Individual Project:

Executive Summary

Matilda Nilsson

Introduction

This executive summary presents the complete design and build of a logical database for ZeroTrace, a platform developed for DOST RIM (Digital Innovations Center). The goal of ZeroTrace is to support individuals with disabilities by providing anonymous assistance. This idea is in response to the need for a accessible system that ensures privacy and meaningful interactions between individuals and helpers.

1. Overview

The PostgreSQL-based relational database was designed with structured workflows that support accessibility, privacy-by-design, and future scalability through secure, consistent data handling. Some key Design Features include:

- User profile roles, demographics, and accessibility preferences.
- Disability Taxonomy allows categorization by specific condition and broader category.
- Assistance Request Logs track support interactions, urgency, and status. Response Records are linked to helper actions and timestamped for auditing.

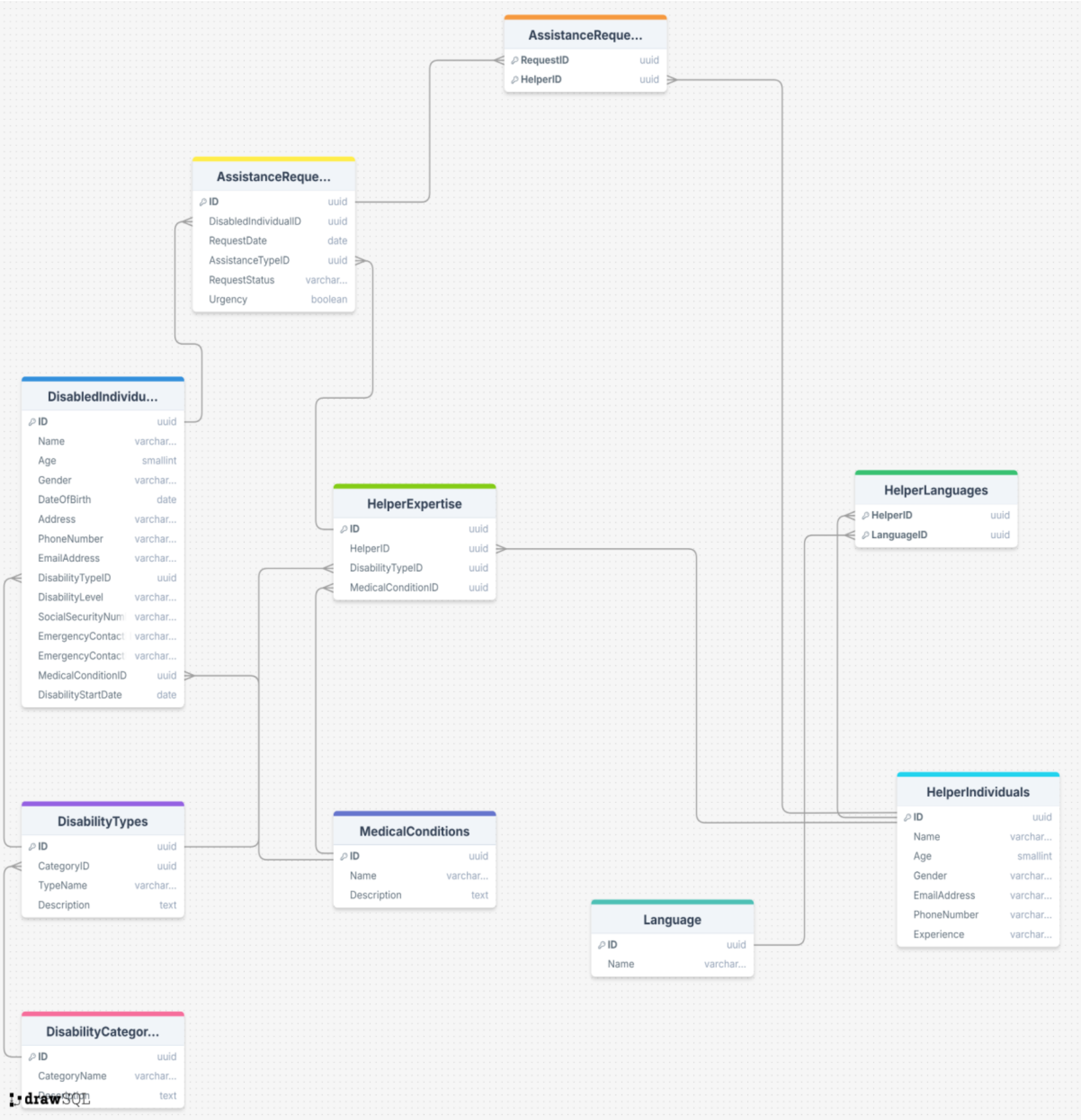


Figure 1 Entity Relationship Diagram (ERD)

The *Figure 1* diagram created illustrates the core entities (Users, Disabilities, Helpers, Assistance Requests) and their relationships. Foreign keys ensure referential integrity, while ENUMs (enumeration) restrict categorical data fields like role and urgency for consistency purposes.

The system also integrates real-time validation (e.g. email formatting) and backend data cleaning via SQL scripts to ensure consistent data quality. Without it, a differently formatted email could cause errors in the system.

