Organ Donation Database

Eric Pham, Kate Dinh, Nicholas Bao
CS157A

Table of Contents

1. Introduction

1.1 Objective

An organ donation database provides a way for various hospital administrators to manage and access patient information, regardless if they're a donor or a recipient. This database also keeps track of organs currently available for donation, as well as those being requested from recipients, in addition to the priority of the organ transfers, which is based on the viability times of each organ.

1.2 Project Goals

The main goal of this project was to be able to implement a fully functional web application with a database that can track and edit organ donations within the system. We aimed to have a simple, efficient, and user-friendly interface that would manage recipients and donors by allowing administrators to add organs to priorities, as well as add, edit, and delete data in the tables. We wanted to model this database off of a real-world situation, therefore we only accounted for the administrator view, because patients wouldn't be able to manage their information

1.3 Project Design

For our tech stack, we decided on utilizing tools that we were all familiar with, which included the front end: HTML, CSS, Javascript, and React.js. Our back-end framework was coded within Node.js, and for our database, we used MySQL. For version control, as well as the utilization of branches to develop features and bug fixes, we decided to use GitHub.

As previously stated, the front-end portion of our project entailed us using HTML, CSS, and Javascript, and combining all of these, we were able to build a website UI, as well as create buttons for the user to interact with, such as the navigation bar, and similar features which included search, edit, delete, and create. For our back-end portion, we mainly used PHP, MySQL, and Node.js, and this allowed us to create the databases for organs, as well as the server that we used to access said database. This also allowed us to store backup data from the database, and from button presses and inputs on the website, we could change values in said database.

1.4 Project Functionality

The primary functionalities of our website are that we can add, edit, delete, and sort from each table, the capabilities depending on which table. Every table will be able to sort the data by any column, all you have to do is click on the column header and it will sort by ascending or descending, depending on what the previous sort order was. In terms of add, edit, and delete, certain tables will contain certain of these features

The first two tables, Organ Request and Organ Availability do not contain any of the 3 features but include a search bar. The search bar can be used to find specific requests, such as

looking for a request that is looking specifically for a heart or maybe an organ that matches a certain blood type.

The third table, Donor List, contains a feature that enables us to edit the information of a donor. This enables administrators to make any changes to the donors as situations may occur that may change the status of their wellbeing, and in change, impact their donation. On top of this, the Donor List also contains a search bar where we can find specific donors through more detailed information.

The fourth table, Organ Priority, enables us to add a new “organ priority”. What this means is that we can add a new organ into the system and we would be able to get a new priority number as well as an organ, and it will become an option of available organs. For example, let's say we want to add a new organ called “Brain” into the system, we would assign a priority number as well as a name to it and add it into the system. This table also has a search bar that enables us to find certain organs and their priority through a quick search.

The final table, Recipient Information, contains a delete function. The idea behind this is that once a recipient has received the organ, they would no longer need another organ donation, thus enabling us to remove them from our list of recipients needing an organ. This table also contains the search bar that lets us find specific recipients through specific pieces of information.

Since some functions can still be added, such as editing organ availability or deleting the donor list, our next steps would be to add these features.

