unified platform where both parties can register, manage their profiles, and connect seamlessly. Volunteers can set their availability, display their skills and browse through upcoming events, whereas organisations can create events and specify their details, specify the number of volunteers required, and filter for candidates with relevant capabilities. By integrating these features, along with skill-matching and event management, the application seeks to reduce inefficiencies in volunteer recruitment and create more meaningful collaborations.

Hence, the proposed solution makes it so that both parties benefit from a more structured, accessible, and reliable system, ultimately fostering stronger community engagement and enabling a wider social impact.

**Summary Problem Definition:**

1. For Volunteers: Difficulty of finding opportunities that match their skills and availability
2. For Organisations: The challenge of identifying and recruiting suitable volunteers to support their events

**Objectives and Scope**

The development of the application was guided by the objective of creating a centralised, user-friendly platform that effectively bridges the gap between volunteers and organisations. To achieve this goal, the following sub-objectives were defined:

1. **Simply volunteer and registration management:** Provide an intuitive system where volunteers can create accounts, record personal details, list skills, and indicate availability
2. **Enable organisations to manage opportunities:** Allow organisations to register, create events, and specify requirements such as skills, location, date, and maximum number of volunteers
3. **Implement skill-based matching:** Introduce filtering mechanisms to connect volunteers with organisations and events that align with their strengths and interests
4. **Support event participation management:** Ensure organisations can track volunteer requests, approve/deny participation, and monitor number of attendees relative to event capacity
5. **Enhance communication through feedback mechanisms:** The use of flash messages & alerts, that keeps users informed of errors, confirmations, or important actions, ensuring smooth user experience
6. **Ensure data integrity and security:** Apply input validation, authentication, and database design principles to protect user information, maintain reliability of the system

The scope of the application was designed to focus on most critical functionalities needed for effective volunteer & organisation collaboration

* **Users and roles:** System supports two primary roles; Volunteers and Organisations, where volunteers can browse and join events, while organisations can create and manage them
* **Event management:** Organisations can create events with details (E.g. Title, Description, Date, Location, Required skills, Maximum volunteers)
* **Volunteer management:** Volunteers can update personal information, view all registered organisations, and request to participate in events
* **Filtering and matching:** Both volunteers & organisations can apply filters (E.g. By skills), to find the most relevant opportunities or candidates
* **Access controls:** Certain features are restricted to specific roles (E.g. Only organisations can view volunteer lists, only volunteers can view organisations), ensures fairness and relevance
* **Database integration:** Relational database stores information about users, events, skills, requests, ensuring consistency and scalability

However, scope intentionally excludes advanced features such as skill-proficiency-based matching, real-time managing, mobile application support, payment processing, or third-party integrations. These were deemed beyond the initial project’s requirements, and will be further discussed as potential areas for future development.

**Design Documents:**

Included are all the design documents from the initial planning stage of the project

**Final Tables Using Relational Notation:**

**User**(user\_id, email, password\_hash, phone\_number, role, created\_at)

**Volunteer**(volunteer\_id, *user\_id*, first\_name, last\_name, dob, availability)

**Organisation**(organisation\_id, *user\_id*, name, description, address, website\_url)

**Event**(event\_id, *organisation\_id*, title, description, event\_date, location, max\_volunteers)

**Skills**(skill\_id, name, description)

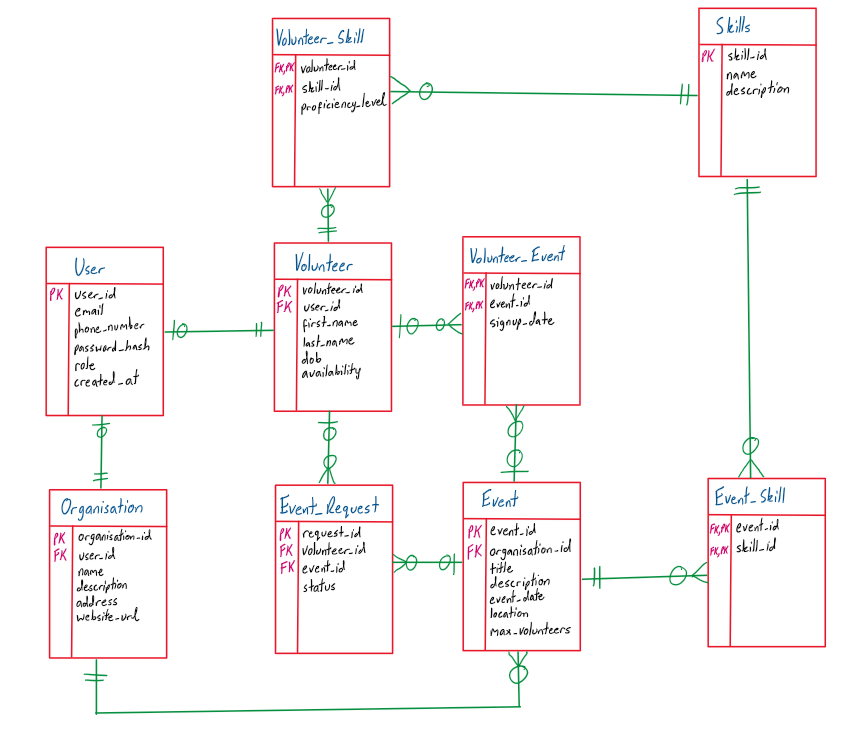
**Volunteer\_Event**(volunteer\_id, event\_id, signup\_date) (Composite primary key)

**Volunteer\_Skill**(volunteer\_id, skill\_id, proficiency\_level) (Composite primary key)

**Event\_Skill**(event\_id, skill\_id) (Composite primary key)

**Event\_Request(**request\_id, *volunteer\_id*, *event\_id*, status)

**Entity Relationship Diagram:**

****

**Data Dictionaries:**

| **User** | | | | |
| --- | --- | --- | --- | --- |
| **ElementName** | **DataType** | **Size** | **Description** | **Constraints** |
| user\_id | Integer |  | Unique identifier for each user | Primary Key  Autoincrement, Unique, Not Null |
| email | Text | 40 | User’s email address | Unique, Not Null |
| phone\_number | Text | 20 | User’s phone number | Optional |
| password\_hash | Text |  | Encrypted/Hashed password | Not Null |
| role | Text |  | Role of user (Either volunteer, or organisation) | Not Null |
| created\_at | Timestamp |  | When the account was created | Default: CURRENT\_TIMESTAMP |

| **Organisation** | | | | |
| --- | --- | --- | --- | --- |
| **ElementName** | **DataType** | **Size** | **Description** | **Constraints** |
| organisation\_id | Integer |  | Unique identifier for each organisation | Primary Key,  Autoincrement, Unique, Not Null |
| user\_id | Integer |  | Identifier for user account of organisation | Foreign Key -> User(User\_id),  Not Null, Unique |
| name | Text | 20 | Name of organisation |  |
| description | Text |  | Description of organisation | Optional |
| address | Text |  | Address of organisation | Optional |
| website\_url | Text |  | Website URL | Optional |

| **Volunteer** | | | | |
| --- | --- | --- | --- | --- |
| **ElementName** | **DataType** | **Size** | **Description** | **Constraints** |
| volunteer\_id | Integer |  | Unique identifier for each volunteer | Primary Key  Autoincrement, Unique, Not Null |
| user\_id | Integer |  | Identifier for user account of volunteer | Foreign Key -> User(user\_id),  Not Null, Unique |
| first\_name | Text | 20 | First name of volunteer | Not Null |
| last\_name | Text | 20 | Last name of volunteer | Not Null |
| dob | Datetime |  | Date of birth of volunteer | Not Null |
| availability | Text | 20 | Availability of volunteer (E.g. Weekdays, and their hours) | Optional |

| **Event** | | | | |
| --- | --- | --- | --- | --- |
| **ElementName** | **DataType** | **Size** | **Description** | **Constraints** |
| event\_id | Integer |  | Unique identifier for each event | Primary Key,  Autoincrement, Unique, Not Null |
| organisation\_id | Integer |  | Identifier for organisation hosting event | Foreign Key -> Organisation(organisation\_id),  Unique, Not Null |
| title | Text | 20 | Title of the event | Not Null |
| description | Text |  | Description of the event | Optional |
| event\_date | Timestamp |  | Scheduled date/time of event | Not Null |
| location | Text | 20 | Event location | Not Null |
| max\_volunteers | Integer |  | Maximum number of volunteers | Check if greater than 1  Optional |

