**<Community Service Volunteer Tracking System>**

Software Requirements Specification

Version <1.0>

**MATHEBULA, NN (47489189)**

**SHEZI, P (45225524)**

**KONDILE, SESETHU (41590503) MZOBE, BAYA (44027516)**

**NYEMBEZI, T (40183955)**

**SEDZE, TK (52827097)**

**KUMANE, QN (45097208)**

**Submitted in partial fulfillment of the requirements of**

**CMPG 224, Software Engineering**

**30/08/2025**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <22/08/25> | <1.0> | SRS 1.0 | Group 6 |
| <xx/xx/xx> | <2.0> | SRS 2.0 |  |
| <30/08/25> | <3.0> | SDD 3.0 |  |

Table of Contents

Introduction 1.0 5

Purpose 1.1 5

Scope 1.2 5

system overview 1.3 6

Definitions and Acronyms 1.4 6

References 1.5 6

ARCHITECTURAL DESIGN 2.0 6

Architectural diagrams 2.1 6

Descriptions Detailed Design 3.0 7

User interface(UI) 3.1 7

Designs 3.2 8

Data design 3.3 8

Component/Module Design 3.4 8

USE CASE REALIZATION 4.0 7

Use case diagram 3.5 8

Use case descriptions 3.6 8

FUNCTIONAL Requirements 4.0 9

F & FR 3.1 9

F & FR 3.2 9

F & FR 3.3 9

F & FR 3.4 10

F & FR 3.5 10

F & FR 3.6 #

F & FR 3.7 #

Non-functional Requirements 5.0 11

Usability 5.1 11

Performance 5.2 11

Security 5.3 11

Reliability 5.4 11

Summary 5.0 12

etc.

aPPENDIX 8.0 17

1.0 Introduction

The Community Service Volunteer Tracking System (CSVTS) is a system designed to help communities organize and manage their volunteer activities. Communities often face challenges when keeping track of volunteers, planning events, and making sure the right people are assigned to the right tasks. This system aims to solve those problems by providing one place where everything can be managed. The main users of the system will be administrators, who set up events and manage volunteers, and the volunteers, who sign up and take part in activities. The system will make it easier to coordinate people and events, save time, and reduce mistakes compared to using manual methods.

* 1. **Purpose**

The way that organizations, coordinators, and volunteers interact and work together to assist their communities is being completely transformed by the Community Service Volunteer Tracking System (CSVTS). CSVTS keeps volunteers motivated and involved by offering a smooth, centralized platform that makes it simple for them to sign up, track their hours, and find worthwhile service opportunities. The system provides administrators with tools to monitor participation, validate contributions, and produce informative reports that promote efficiency and transparency. In short, the system is to make volunteer management simpler, faster, and more reliable for both administrators and volunteers.

* 1. **Scope**

The goal of the Community Service Volunteer Tracking System (CSVTS) is to streamline and improve volunteer engagement management. Important tasks like volunteer registration, precise service hour recording, project participation monitoring, and timely reminders to keep everyone informed and engaged are all streamlined by it. Its main objective is to promote accountability, openness, and operational effectiveness so that organizations can more effectively manage their volunteer programs and increase their social impact.

What the system will include:

* Volunteer registration and profile management.
* Event creation and scheduling by administrators.
* Assigning volunteers to events.
* Viewing volunteer lists and event schedules.
* Basic reporting to track attendance and participation.

What the system will not include:

* Managing finances such as donations or payments.
* Integration with outside systems like social media.
* Advanced reporting beyond simple participation tracking.

The system is focused only on solving the core problem of volunteer and event management, without extra features that may complicate it.

* 1. **System Overview**

The CSVTS will be a simple system with the following parts:

* A user interface where volunteers and administrators can interact with the system.
* A business logic module that handles all the rules, like assigning volunteers to events.
* A data management module that stores information about volunteers and events in files or a database.

Volunteers will use the system to sign up and see what events are available.  
Administrators will use the system to add events, manage volunteers, and keep records of who has participated. The main goal is to make volunteer management easier, reduce errors, and provide a working system that demonstrates good software engineering practices.

**1.4 Definitions and Acronyms**

Concept Definition

CSVTS Community Service Volunteer Tracking System

UI User Interface

**1.5 References**

Volunteer Matters – Volunteer Management Software  
 <https://www.volunteermatters.com>

Galaxy Digital – How to Choose Volunteer Management Software <https://www.galaxydigital.com/blog/how-to-choose-a-volunteer-management-software>

Track It Forward – Volunteer Time Tracking

<https://www.trackitforward.com>

Forecast – What is Volunteer Time Tracking?

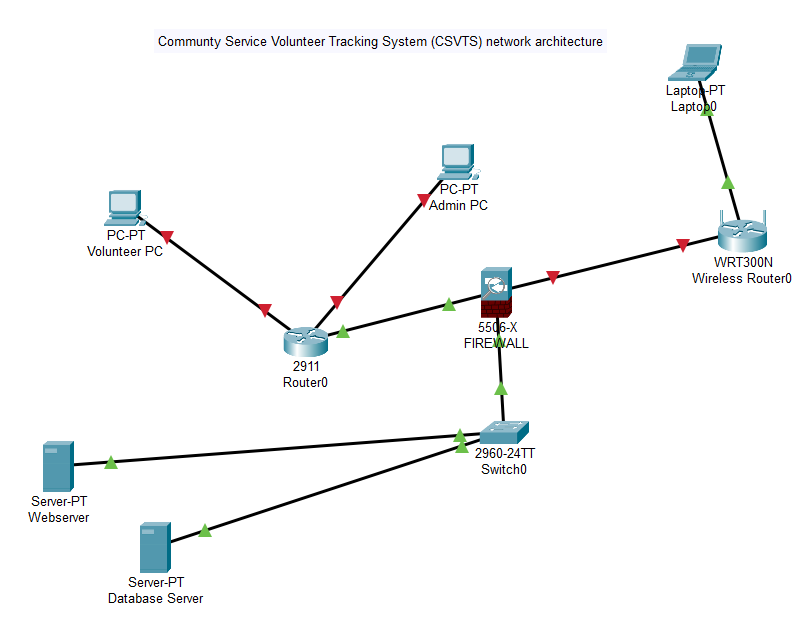
<https://www.forecast.app/learn/what-is-volunteer-time-tracking>