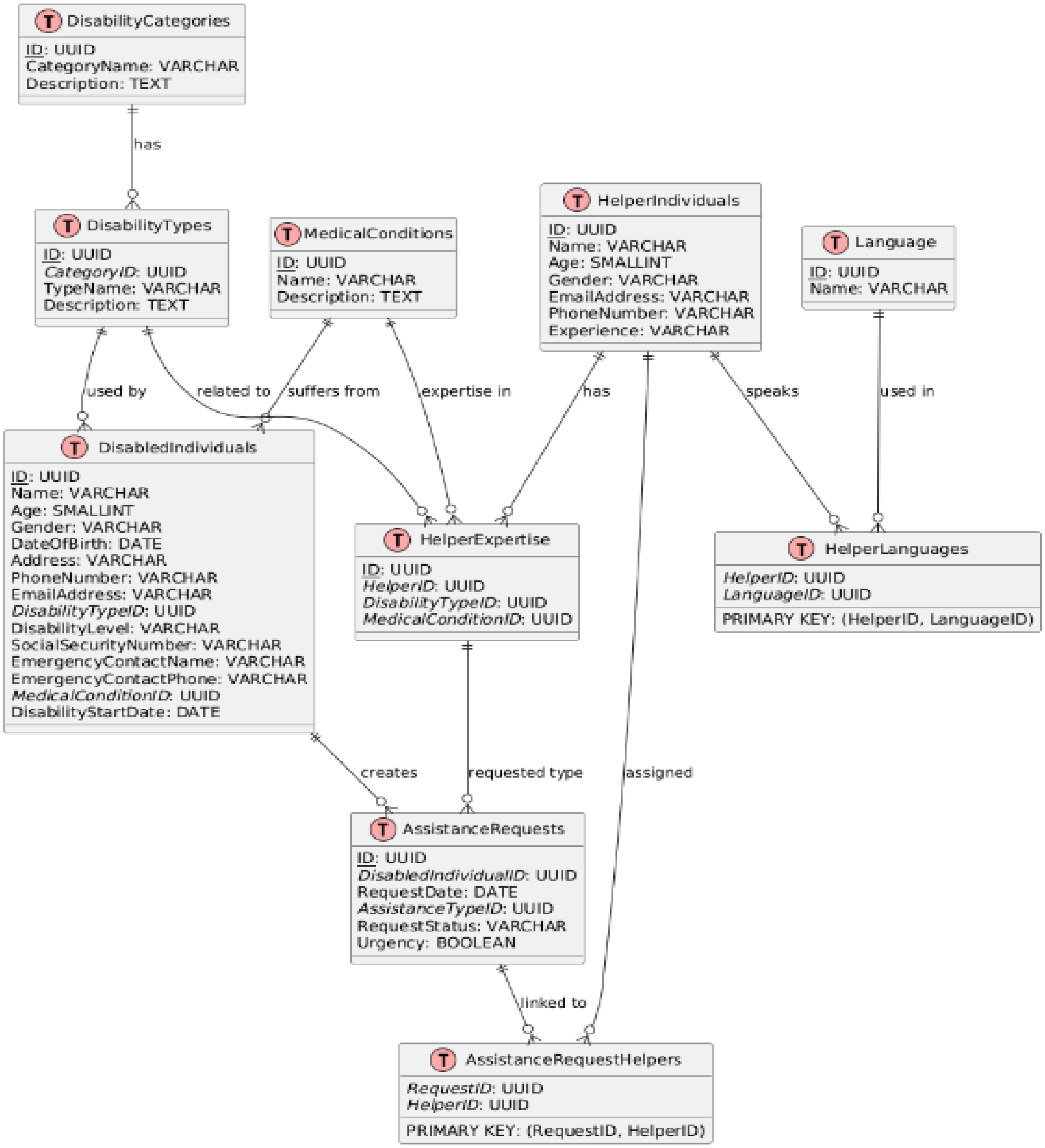


Figure 2 Extended Entity Relationship Flow

This diagram as shown in *Figure 2* provides a more detailed illustration of how data flows across the connected tables within the schema. It highlights the role of keys (e.g., UUIDs) for privacy, how helper actions are logged against user requests, and how categorical ENUM fields (such as urgency or role type) influence processing and reporting.

The following three data flow diagrams (DFDs) provide a multi-layered overview of how ZeroTrace processes user data and interactions securely, from the initial input to the backend storage.