2. Database Design

2.1 ER Diagram

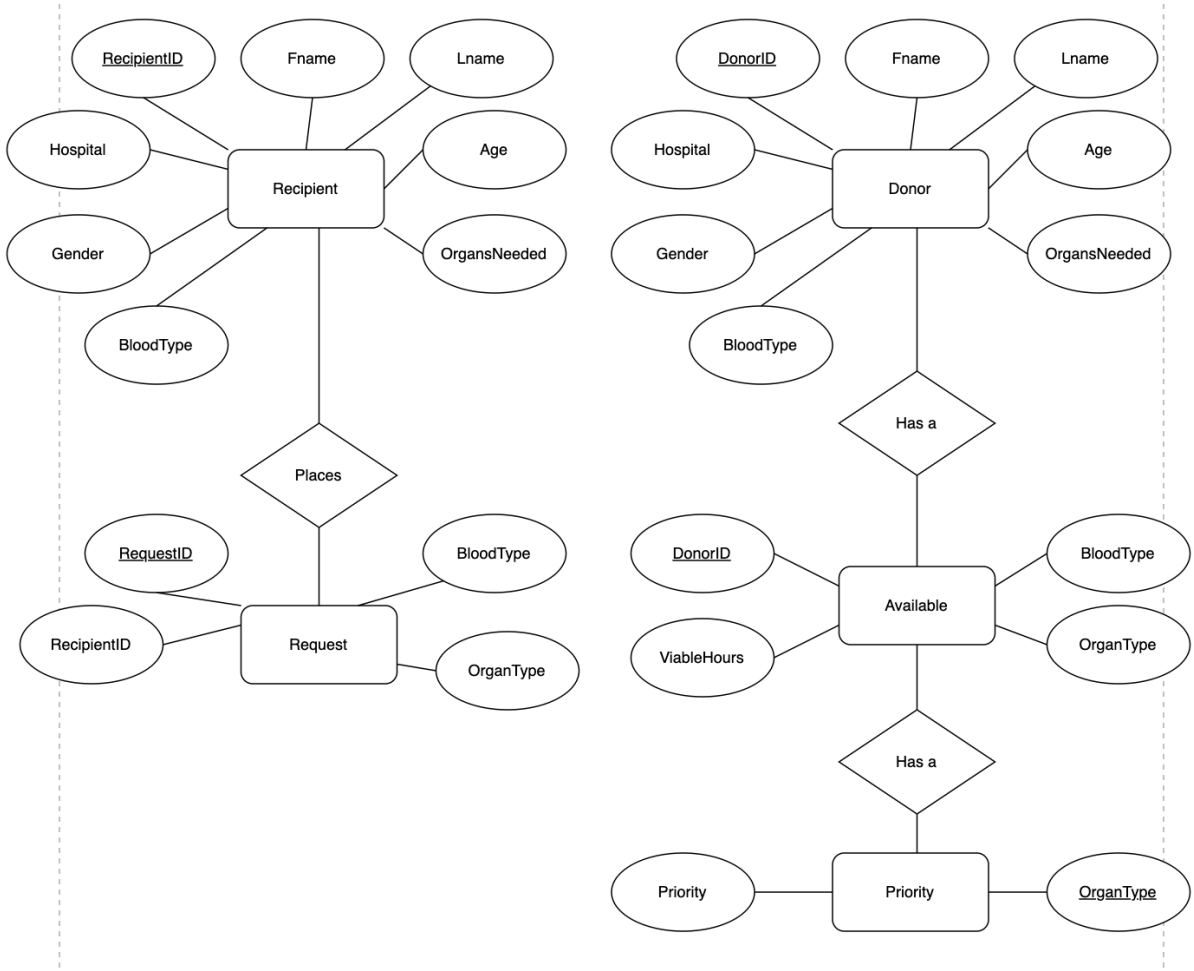

Figure 1: ER Diagram for Database

2.2 Table Schemas

The main tables we created for this project were organ priority, organ recipient, organ donor, organ requests, and organ availability. Our organ priority table stored the type of organ, as well as its priority on a scale of 1-6, where 1 was the highest priority, and 6 was the lowest. In this case, the priority was determined based on the viability of each organ. We also made the primary key the organ type, because organs cannot have the same name.

The next table that we created was the organ recipient table, which stored a recipient ID, first and last name, the recipient's age, organ needed, blood type, gender, and the hospital where the recipient was staying. In this case, the recipient ID would be the primary key since each recipient has an ID that is unique to them.

Our organ donor schema also stored a unique donor ID, which was the primary key, the first and last name, their age, the organs they were donating, as well as their blood type, gender, and the hospital that they were staying at.