Kindful – Volunteer Management Software for Nonprofits

<https://kindful.com/nonprofit-glossary/volunteer-management-software-for-nonprofits>

Wikipedia – Non-profit Technology  
 <https://en.wikipedia.org/wiki/Non-profit_technology>

**Architectural Design**

Understanding how hardware will “shell” the product we are Engineering, since its impossible to see Software while its being engineered, to make things easier for developers to understand the other diagram of things we can’t see.  
  
**Fig 1. Network Architect**  


The following common objects belong to following physical components:

* Since it will be a cooperate setup, Computers are assigned to people: Administrators and Volunteers will exist in the system as Admin PC and Volunteer PC respectively.
* The web UI will exist inside the server.
* Database (My SQRL) will exist inside the Database server.
* Logins are controlled in the firewall.

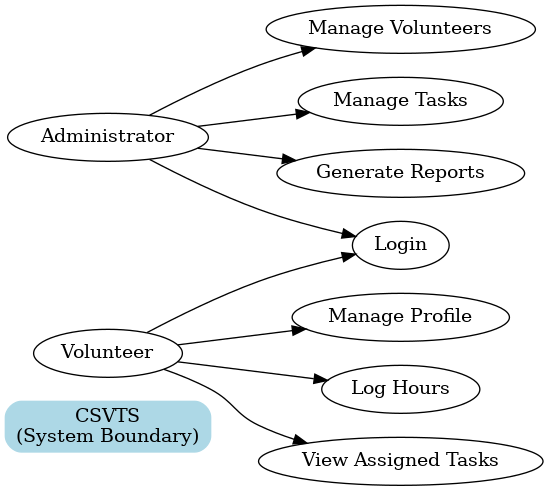
Now that we are familiar with the Typology of product, the other important 5 diagrams are:

**Fig 2. Context Diagram**  
A diagram of a community service

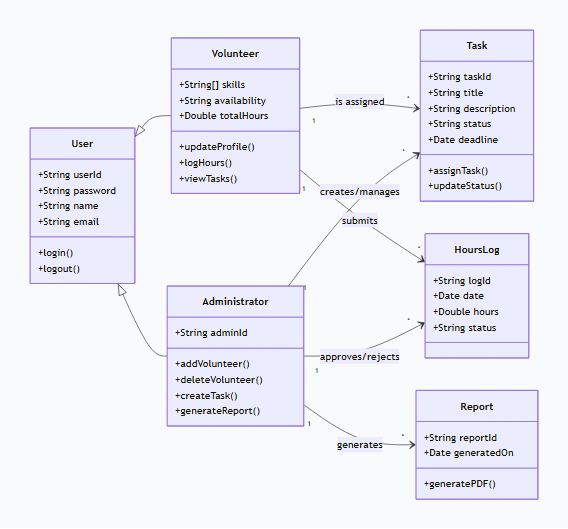
AI-generated content may be incorrect. **External Entities:** Volunteer, Administrator, Database Server.

* **System:** The central CSVTS process.
* **Interactions:** The arrows show the key data flows and commands between the external actors and the system, as defined in the SRS's functional requirements and use cases.

**Fig 3. Use Case diagram**



* Actors: Volunteer, Admin.
* Use cases: Login, Manage Profile, Manage Volunteers, Manage Tasks, Log Hours, Generate Reports.
* System boundary: “Community Service Volunteer Tracking System”.
* Volunteer and Administrator login in the webpage in the webpage using when webserver IP address is entered but every other method is unique to the user. (packet tracer simulation is provided in pocket tracer file to demonstrate this using a login interface).

**Fig 4. Class Diagram**  


* **Core Classes:** User (generalization), Volunteer, Administrator, Task, HoursLog, Report.
* **Key Relationships:**
  + **Inheritance (Generalization):** Volunteer and Administrator inherit from User.
  + **Associations:** Administrator creates/manages many Tasks. A Volunteer is assigned many Tasks and submits many HoursLog entries. An Administrator approves HoursLog and generates Reports.
* **Attributes & Methods:** Based on the features described in the SRS (e.g., logHours(), createTask(), totalHours attribute).

**Fig 5. Sequence Diagram: Volunteer Login Hours (main function)**

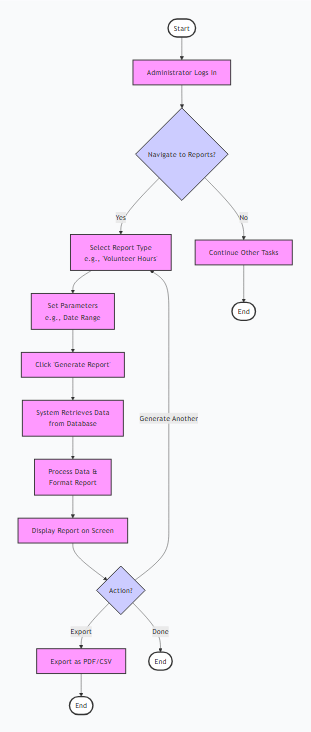
A diagram with text and words

AI-generated content may be incorrect.

This diagram shows the interactions between objects over time for the "Log Hours" use case.