| **Skill** | | | | |
| --- | --- | --- | --- | --- |
| **ElementName** | **DataType** | **Size** | **Description** | **Constraints** |
| skill\_id | Integer |  | Unique identifier for each skill | Primary Key,  Unique, Not Null |
| name | Text | 20 | Name of each skill | Not Null |
| description | Text |  | Description of each skill | Optional |

| **Volunteer\_Skill** | | | | |
| --- | --- | --- | --- | --- |
| **ElementName** | **DataType** | **Size** | **Description** | **Constraints** |
| volunteer\_id | Integer |  | Unique identifier for each volunteer | Primary Key, Foreign Key -> Volunteer(volunteer\_id)  Not Null |
| skill\_id | Integer |  | Identifier for the relevant skill of that volunteer | Primary Key, Foreign Key -> Skills(skill\_id)  Not Null |
| volunteer\_proficiency\_level | Text | 20 | Proficiency rating of the volunteer for that skill | Optional |

| **Volunteer\_Event** | | | | |
| --- | --- | --- | --- | --- |
| **ElementName** | **DataType** | **Size** | **Description** | **Constraints** |
| volunteer\_id | Integer |  | Unique identifier for each volunteer | Primary Key, Foreign Key -> Volunteer(volunteer\_id),  Not Null |
| event\_id | Integer |  | Identifier for the event the volunteer is a part of | Primary Key, Foreign Key -> Event(event\_id),  Not Null |
| signup\_date | Timestamp |  | The date when volunteer signed up | Default: CURRENT\_TIMESTAMP |

| **Event\_Skill** | | | | |
| --- | --- | --- | --- | --- |
| **ElementName** | **DataType** | **Size** | **Description** | **Constraints** |
| event\_id | Integer |  | Unique identifier for the event | Primary Key, Foreign Key -> Event(event\_id),  Not Null |
| skill\_id | Integer |  | Identifier for the skills required for that event | Primary Key, Foreign Key -> Skills(skill\_id),  Not Null |

| **Event\_Request** | | | | |
| --- | --- | --- | --- | --- |
| **ElementName** | **DataType** | **Size** | **Description** | **Constraints** |
| request\_id | Integer |  | Unique identifier for each request | Primary Key, Not Null |
| volunteer\_id | Integer |  | Identifier for the volunteer making the request | Foreign Key -> Volunteer(volunteer\_id),  Not Null |
| event\_id | Integer |  | Identifier for the event the volunteer is making a request to join | Foreign Key -> Event(event\_id),  Not Null |
| status | Text | 20 | Status of sign up | Default: ‘pending’,  Not Null |

**Normalisation:**

**Unnormalised Form (0NF):**

* Is the single table, with everything in it, but only in relational notation, calling it ‘Community\_Connect’

**Issues:**

* Non-Atomic values, where the skills\_required lists all the skills the volunteer needs (Hence is 1NF)
* Organisation details are repeated for every event, and volunteer information repeated for every signup, so data is redundant
* Update Anomaly Example: Updating an organisation’s phone number requires updating it in all of the records
* Insert Anomaly Example: Can’t add a new volunteer until they sign up for an event
* Delete Anomaly Example: Deleting an event of an organisation, if it is their only event, wipes out the entire organisation
* Table also has partial and transitive dependencies, but talked about more in 2NF and 3NF

**Community\_Connect**(user\_id, email, password\_hash, role, created\_at,

volunteer\_id, volunteer\_first\_name, volunteer\_last\_name, volunteer\_email, volunteer\_phone\_number, volunteer\_availability, volunteer\_required\_skills, volunteer\_location, signup\_date

organisation\_id, organisation\_name, organisation\_description, organisation\_contact\_email, organisation\_phone\_number, organisation\_address, organisation\_website\_url,

event\_id, event\_title, event\_description, event\_date, event\_location, max\_volunteers)

**First-Normal Form (1NF):**

* All values are now atomic, with no lists or nested data, where repeating groups are eliminated
* No actual change in the relational notation of the tables, but would see change with actual data, where for example, every skill required would be on separate row

**Issues:**

* Primary key is no longer unique, so is no longer a “primary” key
* The non-key attributes do not depend on the whole primary key, so need to split them up into their separate tables, and hence create new tables for the fields that are partially dependent
* Data is redundant, where organisation information, volunteer information, event information repeated for each required skill of the volunteer

**Second-Normal Form (2NF):**

* Tables are now split up, and all partial dependencies are removed

**Issues:**

* A lot of transitive dependencies, mainly with the email and phone\_number, where, for example, the volunteer\_email, and the organisation\_email should depend on the user\_id, as they are the email of the user
* Need to also resolve the many-to-many relationships using junction tables

**User**(user\_id, password\_hash, role, created\_at)

**Volunteer**(volunteer\_id, *user\_id*, volunteer\_first\_name, volunteer\_last\_name, volunteer\_email, volunteer\_phone\_number, volunteer\_availability, volunteer\_location)

**Organisation**(organisation\_id, *user\_id*, organisation\_name, organisation\_description, organisation\_email, organisation\_phone\_number, organisation\_address, organisation\_website\_url)

**Event**(event\_id, *organisation\_id*, event\_title, event\_description, event\_date, event\_location, max\_volunteers)

**Skills**(skill\_id, skill\_name, skill\_description)

**Third-Normal Form (3NF):**

* No more transitive dependencies in the data
  + Moved email to the user table, and hence avoided repeating it in Organisation and Volunteer entities
* Also moved phone\_number to the user table, and hence avoided repeating it in the Organisation and Volunteer entities

**Issues:**

* Need to also resolve the many-to-many relationships using junction tables, but is a part of database design and not normalising the database
  + Will be later resolved in the database design phase

**User**(user\_id, email, password\_hash, phone\_number, role, created\_at)

**Volunteer**(volunteer\_id, *user\_id*, volunteer\_first\_name, volunteer\_last\_name, volunteer\_availability, volunteer\_location)

**Organisation**(organisation\_id, *user\_id*, organisation\_name, organisation\_description, organisation\_address, organisation\_website\_url)

**Event**(event\_id, *organisation\_id*, event\_title, event\_description, event\_date, event\_location, max\_volunteers)

**Skills**(skill\_id, skill\_name, skill\_description)

**Initial SQL Scripts:**

**SQL Commands to Create Tables:**

CREATE TABLE IF NOT EXISTS user (

user\_id INTEGER PRIMARY KEY AUTOINCREMENT,

email TEXT NOT NULL UNIQUE,

password\_hash TEXT NOT NULL,

phone\_number TEXT,

role TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE IF NOT EXISTS volunteer (

volunteer\_id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER NOT NULL,

first\_name TEXT NOT NULL,

last\_name TEXT NOT NULL,

dob DATE NOT NULL,

availability TEXT,

FOREIGN KEY (user\_id) REFERENCES user(user\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS organisation (

organisation\_id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER NOT NULL,

name TEXT NOT NULL,

description TEXT,

address TEXT,

website\_url TEXT,

FOREIGN KEY (user\_id) REFERENCES user(user\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS event (

event\_id INTEGER PRIMARY KEY AUTOINCREMENT,

organisation\_id INTEGER NOT NULL,

title TEXT NOT NULL,

description TEXT,

event\_date TIMESTAMP NOT NULL,

location TEXT, max\_volunteers INTEGER,

FOREIGN KEY (organisation\_id) REFERENCES organisation(organisation\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS skill (

skill\_id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL UNIQUE,

description TEXT

);

CREATE TABLE IF NOT EXISTS volunteer\_event (

volunteer\_id INTEGER NOT NULL,

event\_id INTEGER NOT NULL,

signup\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (volunteer\_id, event\_id),

FOREIGN KEY (volunteer\_id) REFERENCES volunteer(volunteer\_id) ON DELETE CASCADE,