In our organ request schema, we stored the request ID, which was unique to each request, and the recipient ID, as well as the organ type and the blood type. In this case, we decided to make the request ID the primary key as well, since one recipient could have requested multiple organs.

For the organ availability schema, we stored the donor ID, organ type, blood type, and the viable hours for each organ. In this case, the primary key is the donor ID since one donor could donate multiple organs.

2.3 Normalization of Tables

All of our tables were in the normalization stage of 3NF. This ensured that none of our tables had any partial or transitive dependencies. To first transition to 1NF, we had to make sure that none of our tables had multiple values in columns. This meant that any recipients who needed more than 1 organ would have to have multiple entries into the database. Continuing, this would also mean that any organ donors that were donating more than one organ would also have to be stored as multiple entries in the database.

We then normalized our tables to 2NF, which meant that we had to remove any partial functional dependencies from the tables, specifically, we didn't choose to include the patient or donor SSN since the patient and donor ID is already unique to each patient or donor. Having their SSN in the database as well would contribute to the fact that the table would not be in 2NF normalization.

After the normalization of said tables to 2NF, we were then given the task of normalizing them to 3NF, which meant that we would not have any partial dependencies within the tables. We did this by not including a donor ID and SSN in the tables since both could be used to identify the donor name, therefore they could transitively identify each other(the SSN and donor ID value).

3. Application Views

3.1 Introduction into Views

For our application, we wanted to maintain it in a strictly administrative view. This is because, in a real-world experience, donors and recipients wouldn't be able to access other patient's records, due to doctor-patient confidentiality. All organ donations would be up to the discretion of a patient's care team, thereby the administrative view is the best way to accurately display our application.

3.2 Home Screen

This is the generic view of the homepage of the website.

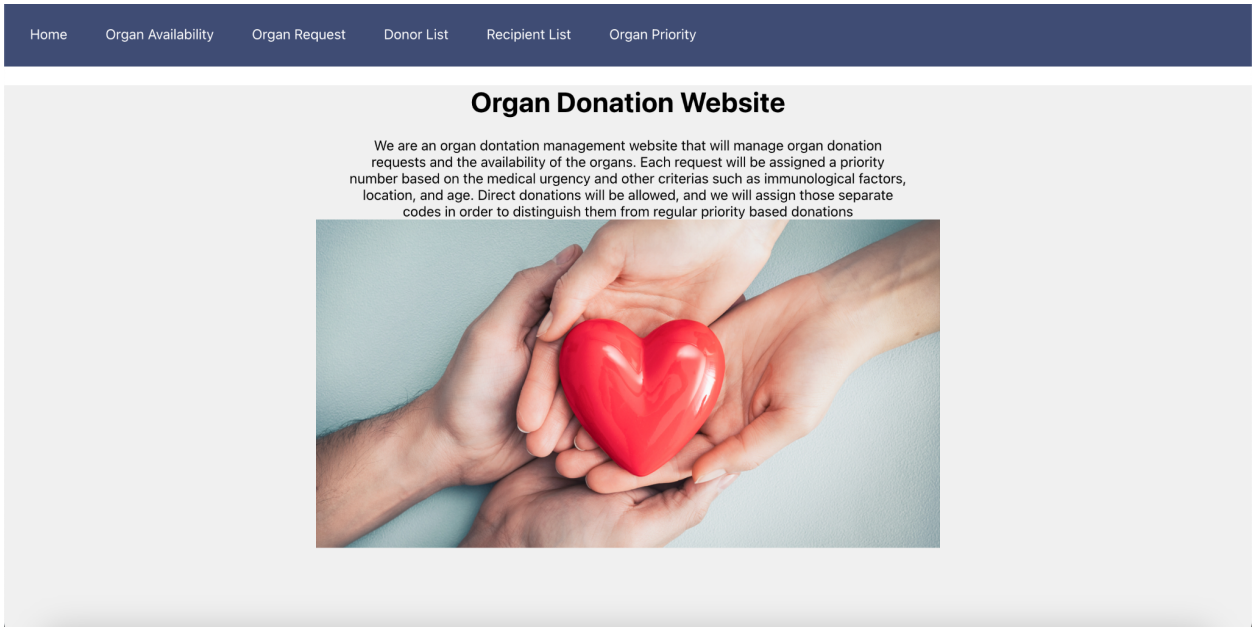

Figure 2: Home Screen of Application

3.3 Tables

3.3.1 Organ Availability Table

Within Figure 3, we can see that there are a total of 7 available organs, each having their unique organ ID. The current table is sorted by descending order of Donor ID.