* **Objects:** The Volunteer actor interacts with the Web UI, which sends messages to the HoursController (a system object that handles the logic).
* **Flow:** The controller creates a new HoursLog object, which saves itself to the Database. Confirmations are passed back up the chain to the user.
* **Lifelines:** The dashed vertical lines show the existence of each object during the interaction.

**Fig 6. Activity Diagram Generating a report**



* **Swimlanes (Implied):** The entire activity is performed by the **Administrator** actor, triggering system processes.
* **Flow:** The workflow begins with login, moves through the steps of selecting and generating a report, and ends with various final actions (export, generate another, done).
* **Decision Points:** The diamonds ({}) represent points where the flow branches based on the administrator's choice (e.g., whether to navigate to reports, what to do after seeing the report).

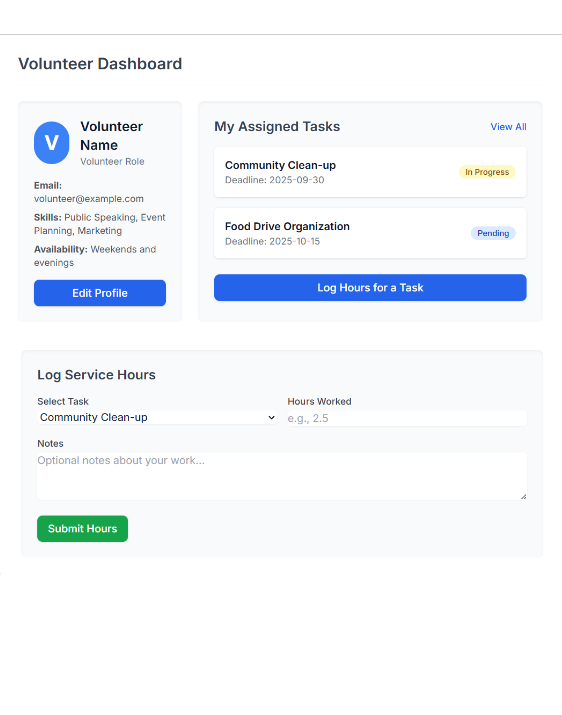
This set of diagrams provides a complete system model, bridging the requirements from the SRS to a detailed design ready for implementation, covering structural, behavioral, and interaction perspectives.

**3.1 User Interface Design**

The user interface is the primary point of interaction for both volunteers and administrators. The design focuses on simplicity, clear navigation, and an intuitive layout.

**Volunteer Dashboard Mockup:**

* Description - The volunteer dashboard provides a quick overview of a volunteer's activities. It includes a profile summary, a list of assigned tasks, and a form to log new hours. This allows volunteers to manage their contributions from a single, centralized page.