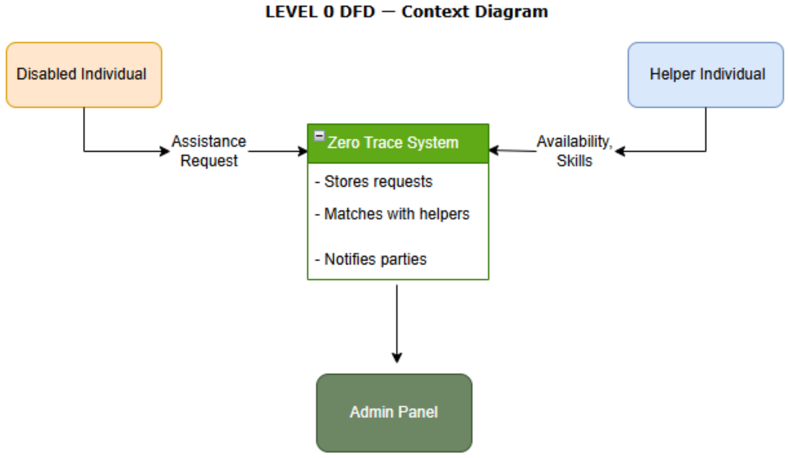


Figure 3 Level 0 - System Context Overview

This is the top-level diagram shown in *Figure 3*, that shows the high-level interaction between external users (the individuals with disabilities and helpers) and the system. It is emphasized with its simplicity and accessibility of user actions, such as submitting anonymous support requests or responding to them. From a business perspective, this level demonstrates how users engage with the platform. This is ensuring a secure way of removing stigma and ensuring inclusivity, aligning with the client's priority of ZeroTrace.