FOREIGN KEY (event\_id) REFERENCES event(event\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS volunteer\_skill (

volunteer\_id INTEGER NOT NULL,

skill\_id INTEGER NOT NULL,

volunteer\_proficiency\_level TEXT,

PRIMARY KEY (volunteer\_id, skill\_id),

FOREIGN KEY (volunteer\_id) REFERENCES volunteer(volunteer\_id) ON DELETE CASCADE,

FOREIGN KEY (skill\_id) REFERENCES skill(skill\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS event\_skill (

event\_id INTEGER NOT NULL,

skill\_id INTEGER NOT NULL,

PRIMARY KEY (event\_id, skill\_id),

FOREIGN KEY (event\_id) REFERENCES event(event\_id) ON DELETE CASCADE,

FOREIGN KEY (skill\_id) REFERENCES skill(skill\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS event\_request (

request\_id INTEGER PRIMARY KEY AUTOINCREMENT,

volunteer\_id INTEGER NOT NULL,

event\_id INTEGER NOT NULL,

status TEXT DEFAULT 'pending', -- Pending / Accepted / Declined

FOREIGN KEY (volunteer\_id) REFERENCES volunteer(volunteer\_id) ON DELETE CASCADE,

FOREIGN KEY (event\_id) REFERENCES event(event\_id) ON DELETE CASCADE,

UNIQUE(volunteer\_id, event\_id) -- prevent duplicate requests

);

**SQL Commands to Insert Example Data Into Tables:**

**Inserting Into Users:**

INSERT INTO user (email, password\_hash, phone\_number, role) VALUES ('alice@example.com', 'hashed\_pw1', '1234567890', 'volunteer');

INSERT INTO user (email, password\_hash, phone\_number, role) VALUES ('bob@example.com', 'hashed\_pw2', '1234567891', 'volunteer');

INSERT INTO user (email, password\_hash, phone\_number, role) VALUES ('carol@example.com', 'hashed\_pw3', '1234567892', 'volunteer');

INSERT INTO user (email, password\_hash, phone\_number, role) VALUES ('org1@example.com', 'hashed\_pw4', '9876543210', 'organisation');

INSERT INTO user (email, password\_hash, phone\_number, role) VALUES ('org2@example.com', 'hashed\_pw5', '9876543211', 'organisation');

**Inserting Into Volunteers:**

INSERT INTO volunteer (user\_id, first\_name, last\_name, dob, availability) VALUES (1, 'Alice', 'Smith', '1998-05-20', 'Weekends');

INSERT INTO volunteer (user\_id, first\_name, last\_name, dob, availability) VALUES (2, 'Bob', 'Johnson', '2000-03-14', 'Weekdays');

INSERT INTO volunteer (user\_id, first\_name, last\_name, dob, availability) VALUES (3, 'Carol', 'Davis', '1995-12-01', 'Flexible');

**Inserting Into Organisations:**

INSERT INTO organisation (user\_id, name, description, address, website\_url) VALUES (4, 'Helping Hands', 'Community support and outreach', '123 Main St', '<http://helpinghands.org>');

INSERT INTO organisation (user\_id, name, description, address, website\_url) VALUES (5, 'Green Earth', 'Environmental conservation projects', '456 Park Ave', '<http://greenearth.org>');

**Inserting Into Events:**

INSERT INTO event (organisation\_id, title, description, event\_date, location, max\_volunteers) VALUES (1, 'Food Drive', 'Distribute food to families in need', '2025-09-15 10:00:00', 'Community Center', 10);

INSERT INTO event (organisation\_id, title, description, event\_date, location, max\_volunteers) VALUES (2, 'Tree Planting', 'Planting trees in the city park', '2025-09-20 09:00:00', 'Central Park', 15);

INSERT INTO event (organisation\_id, title, description, event\_date, location, max\_volunteers) VALUES (2, 'Recycling Workshop', 'Teach recycling and waste management', '2025-09-25 14:00:00', 'Library Hall', 20);

**Inserting Into Volunteer Skills:**

INSERT INTO volunteer\_skill (volunteer\_id, skill\_id, volunteer\_proficiency\_level) VALUES (1, 1, 'Intermediate'); -- Alice has Endurance

INSERT INTO volunteer\_skill (volunteer\_id, skill\_id, volunteer\_proficiency\_level) VALUES (1, 2, 'Advanced'); -- Alice has Listening

INSERT INTO volunteer\_skill (volunteer\_id, skill\_id, volunteer\_proficiency\_level) VALUES (2, 3, 'Beginner'); -- Bob is Talkative

INSERT INTO volunteer\_skill (volunteer\_id, skill\_id, volunteer\_proficiency\_level) VALUES (3, 4, 'Advanced'); -- Carol good at Public Speaking

**Inserting Into Event Skills:**

INSERT INTO event\_skill (event\_id, skill\_id) VALUES (1, 2); -- Food Drive needs Listening

INSERT INTO event\_skill (event\_id, skill\_id) VALUES (2, 1); -- Tree Planting needs Endurance

INSERT INTO event\_skill (event\_id, skill\_id) VALUES (3, 4); -- Recycling Workshop needs Public Speaking

**Inserting Into Event Requests:**

INSERT INTO event\_request (volunteer\_id, event\_id, status) VALUES (1, 1, 'pending'); -- Alice requests Food Drive

INSERT INTO event\_request (volunteer\_id, event\_id, status) VALUES (2, 2, 'accepted'); -- Bob joins Tree Planting

INSERT INTO event\_request (volunteer\_id, event\_id, status) VALUES (3, 3, 'pending'); -- Carol requests Recycling Workshop

**Inserting The Actual Skills:**

INSERT OR IGNORE INTO skill (name, description) VALUES (?, ?) ''', [

("Endurance", "Ability to sustain effort for long periods"),

("Listening", "Good at understanding and following others"),

("Talkative", "Engages easily in conversation"),

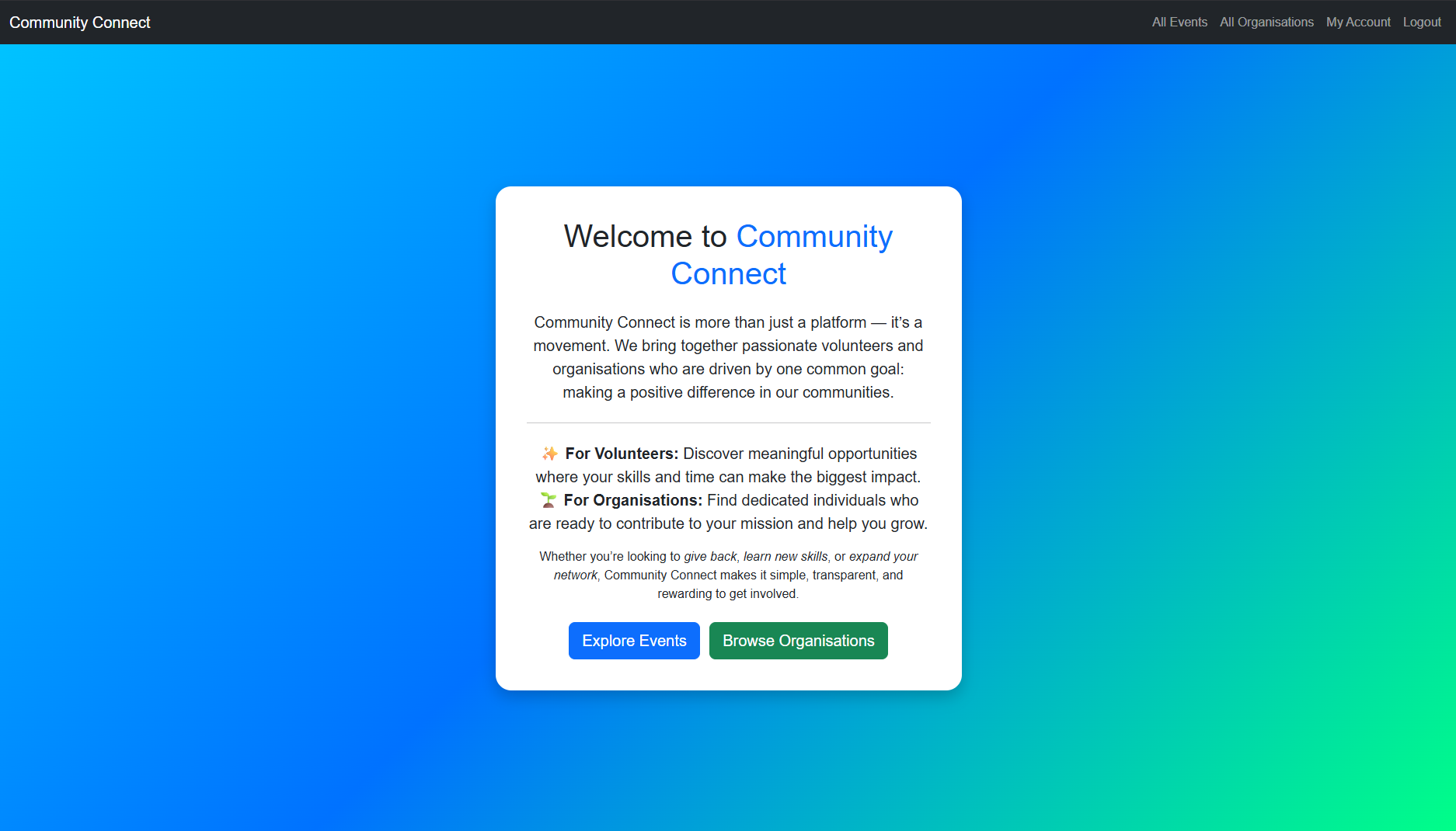
("Public Speaking", "Confident in speaking to groups")