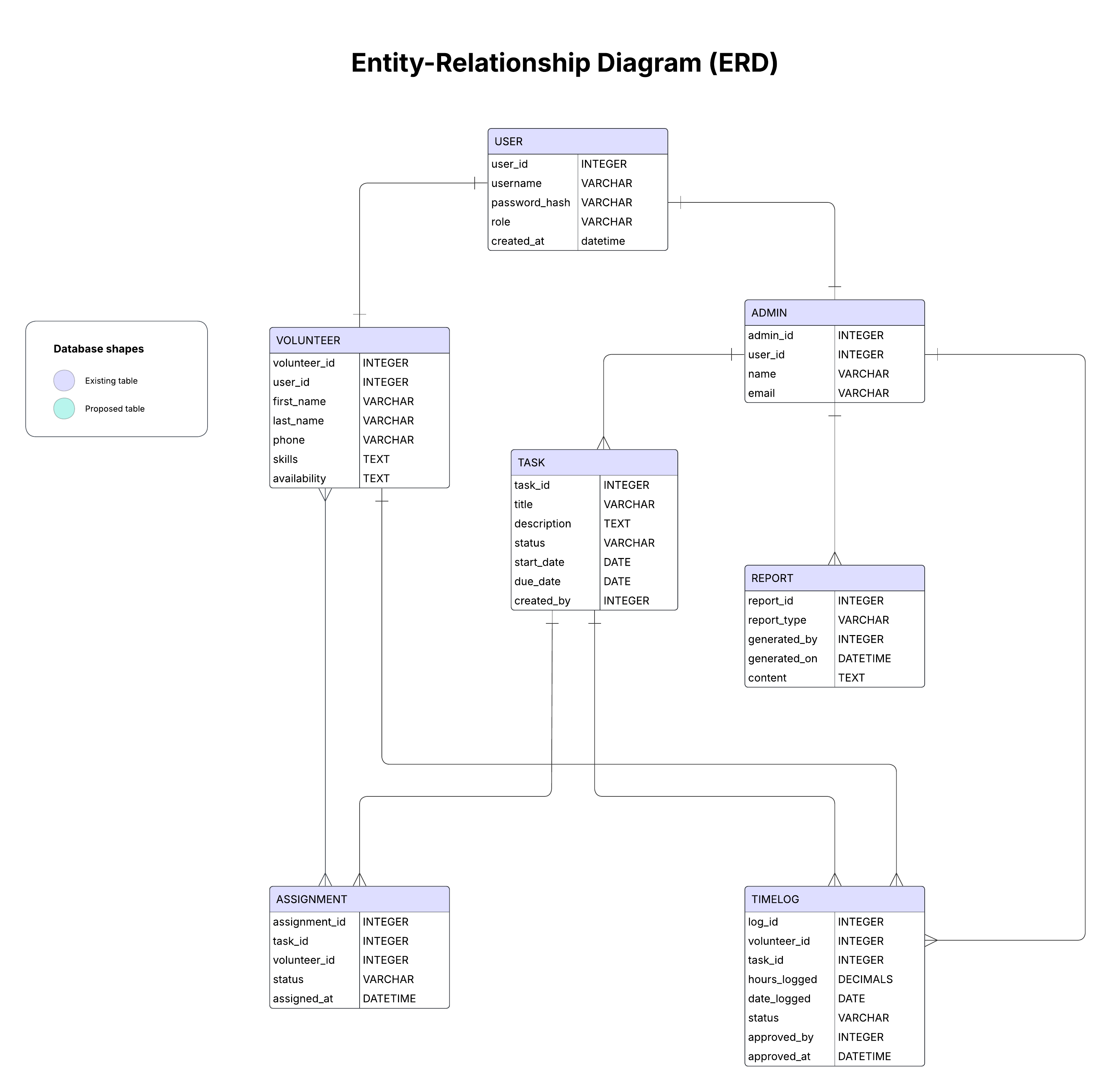


**Administrator Dashboard Mockup:**

* Description **-** The administrator's dashboard is a command center for managing the entire volunteer program. It features a high-level summary of total volunteers and tasks, a searchable list of all volunteers, and quick links to manage tasks and generate reports. This design facilitates efficient oversight and management.



3.2. **Data/Database design**



**Key Relationships Explained:**

1. User Inheritance: The admin and volunteer tables extend the user table (user\_id FK). A user can only be one type.
2. Task Creation: A task is created by an admin (created\_by).
3. Task Assignment: An assignment is a junction between a task and a volunteer. This allows a task to be assigned to multiple volunteers and a volunteer to have multiple tasks.
4. Hour Logging: A timelog is created by a volunteer for a specific task. It can be approved by an admin (approved\_by).
5. Reporting: A report is generated by an admin (generated\_by).

Database Schema (PostgreSQL)

Users Table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Administrators |  |  |
| Field | **Type** | **Constraints** | **Feature Key** |
| admin\_id | SERIAL | PK |  |
| user\_id | INTEGER | NOT NULL, UNIQUE | user.user\_id |
| name | VARCHAR |  |  |
| email | VARCHAR | UNIQUE, NOT NULL |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Users |  |  |
| Field | **Type** | **Constraints** | **Feature Key** |
| user\_id | SERIAL | PK |  |
| username | VARCHAR | UNIQUE, NOT NULL |  |
| password\_hash | VARCHAR | NOT NULL |  |
| role | VARCHAR | NOT NULL, CHECK |  |
| created\_at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |  |

Administrator Table:

Volunteer Table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Volunteer |  |  |
| Field | **Type** | **Constraints** | **Feature Key** |
| volunteer\_id | SERIAL | PK |  |
| user\_id | INTEGER | UNIQUE, NOT NULL | user.user\_id |
| first\_name | VARCHAR | NOT NULL |  |
| last\_name | VARCHAR | NOT NULL |  |
| phone | VARCHAR | Optional |  |
| skills | TEXT |  |  |
| availability | TEXT |  |  |

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Task |  |  |
| Field | **Type** | **Constraints** | **Feature Key** |
| task\_id | SERIAL | PK |  |
| title | VARCHAR | NOT NULL |  |
| description | TEXT |  |  |
| status | VARCHAR | NOT NULL, DEFAULT ‘OPEN’ |  |
| start\_date | DATE |  |  |
| due\_date | DATE |  |  |
| created\_by | INTEGER | NOT NULL | admin.admin\_id |

Assignment table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Assignment |  |  |
| Field | **Type** | **Constraints** | **Feature Key** |