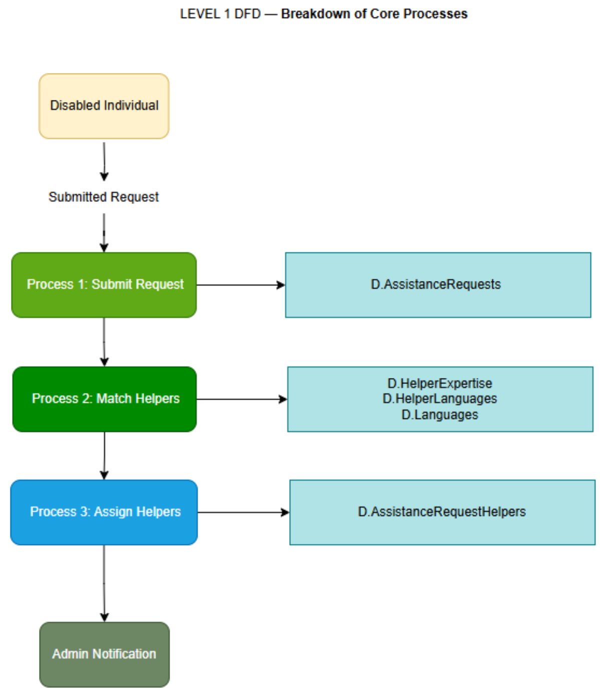


Figure 4 Level 1 of Internal System Operations

The Level 1 of the internal system operations build on the previous overview and introduces core system processes such as user role verification, request validation, and the matching algorithm that connects users to an appropriate helper. This level shows how requests are categorized based on urgency and status, which allows the platform to prioritize critical needs and assign verified responders.

Additionally, this system applies real-time validation rules at this stage, such as formatting checks on emails, flag duplicate entries, and role-based restrictions on access. This ensures that only authorized helpers can view or respond to sensitive issues containing private information, which supports both user safety and GDPR-compliant access control. This tier guarantees that data entering the system meets a strict quality standard before continuing, protecting both users and the organization.

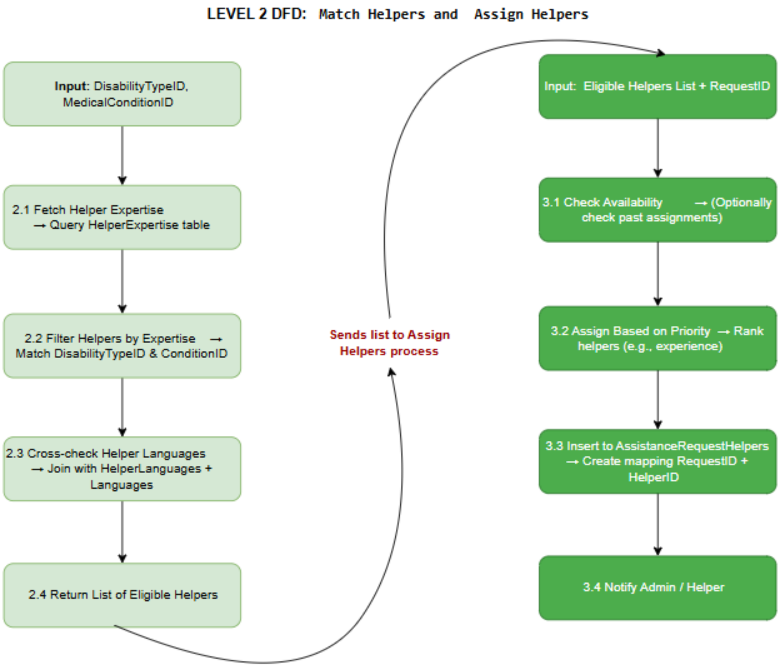

LEVEL 2 DFD: Match Helpers and Assign Helpers
Input: DisabilityTypeID, MedicalConditionID
2.1 Fetch Helper Expertise → Query HelperExpertise table
2.2 Filter Helpers by Expertise → Match DisabilityTypeID & ConditionID
2.3 Cross-check Helper Languages → Join with HelperLanguages + Languages
2.4 Return List of Eligible Helpers
Sends list to Assign Helpers process
Input: Eligible Helpers List + RequestID
3.1 Check Availability → (Optionally check past assignments)
3.2 Assign Based on Priority → Rank helpers (e.g., experience)
3.3 Insert to AssistanceRequestHelpers → Create mapping RequestID + HelperID
3.4 Notify Admin / Helper

Figure 5 Level 2 Data Handling and Integrity Enforcement

Figure 5 shows the backend mechanics that are responsible for data cleaning, error handling, and log generation. This is where automated SQL scripts perform tasks such as converting dates to a unified format, removing redundant or repeated records, applying default values when non-required fields are left blank by users, and creating auditable records.