])

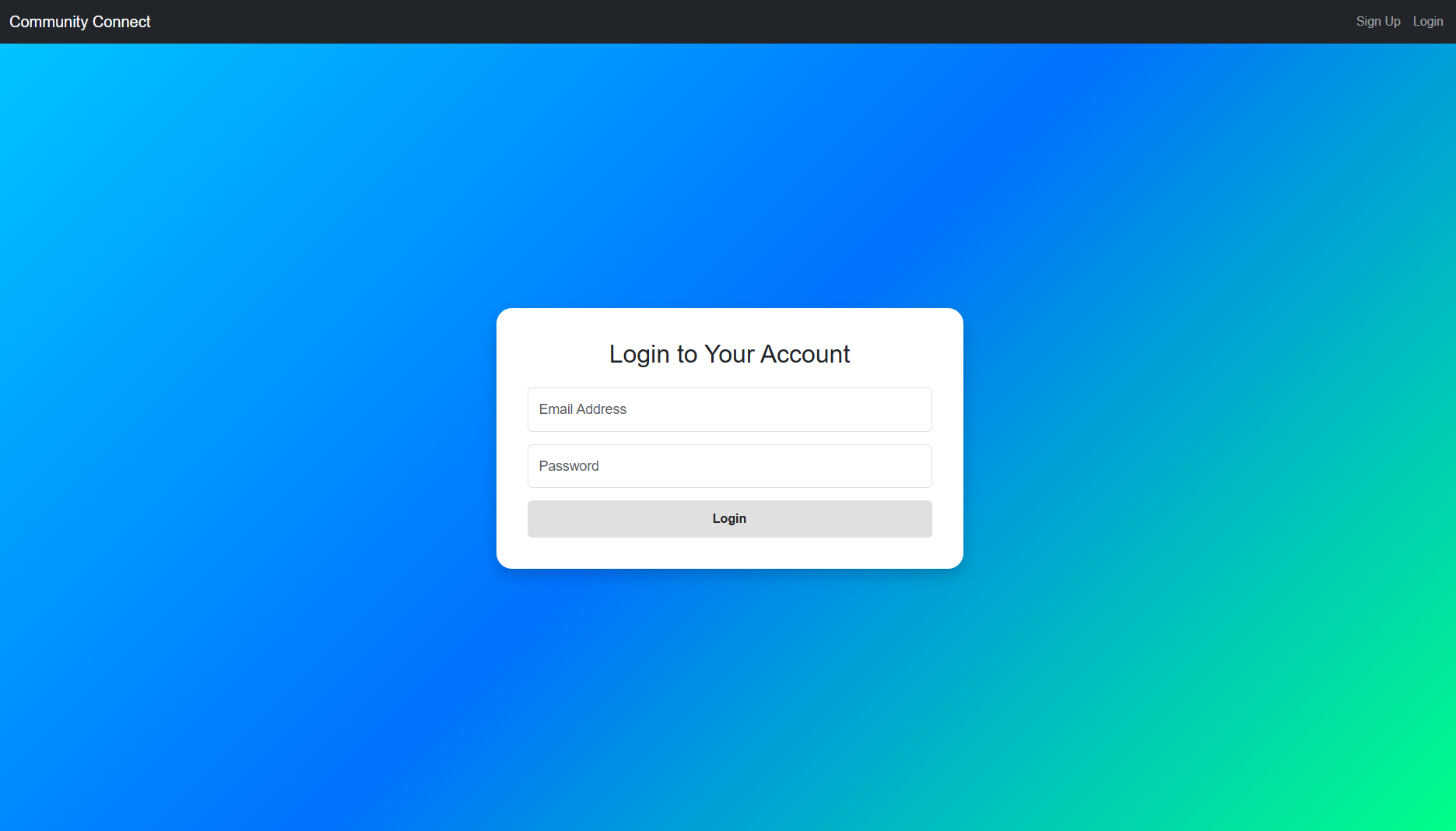
* Is a pythonic way of doing it

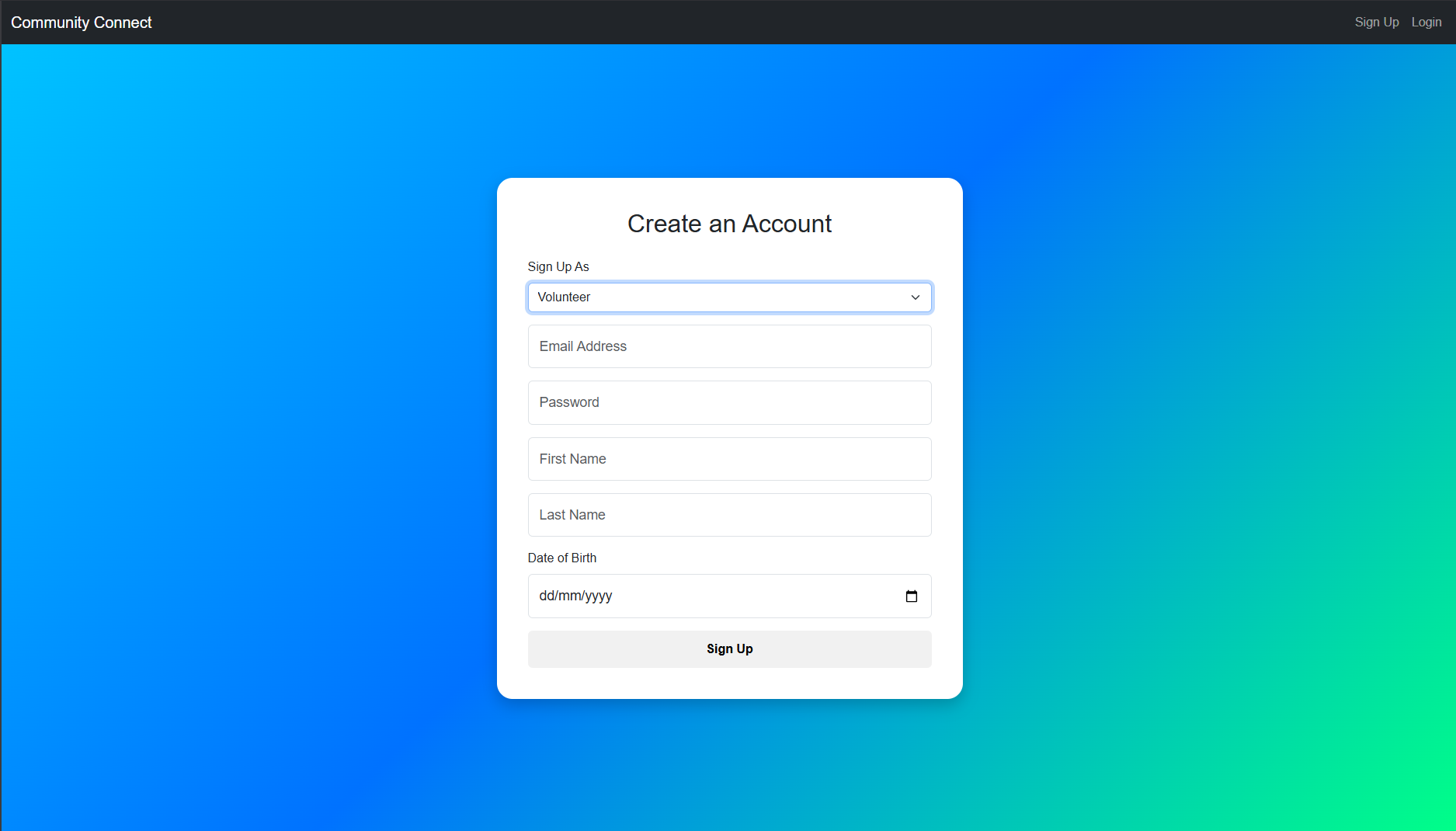
**Screen Captures of Final Application:**