| assignment\_id | SERIAL | PK |  |
| task\_id | INTEGER | NOT NULL | task.task\_id |
| volunteer\_id | INTEGER | NOT NULL | Volunteer.volunteer\_id |
| Status | VARCHAR | NOT NULLL, DEFAULT ‘ASSIGNED’ |  |
| assigned\_at | TIMESTAMP | NOT NULL |  |

TimeLog table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | TimeLog |  |  |
| Field | **Type** | **Constraints** | **Feature Key** |
| log\_id | SERIAL | PK |  |
| volunteer\_id | INTEGER | NOT NULL | volunteer.volunteer\_id |
| task\_id | INTEGER | NOT NULL | task.task\_id |
| hours\_logged | DECIMAL | NOT NULLL, CHECK > 0 |  |
| date\_logged | DATE | NOT NULL |  |
| Status | VARCHAR | NOT NULL, DEFAULT ‘PENDING’ |  |
| approved\_by | INTEGER |  | admin.admin\_id |
| approved\_at | TIMESTAMP |  |  |

Report table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Report |  |  |
| Field | **Type** | **Constraints** | **Feature Key** |
| report\_id | SERIAL | PK |  |
| report\_type | VARCHAR | NOT NULL |  |
| generated\_by | INTEGER | NOT NULL | admin.admin\_id |
| generated\_on | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |  |
| content | TEXT |  |  |

Data Storage and Retrieval Strategies

The system will use a relational database management system (RDBMS) for persistent data storage. This approach provides a structured and reliable way to manage the relationships between different data entities.

Storage Strategy

Data will be stored in normalized tables to minimize redundancy and improve data consistency. For instance, volunteer information is stored once in the Volunteer table, and each time log entry simply references the volunteer\_id, preventing duplicate data.

Retrieval Strategy:

Data retrieval will be managed using structured query language (SQL). This allows the system to efficiently perform complex queries, such as:

* Retrieving all tasks assigned to a specific volunteer.
* Calculating the total hours logged for a particular task or by a specific volunteer.
* Generating reports on volunteer engagement, as outlined in the project requirements.

By leveraging a relational database and SQL, the system can ensure fast and reliable access to the data.

**3.3. Component/Module Design**

**Major Components**

1. User Authentication Module

* Purpose **-** to manage all user access to the system
* Responsibilities:
  + - Handle user sign-up, including creating a new user record in the Volunteer or Administrator tables.
    - Validate user credentials during login.
    - Manage sessions to ensure users remain authenticated.
    - Enforce role-based access control, ensuring that administrators and volunteers only see the data and features relevant to their roles.