This level ensures data quality, which aligns with GDPR principles of accuracy and traceability, and created logs that support future audits or recovery needs, acting like a failsafe. It also illustrates the use of ENUM fields and key foreign constraints to enforce compliance with the database schema and maintains integrity. These DFDs illustrate how PostgreSQL enables secure, scalable, and integrity-driven data processing—aligning with ZeroTrace's privacy-first mission.

2. A Critical Review of the Database Design

The ZeroTrace platform adopted a relational database model using PostgreSQL to address requirements, integrity enforcement, and long-term scalability. This model was well-suited to the project's initial scope, which demanded strict privacy controls, defined relationships (e.g., between users, disabilities, and support requests), and the ability to validate inputs in a controlled schema.

2.1 Strengths of the Relational Design

The database structure uses third normal form (3NF) to eliminate redundancy and maintain clean data relationships. This ensures that each data point (e.g., a disability type or request status) exists in only one place, reducing inconsistencies and improving maintainability. Foreign keys, ENUMs, and NOT NULL constraints were implemented to enforce relational rules, prevent orphaned records, and maintain referential integrity.

A relational model's rigidity supports GDPR compliance by allowing fine-grained control over user data access, deletion, and modification. For instance, the use of UUIDs for primary keys supports pseudonymization — a core GDPR principle — while foreign key cascading allows the platform to reliably erase user data across linked tables when requested.

Finally, PostgreSQL's structured format allows for schema migrations using tools like Alembic or pgAdmin. While changes require coordination, PostgreSQL supports versioning and rollback, which makes it feasible to evolve the database as the platform grows. Features like indexing and partitioning can be used to scale performance for increased data volumes. For example, as the platform adds more disability classifications or request types, new ENUM values or lookup tables can be added without disrupting existing data relationships.

2.2 Limitations and Solutions

The relational design is optimized for predefined fields but is not well-suited to handling free-text feedback, survey responses, or chat logs. Attempting to store such data in VARCHAR or TEXT columns limits searchability, semantic analysis, and indexing. Also, while PostgreSQL offers tools for schema versioning, adding or altering relationships (e.g., introducing new user types or multi-language support) can involve downtime and migration risks. In a production environment, these changes need to be tested extensively to prevent cascading failures or data loss. To address these limitations, the approach of hybrid model architecture should be considered. Where PostgreSSQL continues to handle the structured, critical data, a NoSQL database like MongoDB or CouchDB can manage unstructured data.

NoSQL is also flexible in its querying of JSON-like formats, creating a scale of functionality for ZeroTrace without sacrificing the security and benefits of the relational model. Many platforms utilize this combination of SQL for structure and NoSQL for flexibility, to meet both regulatory and user-experience needs (Božić, 2022).

3. Database Management System (DBMS)

While designing ZeroTrace, a critical decision involved selecting the appropriate Database Management System (DBMS) to support the platform’s needs. After evaluating several options, we selected PostgreSQL, an advanced open-source SQL-based system. However, we also explored alternatives, including NoSQL databases, to weigh their strengths and weaknesses for future platform development. This section explains what SQL and NoSQL systems are, why PostgreSQL was selected, and where a hybrid approach may benefit ZeroTrace going forward.