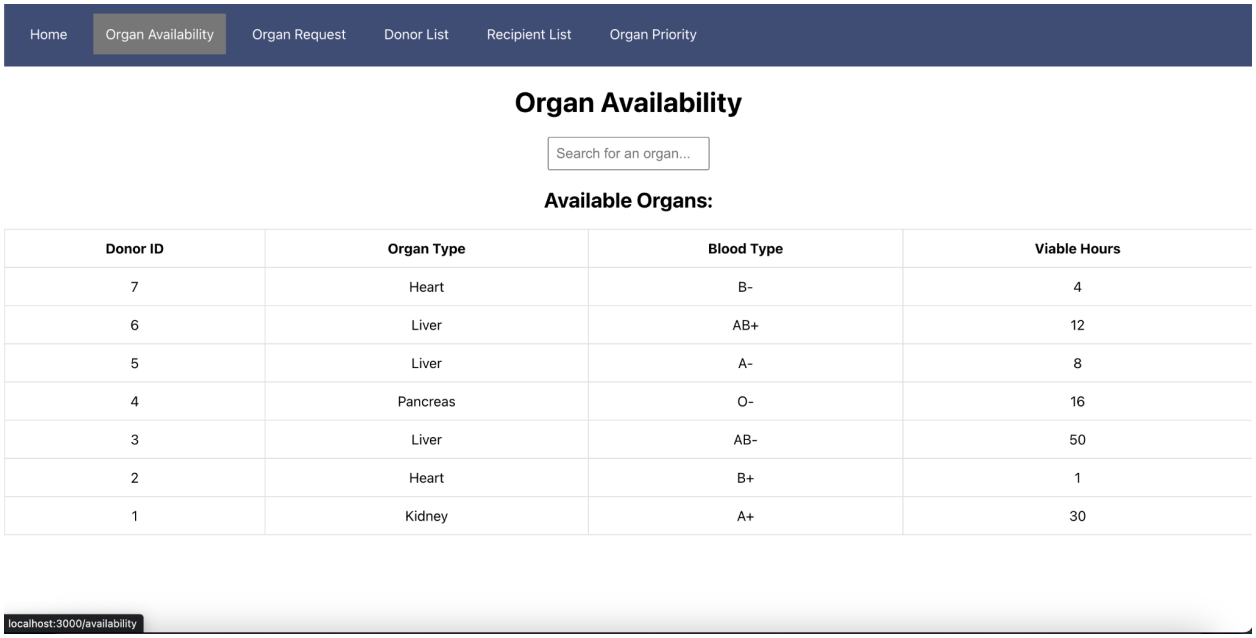

Figure 3: Organ Availability Screen

3.3.2 Organ Request Table

The organ request table contains all the requests made for organs. Within the table, information such as organ type, blood type, and recipient ID is recorded. This table is ordered in descending order of Request ID.