1. Volunteer Profile Management Module
   * Purpose - to allow volunteers to create, view, and update their personal information.
   * Responsibilities:
     + Provide an interface for volunteers to edit their profile details.
     + Update the Volunteer table with any changes to a volunteer's name, email, skills, or availability.
     + Display a volunteer's profile information.
2. Task Management Module
   * Purpose - to manage the creation, assignment, and status tracking of all community service tasks.
   * Responsibilities:
     + Allow administrators to create new tasks, which are stored in the Task table.
     + Enable administrators to update the status of existing tasks (e.g., from "Open" to "Completed").
     + Provide an interface for both volunteers and administrators to view a list of available or assigned tasks.
3. Time Logging & Reporting Module
   * Purpose - to accurately track volunteer hours and generate meaningful reports.
   * Responsibilities:
     + Accept time log submissions from volunteers, recording the data in the TimeLog table.
     + Validate that the time log corresponds to an existing task and volunteer.
     + Allow administrators to review and approve submitted hours.
     + Generate reports on total volunteer hours, completed tasks, and other key metrics by querying the TimeLog, Volunteer, and Task tables.

**Critical Algorithms and Data Structures**

User Authentication

A crucial algorithm for this system is secure password hashing. Instead of storing plaintext passwords, the system will hash them using a one-way cryptographic function. This protects user data in the event of a security breach.

**Algorithm for Password Hashing (Pseudocode):**

function hashPassword(password):

// Generate a unique, random salt for each user

salt = generateRandomSalt()

// Combine the password and the salt and hash them

hashedPassword = PBKDF2\_HASH(password + salt)

// Store both the hashed password and the salt in the database

return {hashedPassword, salt}

function verifyPassword(inputPassword, storedHashedPassword, storedSalt):

// Re-hash the user's input password using the stored salt

hashedInput = PBKDF2\_HASH(inputPassword + storedSalt)

// Compare the newly hashed password with the one from the database

if hashedInput == storedHashedPassword:

return true

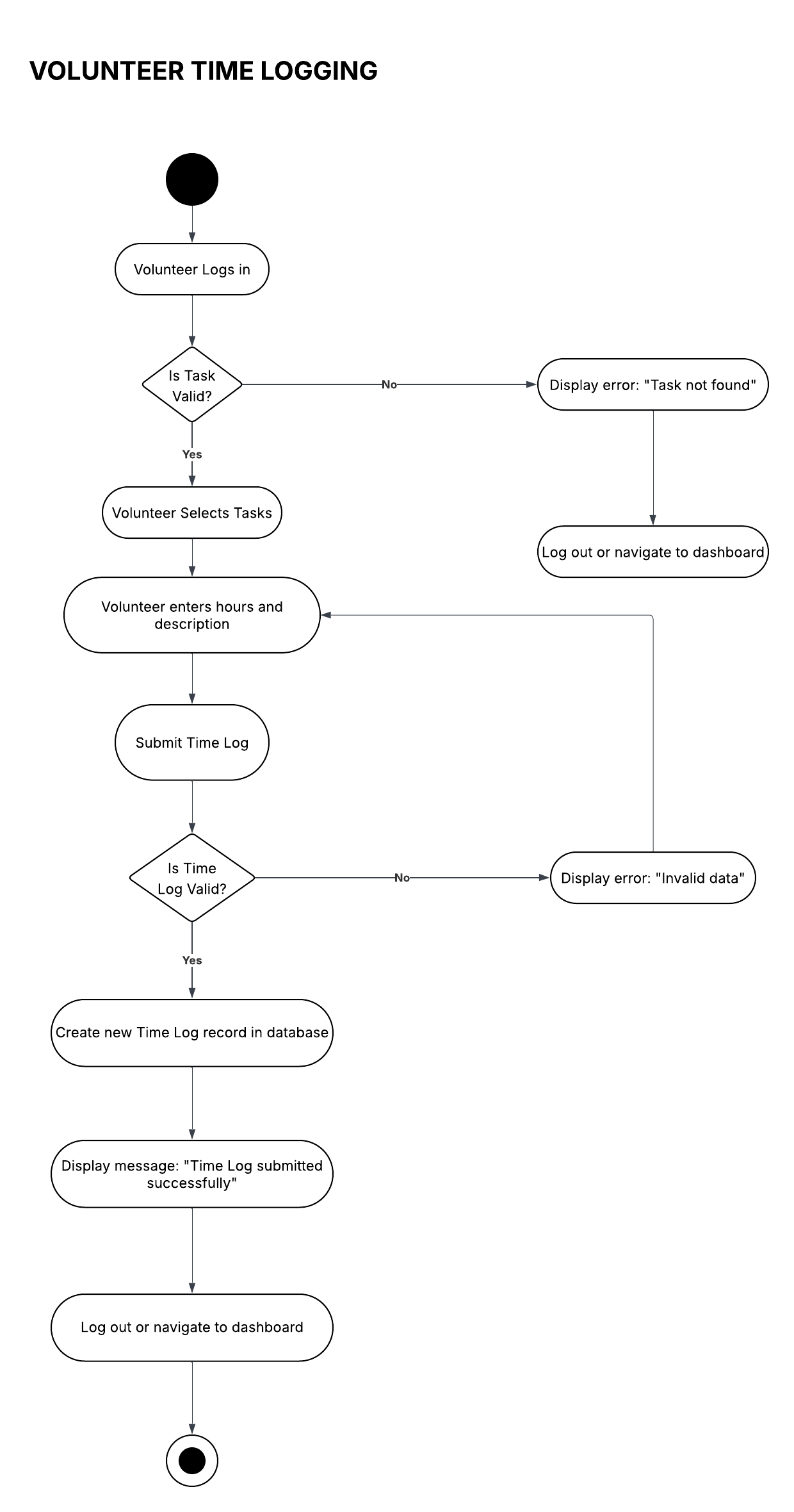
else:

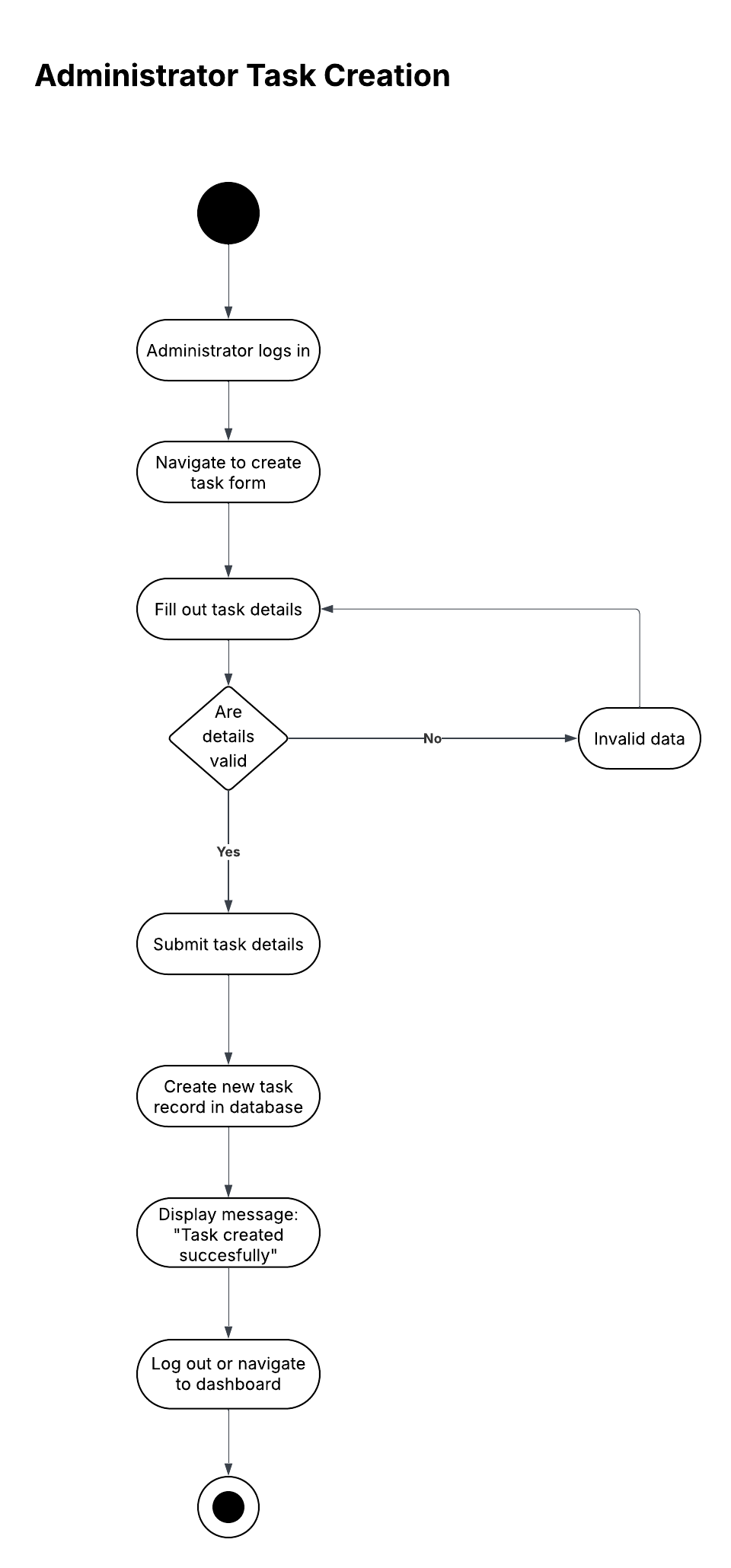
return false

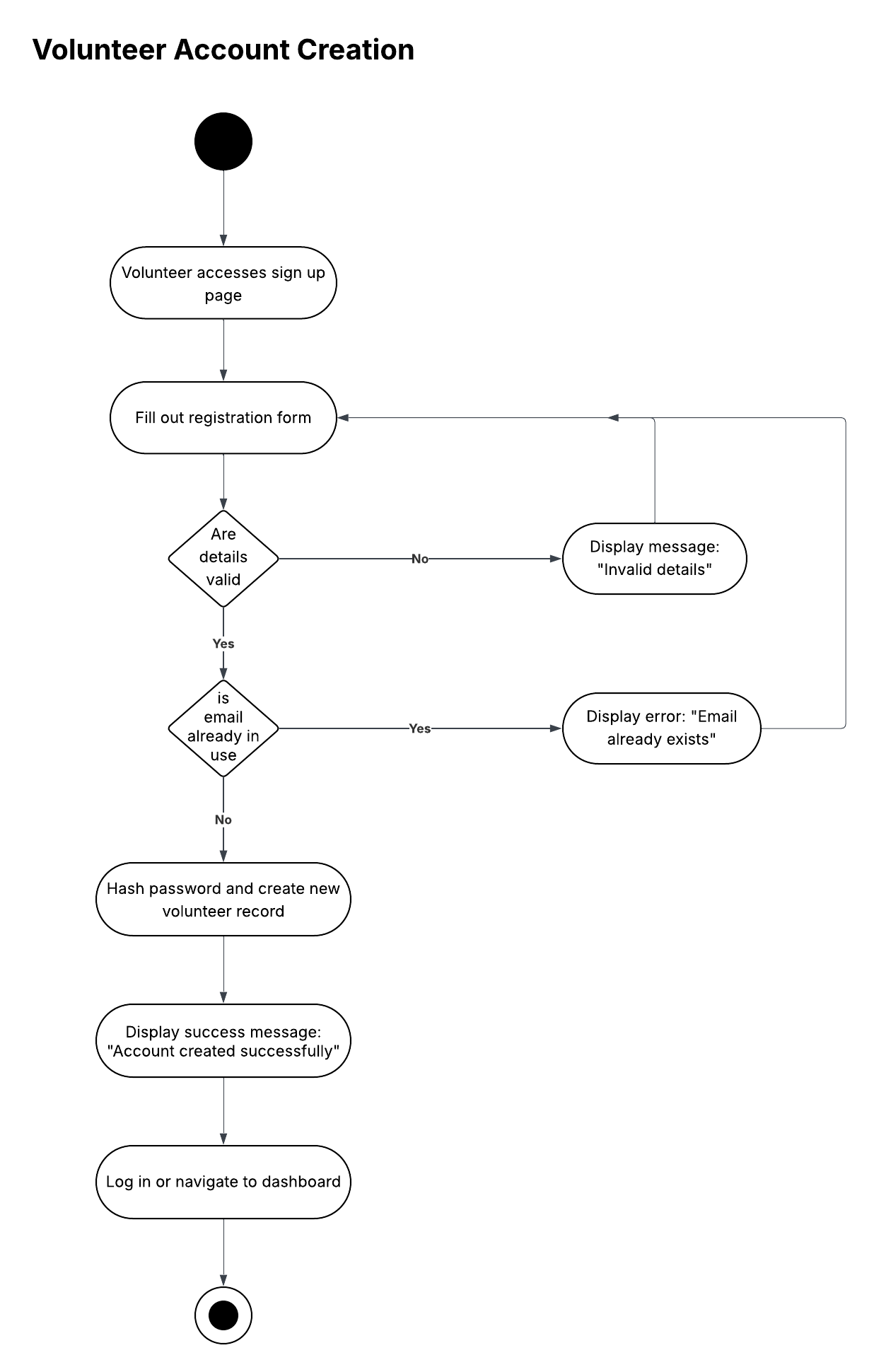
**Data Structures**

The database schema itself acts as the primary data structure for the system. The four tables (Administrator, Volunteer, Task, TimeLog) are relational, with links established via foreign keys. This structure allows for efficient data retrieval and ensures data integrity. For example, to find all hours a specific volunteer has logged, the system can use the volunteer\_id in the TimeLog table to quickly retrieve all relevant records.