3.1 SQL database

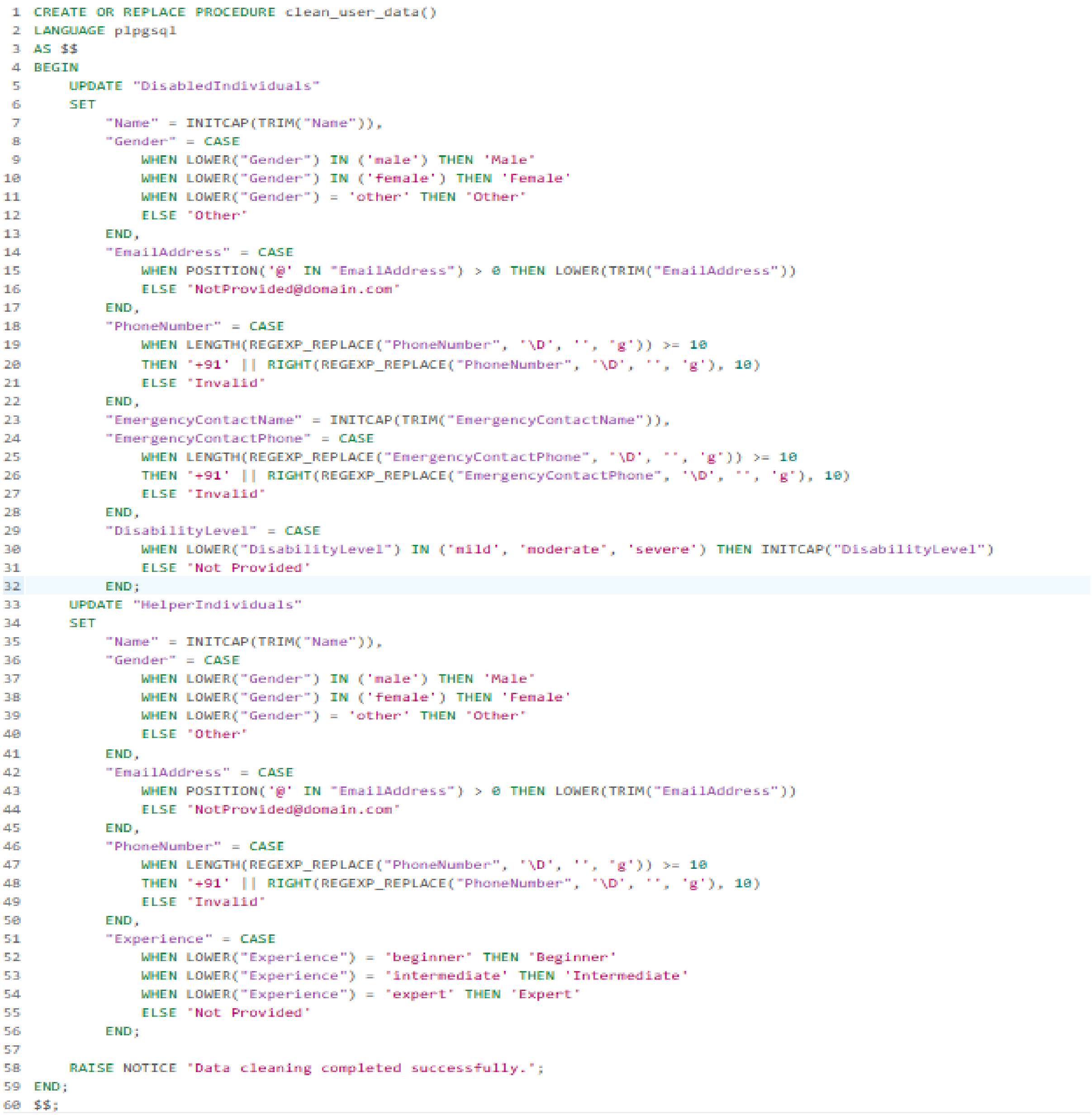
SQL (Structured Query Language) databases, like PostgreSQL or MySQL, are designed for storing structured data in a highly organized format. This means tables with rows and columns, like a spreadsheet. This format gives each table a label format with each “row” of information a consistent, expected form of information inside of it. This model ensures we have predictable relationships with the data; users, disability types, support requests, helper responses. Since these types of data do not change frequently and has a great deal of validation which ensures rules are consistently followed. Some rules include: every user has a unique ID number, each support request has a valid urgency level, and every helper is linked to a specific response. This flow pass through well-defined parameters which maintains a consistent, rule-following connection between each piece of data.