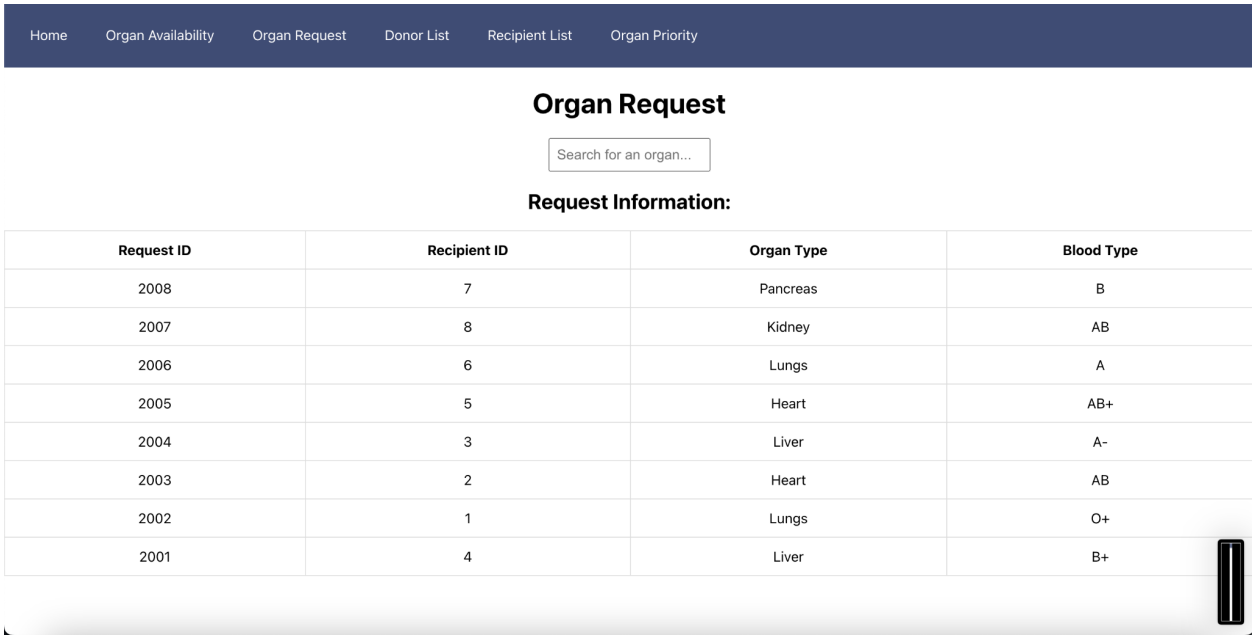

Figure 4: Organ Request Screen

3.3.3 Donor List Table

The Donor List table, in Figure 5, contains information such as the Donor ID, the First name of the donor, the Last Name of the donor, the age of the donor, and more. This table can be edited depending on the situation of the donors.