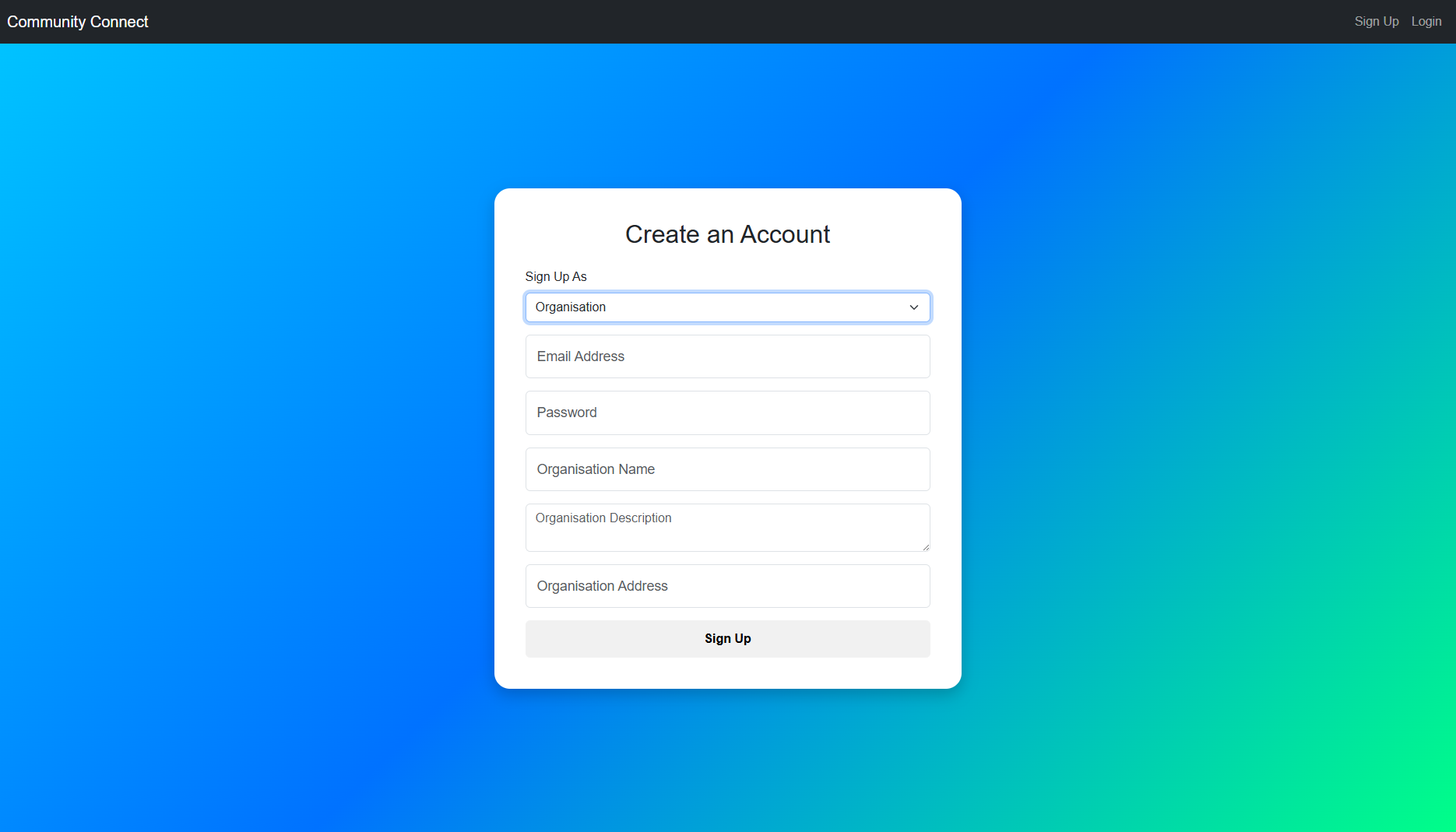
**Home Page:**



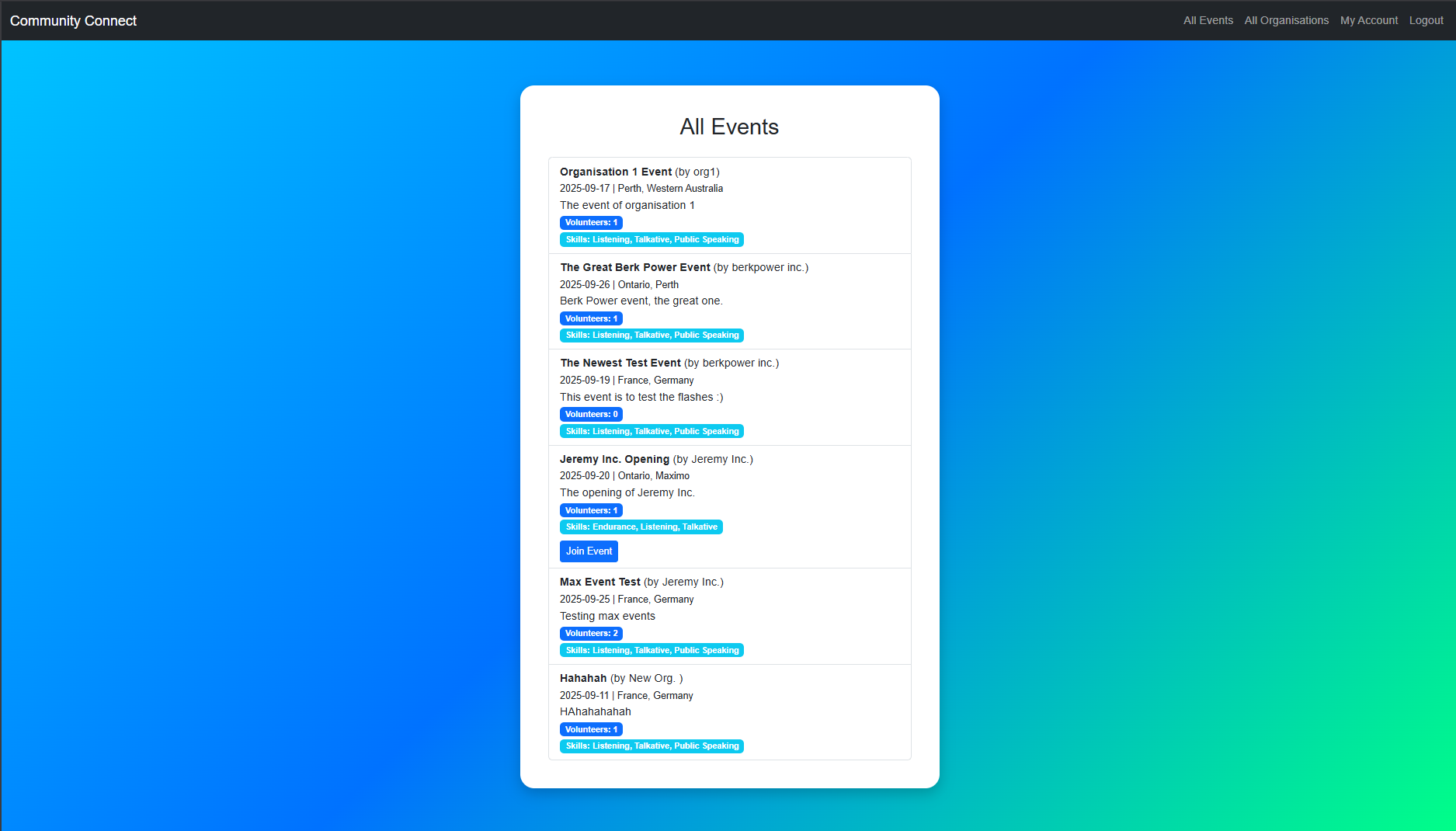
**Login and Register Pages:**

****

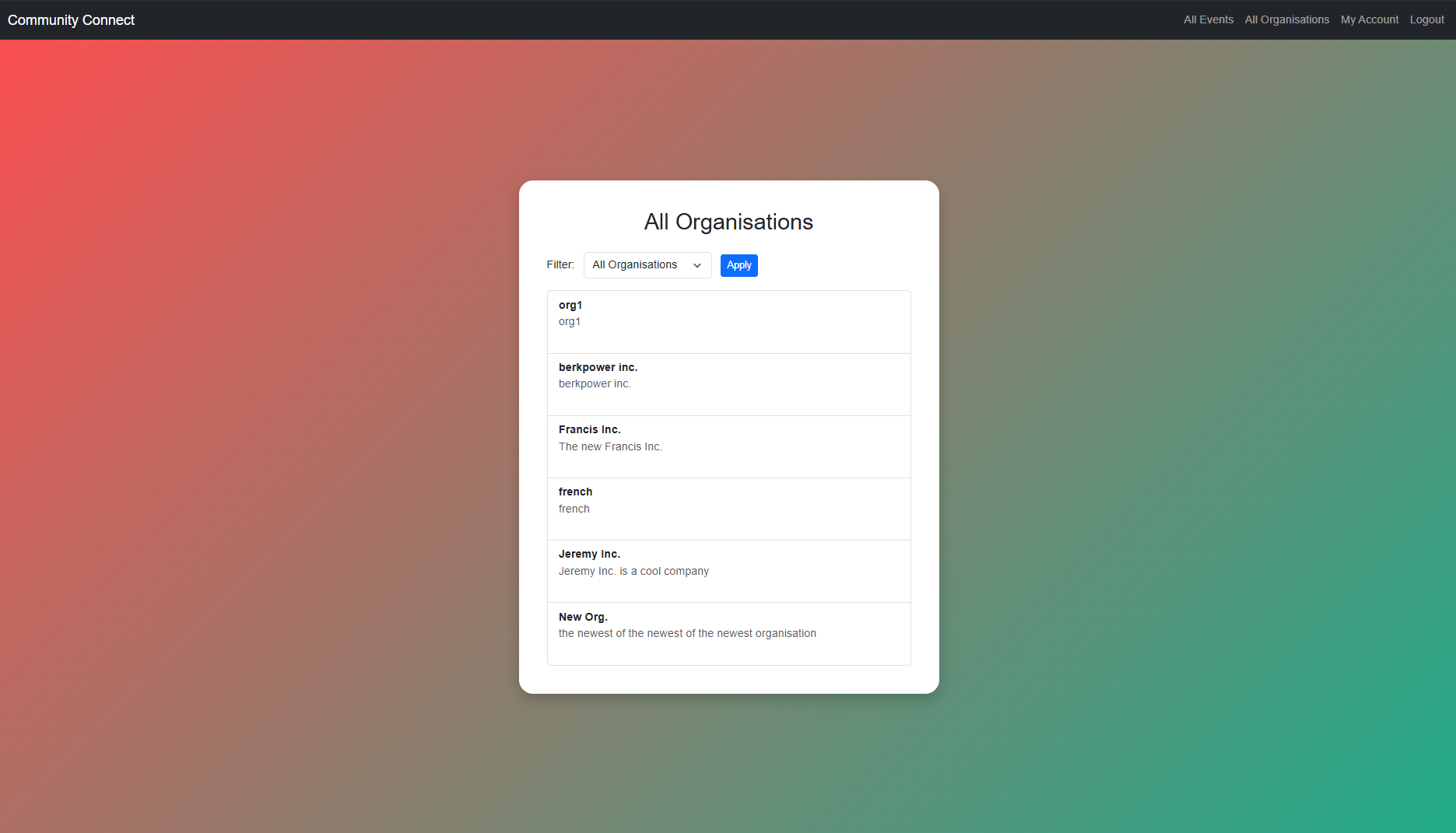


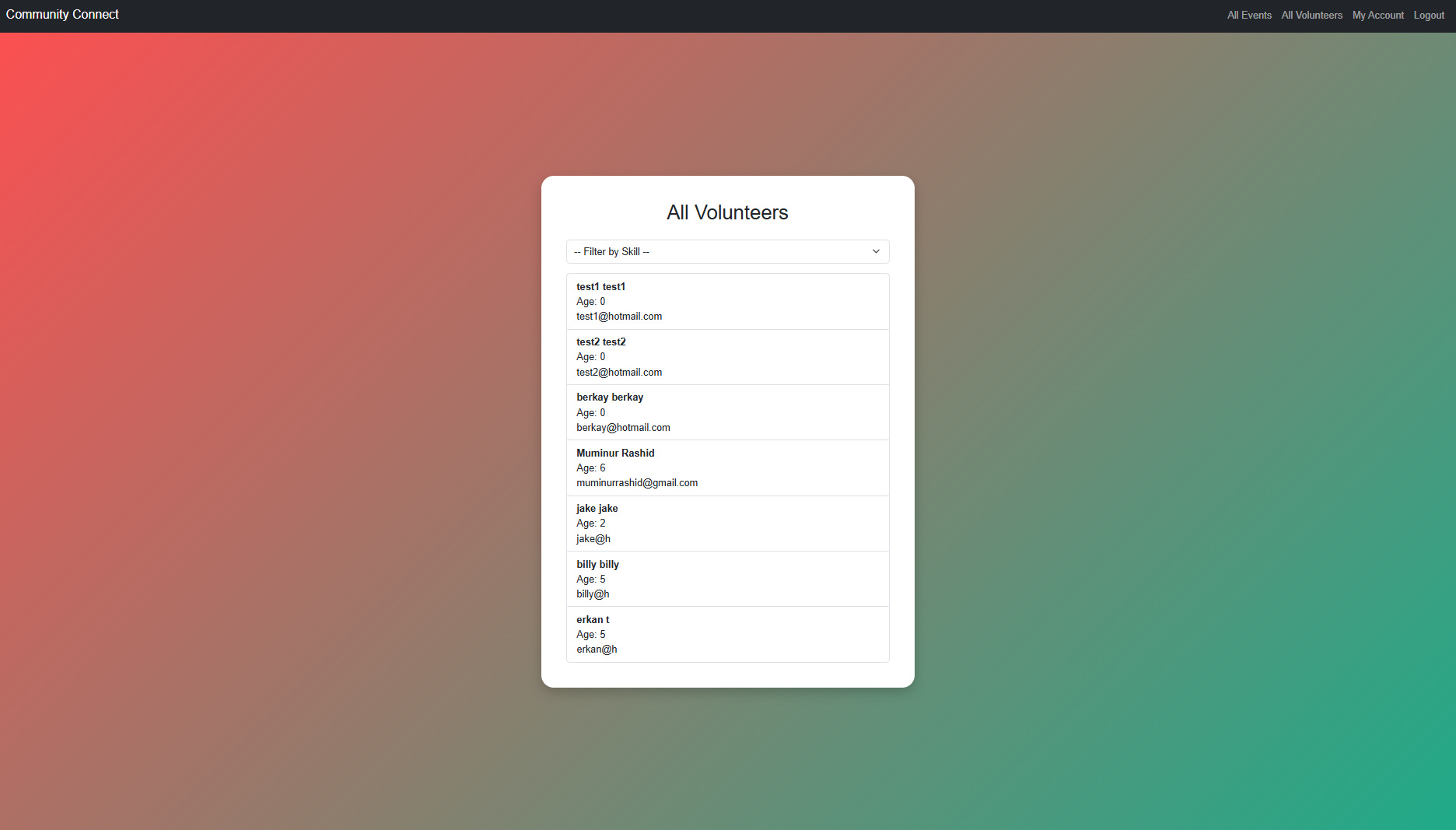


**All Events Page:**

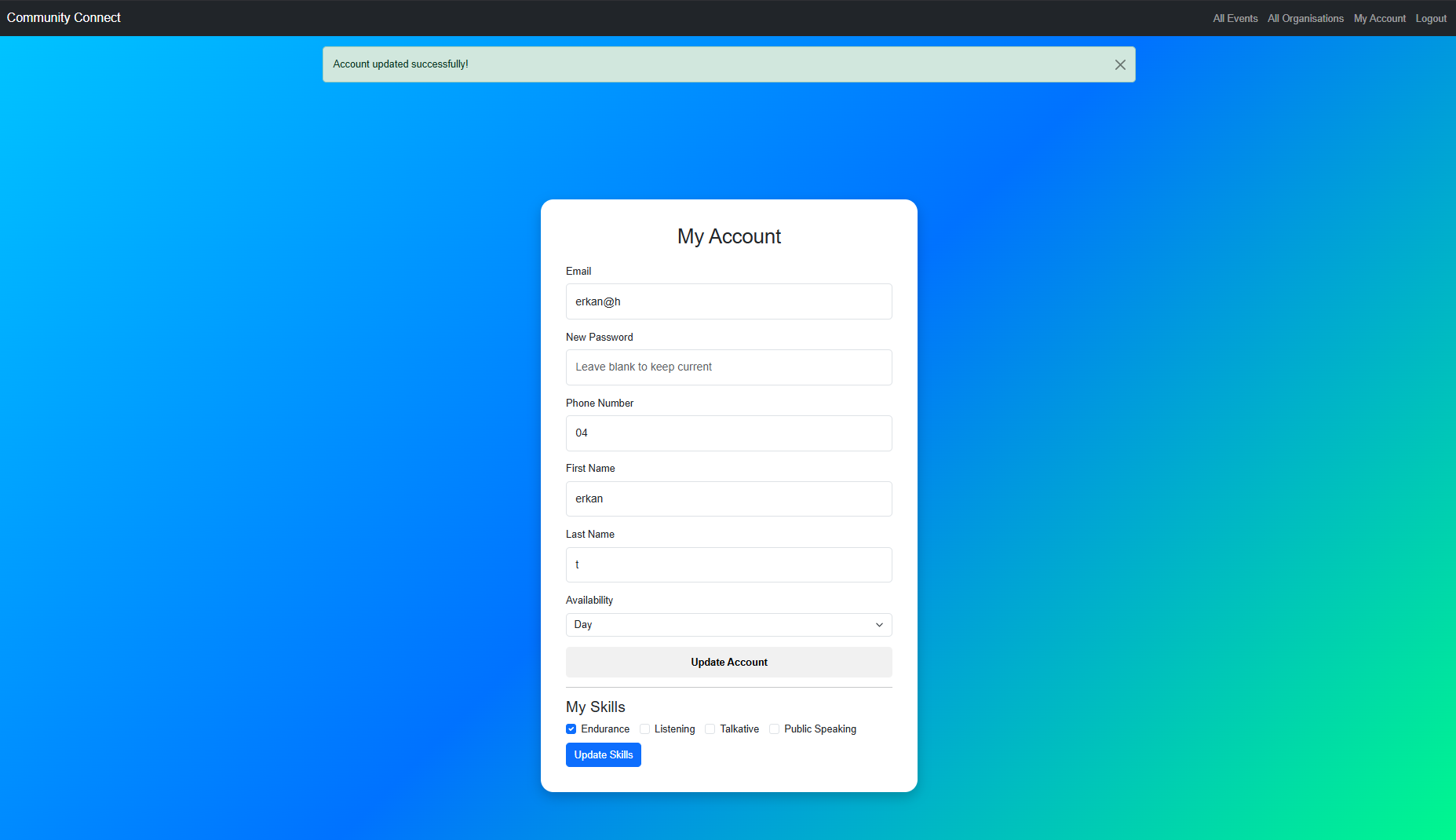
****

**All Organisations and Volunteers Pages:**

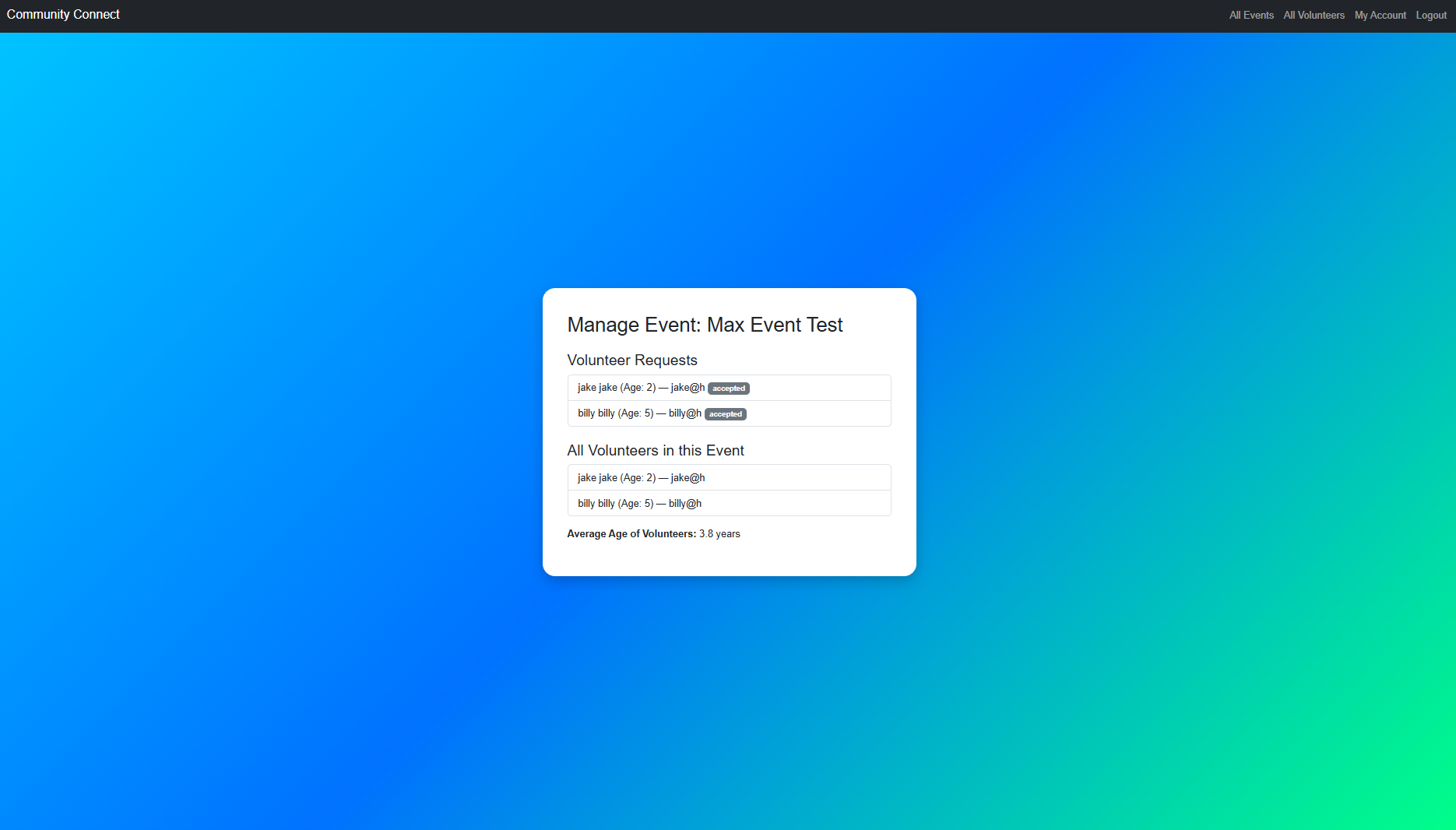
****

****

**Account Update Page:**

****

**Manage Event Page:**

****

**Data Quality and Cleaning:**

Quality of data in the system is influenced by several factors, including currency (Which is whether the information stored by the database is up-to-date, and reflects most recent state of the real world), authenticity (Ensuring that users provide authentic data, provided it comes from credible & verifiable sources, and hasn’t been tampered with), relevance (Data is relevant if it directly pertains to the question or task at hand), and accuracy (Ensuring data is accurate, where it is if it correctly represents the real-world values it’s supposed to represent.