3.2 NoSQL database

NoSQL (Not Only SQL) databases, such as MongoDB, are designed with flexibility in mind. Instead of fixed tables, they function like free-form documents. This is ideal when user data varies or might change often over time (Estuary, 2025). NoSQL therefore can be thought of as a flexible list, where some information is a paragraph of text, others a bullet-point list. NoSQL databases are good at managing unstructured information which is ideal for things such as: anonymous user feedback, transcripts, journal entries, or optional accessibility preferences. These are areas where the strict rules of SQL can be limiting.

3.3 Why PostgreSQL Was Chosen

ZeroTrace chose to use PostgreSQL for several reasons that overall aligned with the main priorities. The platform needed strong relational enforcement to track users, requests, and information securely. PostgreSQL features security and compliance such as UUIDs, role-based access, cascading deletion, which are all important for GDPR compliance (GDPR.eu, n.d.). Lastly, PostgreSQL is a widely supported and proven choice for large-scale structured data.



```sql
CREATE OR REPLACE PROCEDURE clean_user_data()
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE "DisabledIndividuals"
    SET
        "Name" = INITCAP(TRIM("Name")),
        "Gender" = CASE
            WHEN LOWER("Gender") IN ('male') THEN 'Male'
            WHEN LOWER("Gender") IN ('female') THEN 'Female'
            WHEN LOWER("Gender") = 'other' THEN 'Other'
            ELSE 'Other'
        END,
        "EmailAddress" = CASE
            WHEN POSITION('@' IN "EmailAddress") > 0 THEN LOWER(TRIM("EmailAddress"))
            ELSE 'NotProvided@domain.com'
        END,
        "PhoneNumber" = CASE
            WHEN LENGTH(REGEXP_REPLACE("PhoneNumber", '\D', '', 'g')) >= 10
            THEN '+91' || RIGHT(REGEXP_REPLACE("PhoneNumber", '\D', '', 'g'), 10)
            ELSE 'Invalid'
        END,
        "EmergencyContactName" = INITCAP(TRIM("EmergencyContactName")),
        "EmergencyContactPhone" = CASE
            WHEN LENGTH(REGEXP_REPLACE("EmergencyContactPhone", '\D', '', 'g')) >= 10
            THEN '+91' || RIGHT(REGEXP_REPLACE("EmergencyContactPhone", '\D', '', 'g'), 10)
            ELSE 'Invalid'
        END,
        "DisabilityLevel" = CASE
            WHEN LOWER("DisabilityLevel") IN ('mild', 'moderate', 'severe') THEN INITCAP("DisabilityLevel")
            ELSE 'Not Provided'
        END;
    UPDATE "HelperIndividuals"
    SET
        "Name" = INITCAP(TRIM("Name")),
        "Gender" = CASE
            WHEN LOWER("Gender") IN ('male') THEN 'Male'
            WHEN LOWER("Gender") IN ('female') THEN 'Female'
            WHEN LOWER("Gender") = 'other' THEN 'Other'
            ELSE 'Other'
        END,
        "EmailAddress" = CASE
            WHEN POSITION('@' IN "EmailAddress") > 0 THEN LOWER(TRIM("EmailAddress"))
            ELSE 'NotProvided@domain.com'
        END,
        "PhoneNumber" = CASE
            WHEN LENGTH(REGEXP_REPLACE("PhoneNumber", '\D', '', 'g')) >= 10
            THEN '+91' || RIGHT(REGEXP_REPLACE("PhoneNumber", '\D', '', 'g'), 10)
            ELSE 'Invalid'
        END,
        "Experience" = CASE
            WHEN LOWER("Experience") = 'beginner' THEN 'Beginner'
            WHEN LOWER("Experience") = 'intermediate' THEN 'Intermediate'
            WHEN LOWER("Experience") = 'expert' THEN 'Expert'
            ELSE 'Not Provided'
        END;

    RAISE NOTICE 'Data cleaning completed successfully.';
END;
$$;
```

Figure 6 Data Cleaning Script

This SQL script shown in *Figure 6* is an example of how PostgreSQL allows us to clean and validate data using structured queries. This reduces human error and improves overall data quality.