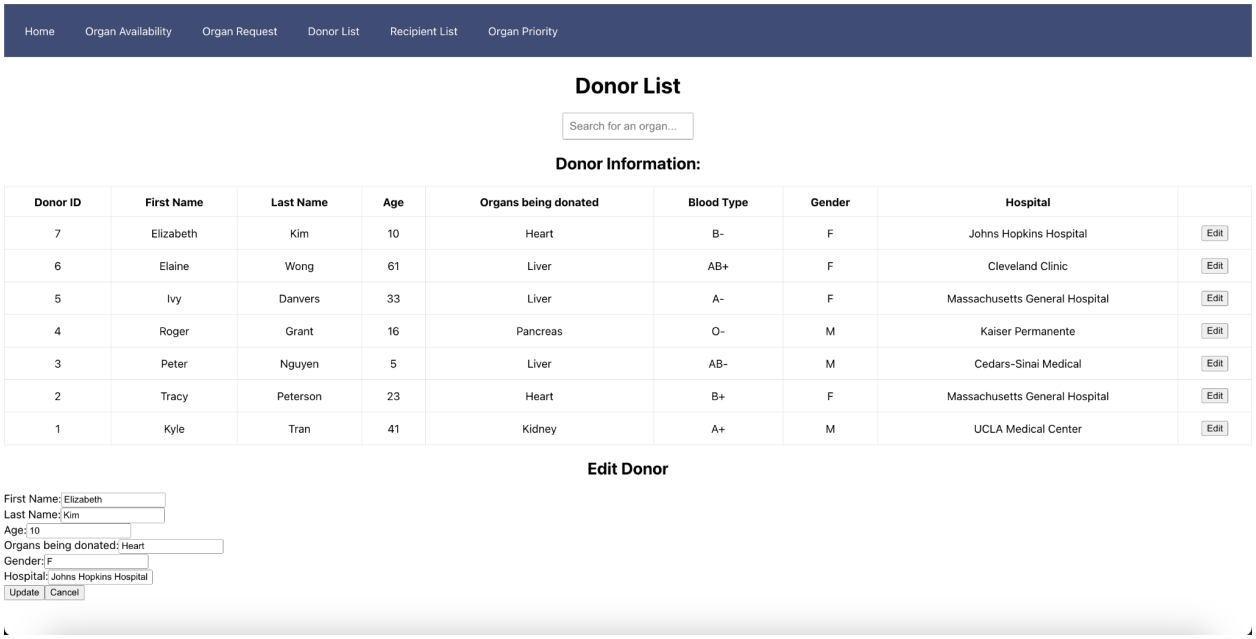

Figure 5: Donor List Table

3.3.4 Organ Priority Table

The organ priority table lists all the organ types within the system and their priority level based on how long they are viable. For example, the heart has the #1 priority since it has the least amount of viability time

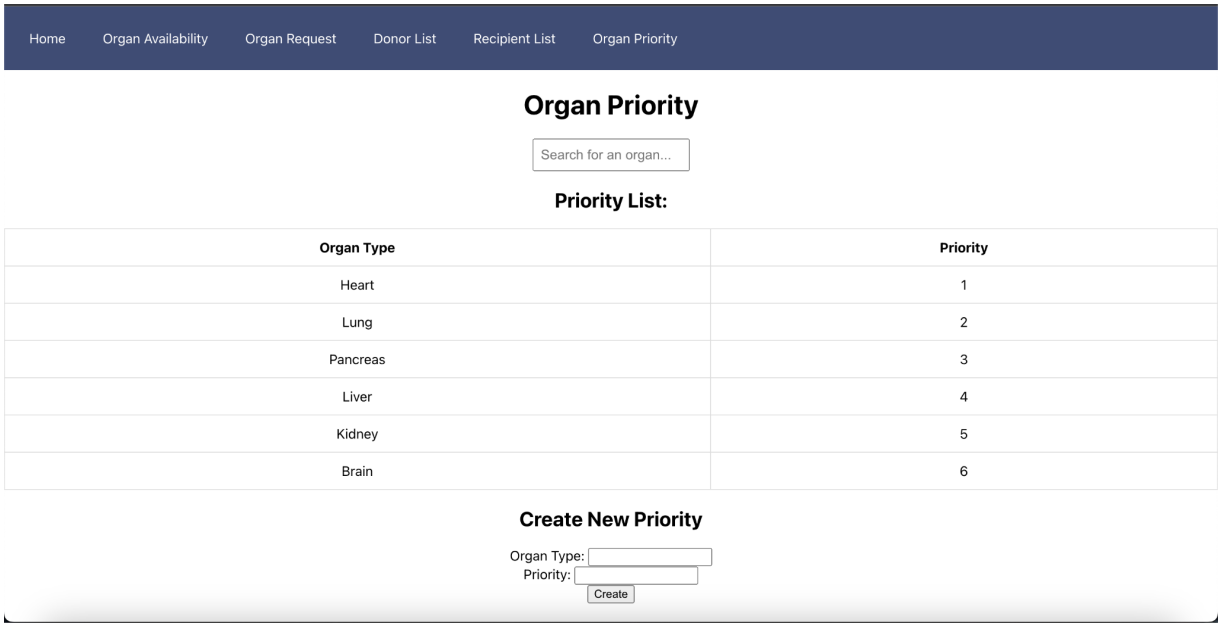

Figure 6: Organ Priority Table

3.3.5 Recipient Information Table

The Recipient table in Figure 7 shows various pieces of information, from Recipient ID to the age of the recipient, to the hospital they are staying at.