* Currency is important for checking if the volunteer’s availability and contact details are up to date, where if it isn’t, it may cause miscommunication between volunteers and organisations
* Authenticity is important for checking that the user provides genuine personal details rather than fake or duplicate accounts
* Relevance is important, where the application only collects information that is necessary for matching volunteers to organisations and events
* Accuracy is important, where the application ensures that data such as date of birth, email addresses, and event dates are valid

To maintain clean data, validation rules are implemented at the form level, where valid e-mail formats are checked, and unique constraints are enforced to ensure that duplicate registrations are not performed (These are only some examples)

As for outliers, such as an invalid date of birth (E.g. A future date or year that makes a volunteer over 120 years old), can be handled by input validation\ at entry, and periodic database audits. Similarly, constraints like limiting the maximum number of skills selected for an event, and enforcing the unique e-mail addresses improve the reliability of the stored data.

**Evaluation:**

The final product successfully meets the initial requirements and sub-objectives of creating a platform where volunteers & organisations can register, manage events, and be matched based on relevant skills. Core features such as user authentication, event creation, skill-based filtering, and request management were implemented and function as intended.

Though the project does include some limitations:

* The interface is functional, but lacks advanced user experience features such as a real-time search bar, notifications, or optimisations for mobile support
* The system relies on manual validation of authenticity, meaning fake accounts could still be created
* Where skill matching is available, it is basic, and does not account for proficiency levels or more advanced recommendation algorithms
* The CSS used in the project is not as advanced, where fade screens and such are not implemented, which would’ve enhanced the user experience.

Hence, future improvements that is looked into could include:

* Automated e-mail verification, where along with checking e-mail format where it currently does, the verification algorithm could check other properties such as date of creation of the e-mail, to verify if the account created is a bot account or not
* Profile completion tracking, where currently there is no warning or notification asking the user to complete their profile (E.g. Selecting their skills), so a profile completion tracker would enhance user experience
* Better search and filtering tools, such as a proper search bar where user can type in the skill they are looking for, with auto-complete features integrated as well
* Expanding reporting and analytics for organisations would be a strong addition, such as volunteer participation history
* Integrating third-party services using external APIs (E.g. Google Maps for event locations)
* Mobile-friendly design to improve accessibility and functionality
* Could be scaled for more than just a volunteering system, where it could be an app to track actual work and employment, such as including payment processing, to pay employees

**Development Issues:**

**Ethical Issues:**

Data ethics go beyond the legal compliance, where it is about doing what is right when collecting, storing, and using personal data

* **Collecting Data About Individuals:** Consent is essential, where users must know what their personal data is being collected for and its purpose
  + Application ensures that data is gathered only for legitimate purposes (E.g. Account creation, Event management), and data minimisation is applied to avoid collecting unnecessary information
  + Sensitive information like dates of birth is stored securely, and only used for relevant functions (E.g. Calculating age)
  + Anonymisation could be explored further, to prevent data being traced back to individuals if datasets are exported
* **Privacy Concerns:** Respecting individuals’ right to control their personal data is critical
  + The system prevents surveillance or unauthorised monitoring, where measures are taken to avoid risks of data breaches (E.g. Passwords are hashed, Access restrictions prevent users from viewing unauthorised data)
* **Appropriate Use of Data:** Data collected for event matching must not be reused for unrelated purposes without explicit consent
  + Fairness and bias are considered when designing skill-based matching, so that no volunteer group is unfairly excluded
* **Use of Data Mining:** While data mining could enhance the volunteer-event matching in the future (Such as profiling skills, and predicting availability), safeguards must ensure profiling does not become intrusive or discriminatory, and consent must be first gained from the users
  + Ethical use of predictive analysis is essential to avoid the negative real-world consequences

**Legal Issues:**

The project complies with the Australian Privacy Act 1988, and the Australian Privacy Principles (APPs) enforced by the Office of the Australian Information Commissioner (OAIC)

**APP5 - Notification of Collection of Personal Information:**

* Organisation must take reasonable steps to notify individual about what personal information they might be collecting and why
* System provides clear communication during registration about why personal data (E.g. E-Mail, Date of Birth, Skills) is collected, and how it will be used

**APP8 - Cross-Border Disclosure of Personal Information:**

* Organisation must take reasonable steps to ensure that any overseas recipient of personal information complies with APPs (Crucial for data on international servers)
* If system was deployed on international servers, safeguards would be required to ensure overseas providers follow the APPs

**APP10 - Quality of Personal Information:**

* Organisation must take reasonable steps to ensure personal information collected is accurate, up-to-date, complete
* Data validation checks (E.g. Unique e-mails, Valid dates, Non-Empty names), help ensure accuracy, completeness, and currency of user data

**APP11 - Security of Personal Information:**

* Organisation must take reasonable steps to protect personal information from misuse, interference, unauthorised access, modification, disclosure
  + Must also have safe destruction of data it no longer uses
* Technical safeguards, such as password hashing, restricted access, and database integrity constraints, reduce the risks of misuse, interference, or unauthorised access
  + Safe deletion protocols should be added for when data is no longer required

**APP12 - Access to Personal Information:**

* Organisation must give individual access for personal information upon request
  + Gives individuals right to see what an organisation holds about them

* Ensures transparency, allows individuals to check for accuracy and request corrections
* Users are able to view and update their own data via the ‘Manage Account’ page (E.g. Contact details, Availability), this transparency ensures accuracy, and empowers users to control their personal information

**Security Issues:**

Security is crucial in maintaining trust and protecting personal data

* Data security is the technical and procedural protection of data, secure data can still violate privacy if used inappropriately
* **Keeping Personal Data Private:** Measures such as encryption, access controls, and anonymisation, protect against misuse even if data is accessed improperly. Privacy and security are treated as distinct, but interdependent
* **Backups of Organisational Data:** Backups are copies of data made to restore it in the event of data loss
  + Where in cases such as accidental deletions, or disasters (E.g. Cyberattacks like ransomware, Natural disasters, Hardware failure), backups ensure business continuity, where the platform can quickly recover and continue its operations after the incident
* Both on-site and off-site / cloud backups are used in relation to user data
* **Restricting Access to Data:** Access controls are security techniques that regulate who or what can view/use resources in a computing environment
  + Authentication (Through e-mail and password) is used to verify the user identity, where authorisation follows from this, granting specific permissions to specific users based on their verified identity (E.g. Volunteers can see all organisations, and Organisations can see all volunteers)
  + Principle of least privilege is followed, where users are granted a minimum level of access necessary to perform their jobs
* **Ownership and Control of Data:** Individuals have the fundamental right to ownership of their own personal data, where the question is who decides how data is used, stored, and shared?
  + Platform is designed to respect this, where data is stored for as long as it is needed under the control of the user, and is then deleted afterwards (Through a structured approach of verify -> soft deletion -> physical deletion)

**In Summary:**

In developing this project, a functional web application that connects volunteers with organisations in need has been developed, addressing the initial problem of streamlining community engagement. By designing a relational database, implementing user authentication, and creating event creation and skill-matching, the system demonstrates both technical feasibility and social impact.

Careful consideration was given to data quality, ethical responsibilities, legal compliance under the APPs, and robust security measures to ensure user trust.

While limitations exist, such as scalability, more advanced matching algorithms, and enhanced user interfaces, the platform establishes a strong foundation for user development. Ultimately, the project highlights the potential of technology to strengthen community connections, empowering individuals to contribute their skills, while enabling organisations to achieve meaningful outcomes.

This project was very fun to make, and I hope this project scales in the future!

Thanks,

* Berkay Topal