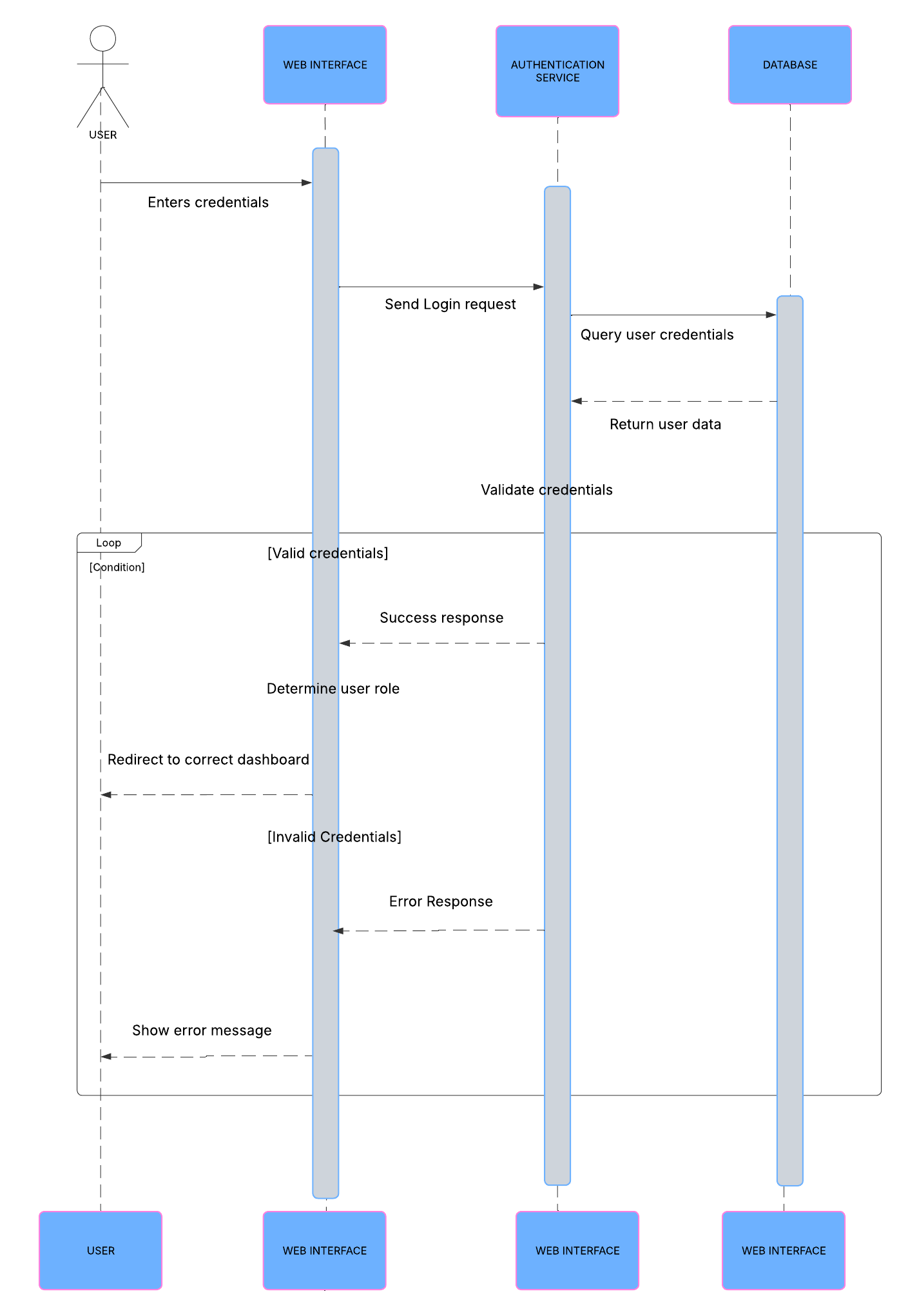
**Activity diagrams**

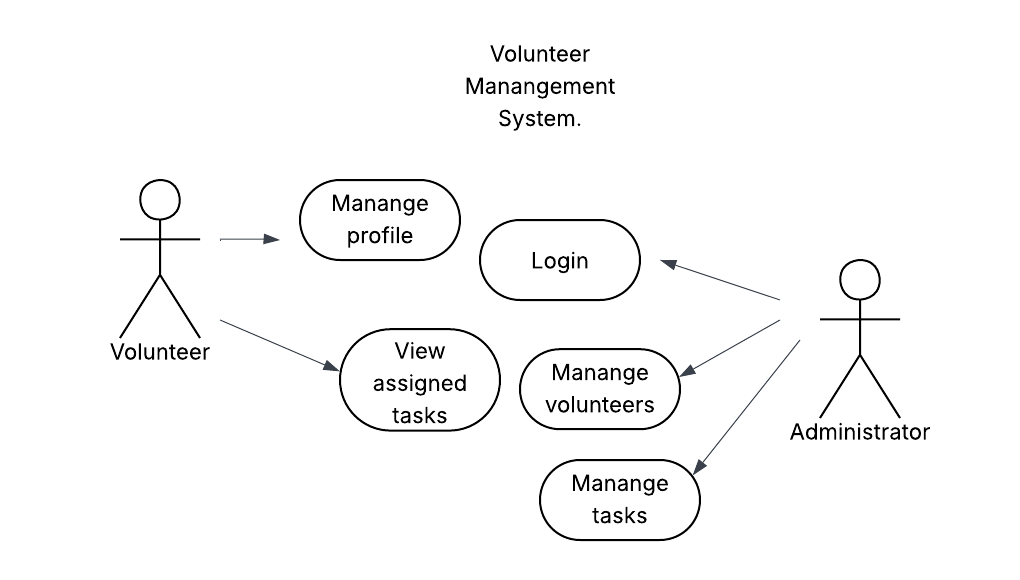






**Sequence Diagram**





4.

1. Login

Admin enters username and password. System validates credentials against the admin database. If valid → access granted; else → error message displayed.

2. Manage Volunteers

Admin can add, edit, or remove volunteer records. System stores volunteer details (contact info, skills, availability) in the database. Admin can also approve or reject volunteer applications.

3. Manage Tasks

Admin creates tasks/opportunities and assigns them to volunteers. System links tasks to volunteer profiles and updates availability. Admin can edit or close tasks once completed.

Volunteers

1. Manage Profile

Volunteer logs in and updates personal details (contact info, skills, availability).System saves changes and ensures updated data is reflected in future assignments.

2. View Assigned Tasks

Volunteer logs in and selects "My Tasks." System retrieves assigned tasks from the database and displays deadlines, details, and status. Volunteer can mark tasks as completed once done.

**5.0 Non-Functional Requirements**

**5.1 Usability**

* The system shall have an intuitive user interface that allows volunteers and administrators to easily navigate with minimal training.
* The system shall provide mobile-friendly access so volunteers can log hours and activities from smartphones or tablets and PCs.
* The system shall offer search, filter, and sorting options to quickly locate volunteers, events, or logged hours.
* The system shall provide accessibility support (e.g., screen reader compatibility, high-contrast mode) to accommodate users with disabilities.

**5.2 Performance**

* The system shall support at least **800 concurrent users** without performance degradation.
* The system shall load the dashboard within **8 seconds** under normal network conditions.
* The system shall process and store a volunteer’s logged hours in less than **8 seconds**.
* The system shall handle a database a max of at least **200,000 volunteers on records** with efficient query response times.

**5.3 Security**

* The system shall require secure user authentication with unique usernames and strong passwords.
* The system shall also support role-based access control (e.g., volunteer, coordinator, administrator).
* All data in transit shall also be encrypted using **HTTPS/SSL and other implements**.
* Sensitive information (such as personal details of volunteers) shall be encrypted at rest.
* The system shall log user activities (e.g., login, logout, data editing) for system purposes.

**5.4 Reliability**

* The system shall provide **99.6% uptime** availability excluding scheduled maintenance.
* The system shall include automatic data backup at least once every **24 hours**.
* In the event of a system crash, the system shall recover to the last consistent state without data loss. The system shall be capable of scaling horizontally to support additional load during peak volunteer events.

6.Summary

The Community Service Volunteer Tracking System (CSVTS) provides an efficient and centralized platform for managing volunteer engagement. It allows volunteers to register, log their service hours, and track their participation in community projects, while giving administrators tools to monitor contributions, validate records, and generate reports. This reduces manual paperwork, improves transparency, and enhances accountability within organizations.  
  
The system is designed with a modular approach, consisting of core components such as Login, Volunteer Management, and Reporting. It ensures usability across devices through a simple and accessible interface, while maintaining performance, security, and reliability. Reports can be quickly generated to provide insights into volunteer contributions and project impact.  
  
Looking ahead, the design can be extended to include mobile integration, automated notifications, and additional features to further support engagement and accessibility. Overall, CSVTS transforms volunteer management into a streamlined and impactful process that strengthens community service initiatives.