4. General Data Protection Regulation

As this product is developed to support individuals with disabilities, it handles sensitive personal information. This means the legal and ethical data practices are not optional, but essential. One of the most important frameworks regarding data protection is the General Data Protection Regulation (GDPR), which applies to any organization handling personal data of users in the EU.

GDPR is a European law that requires systems to collect only what is necessary (referred to as data minimization), allow users to request deletion of their data, prevent unauthorized access, and to log and justify all data handling (Bytebase, 2024). ZeroTrace also pseudonymizes data via UUIDs, which anonymizes a user's identity within the database by assigning them a Universally Unique Identifier, making it more difficulty for personal information to be exposed. (European Commission, 2021) This aligns with Article 5 within GDPR principle of data protection by design and default (GDPR.eu, n.d.).

Access to sensitive tables, for example disability types or user preferences, are restricted through role-based filters. PostgreSQL's foreign key cascading feature means that when a user profile is deleted, all related records such as assistance requests or logs are also removed. This supports a user's legal right to request deletion of their data as outlined in GDPR Article 17, Right to Erasure (Bytebase, 2024). To meet the requirements of data accuracy, the platform also performs automated validation and post-entry cleanup using SQL scripts. This is shown in *Visual 6* where a script is cleaning data, checking for duplicates and correcting format issues in real time.

Lastly, to adhere to audits and monitoring for accountability, ZeroTrace system logs actions like user sign-ups, helper responses, and support request edits. These logs allow the client organization to perform audits if needed. Additional tools like Prometheus were proposed in our recovery plan to track unusual activity and system health.

Summary and Conclusion

The ZeroTrace project successfully delivered a scalable and compliance-ready database solution for a digital support platform serving individuals with disabilities. The platform was built using a PostgreSQL relational model, chosen for its strengths in data integrity, structured validation, and legal compliance. Through careful design, the system supports anonymous user interaction, real-time validation, secure logging, and GDPR-aligned data management.

Visual data flows, cleaning scripts, and relational mappings all reinforce the platform's mission to offer a safe, stigma-free digital space for vulnerable users. Moreover this system is laying the groundwork for extended services and integrations. While the current implementation meets the immediate business requirements, ongoing improvements are essential to adapt to growing user needs and expanding service complexity. As Cavoukian (2009) notes, "privacy by design" requires that compliance not be an afterthought, but rather built into the data structure itself, which ZeroTrace achieves through its PostgreSQL architecture.

References

Božić, R., 2022. *Advantages and challenges of NoSQL compared to SQL databases: A systematic literature review*. Brčko: Faculty of Economics Brčko. Available at: https://www.researchgate.net/publication/365132154 [Accessed 16 Jun. 2025].

Bytebase, 2024. *Database compliance for GDPR: Implications and best practices*. Available at: https://www.bytebase.com/blog/database-compliance-gdpr/ [Accessed 16 Jun. 2025].

Cavoukian, A., 2009. *Privacy by Design: The 7 Foundational Principles*. Information & Privacy Commissioner of Ontario.

Estuary, 2025. *PostgreSQL vs. MongoDB: Differences, strengths, and use cases*. Available at: https://www.estuary.dev/postgresql-vs-mongodb/ [Accessed 16 Jun. 2025].

European Commission, 2021. *Commission Implementing Decision (EU) 2021/914*. Official Journal of the European Union, L199, 7 June. Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32021D0914 [Accessed 17 Jun. 2025].

GDPR.eu, n.d. *General Data Protection Regulation (GDPR) compliance guidelines*. Available at: https://gdpr.eu/ [Accessed 16 Jun. 2025].

Wikipedia, 2025c. *Privacy by design*. Available at: https://en.wikipedia.org/wiki/Privacy_by_design [Accessed 16 Jun. 2025].