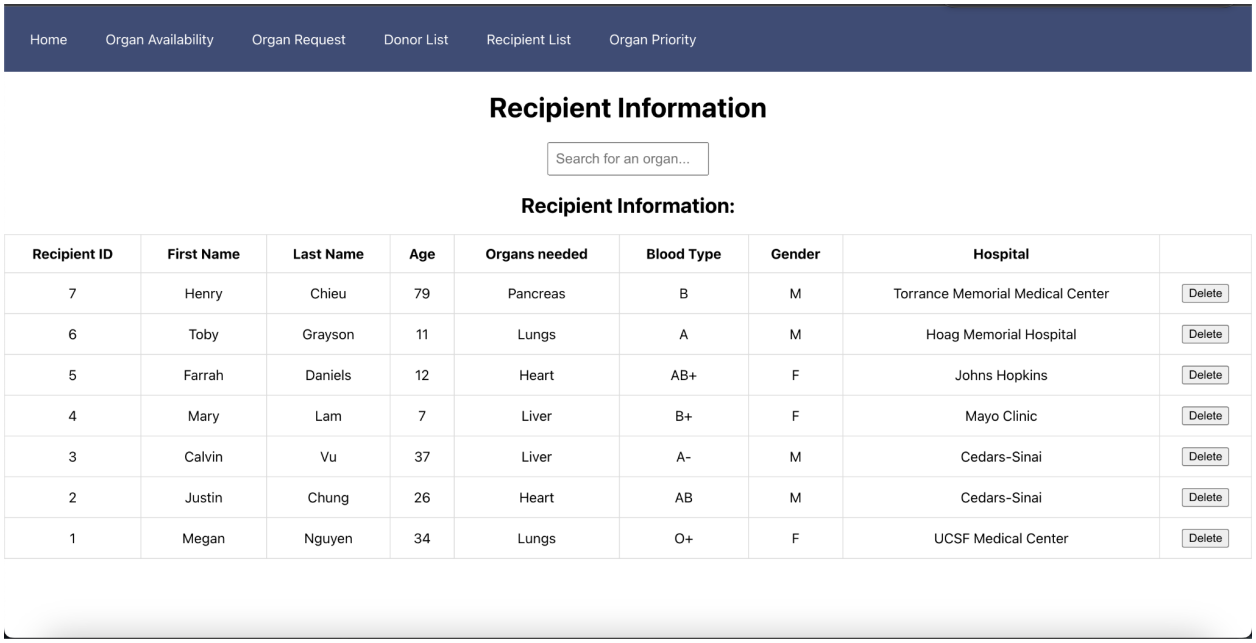

Figure 7: Recipient Information Table

4. Conclusion

After the completion of this project, we're confident in our ability to represent a real-world, needed application using a database. We were able to first of all represent patients as entities in SQL, as well as their personal information as attributes of said entity. We were then able to represent the relationships between particular attributes as certain tables. Our group had the opportunity to link MySQL with Node.js and recognize how each tool interacted with the other. Furthermore, we were able to relate what we learned in this course to the course project successfully.

4.1 Further Steps

To take this website to the next level, a couple of our next steps would be to first add the edit, delete, and add functions to every single table rather than only specific tables. Although our website is through an administration view, there is the worry that donors or recipients might make changes straight to the tables themselves since there is no security. To solve this issue, we plan to implement a login and sign-up feature. To make it even more detailed, there can be accounts for regular users and accounts for administration, or employees. After completing these steps, the final step would be to publish this website and advertise for commercial usage.

5. Contribution of Team

Eric Pham: Front end UI, backend connecting server with external database XAMPP, handling buttons, as well as testing functionality
Kate Dinh: Front-end design, handled databases and SQL
commands, as well as the presentation slides, and the project report
Nicholas Bao: Back-end database, presentation slides, project report