Revisions Table

Revision No.	Date	Sections updated
1	May 20, 2022	• Introduction • Functional Requirements • Non-Functional Requirements • Risks • Definition of Done • System Architecture • Software Test and Quality • Project Ethical Considerations • Project Completion Status
2	May 27, 2022	• System Architecture
3	June 3, 2022	• Software Test and Quality • Project Ethical Considerations • Team Profile
4	June 10, 2022	• Revisions Table • Definition of Done • Future Work • Lessons Learned
5	June 14, 2022	All – moved entire report to website format • Introduction • Functional Requirements • Non-Functional Requirements • System Architecture • Technical Design • Quality Assurance • Results • Future Work • Lessons Learned
6	June 17, 2022	Addressed review group's criticism • Introduction • System Architecture • Functional Requirements • Appendix

Table of Contents

Introduction

When pharmacists write prescriptions for some cancer patients, there is a risk that the patient may pass before they use their chemotherapy medication. In this instance, their loved ones may not be able to donate these chemotherapy medications to help someone else who needs that same treatment. At the same time, some of these medications can cost patients anywhere from $15,000 to $45,000 per bottle. To connect cancer patients with these unused medications, Good Shephard Pharmacy through their RemediChain program allows those with unused medication to donate their medication for reuse with their pharmacists able to provide this recycled medication to patients in need.

RemediChain's current system uses a simple spreadsheet to store information on medications and patients, which is insecure and time consuming to search through. Stemming from a desire to address these concerns, as well as to expand the service's reach, BurstIQ has been contracted to develop a more advanced system that uses blockchain technology to organize and secure this sensitive data. BurstIQ's proprietary blockchain implementation (BurstChain) is used to store assets representing medications and requests for those medications.

Previous field sessions have created a system where users can donate and receive medications just as on RemediChain's website, this time using the BurstChain to store the information. Our task was to enable a pharmacist user to accept those donations, update and maintain the inventory, and assign accepted medication to corresponding requests.

Functional Requirements

Functionality for a "Pharmacist" role. A user with this role must be able to:

- View donations submitted for consideration.
- View unfilled requests.
- Accept incoming donations:
    - Changing status to "Approved"
    - Moving ownership to the inventory
- Deny incoming donations
    - Changing status to "Unusable"
    - Unusable medications are not displayed in general inventory.
- Assign medications to a request
    - Change ownership of the medication to the recipient.
- Allow for changes to be made to medications in an inventory page.

Functionality for an "Admin" role. A user with this role must be able to:

- View a dashboard which displays information about donations, inventory, and requests

Non-Functional Requirements

- Only allow for authorized users to update assets.
- Provide a clear and simple way for a pharmacist user to interact with the donations, inventory, and requests.

System Architecture

The backend blockchain system behind RemediChain can be boiled down to a database of dictionaries. These dictionaries can have numerous fields and are all owned by an account (represented by a public ID). The owners of an asset are the public ID(s) of the account that owns the asset. In the case of a Medications2 asset, upon donation (and therefore entry into the chain), the donor owns the asset.

The pharmacist must interact with various assets, and using API calls, we can create, alter, and delete assets. An asset is a member of a dictionary created on the blockchain, and currently, the assets that need to be queried and accessed are users, medications, and prescriptions. These assets are all differentiated by the varying attributes of each asset, and a permissioned user can query assets based on these attributes.

Previous development implemented functionality for adding medications and requests to the BurstChain. In the case of medications, they are stored in the chain in the form of a “Medications2” dictionary. This dictionary contains information about medications, but for the purposes of this project, we only focused on the “status” and “owners” of the asset. Requests are stored in the chain in the form of a “Prescriptions” dictionary, which contains information about the patient and the medication they are requesting. For the purposes of this project, we only focused on the “status” and “owners” of the asset.

To interact with donations, inventory, and prescriptions, the pharmacist needs to be able to view medication assets depending on status. On the website, this is done when the pharmacist clicks on a button linking to a specific page where a script calls the functions needed to query the assets. Once these assets are queried, the pharmacist then needs to click on a specific asset and change the status attribute of the asset. These changes are then applied through another API call.

Pharmacist flow diagrams:

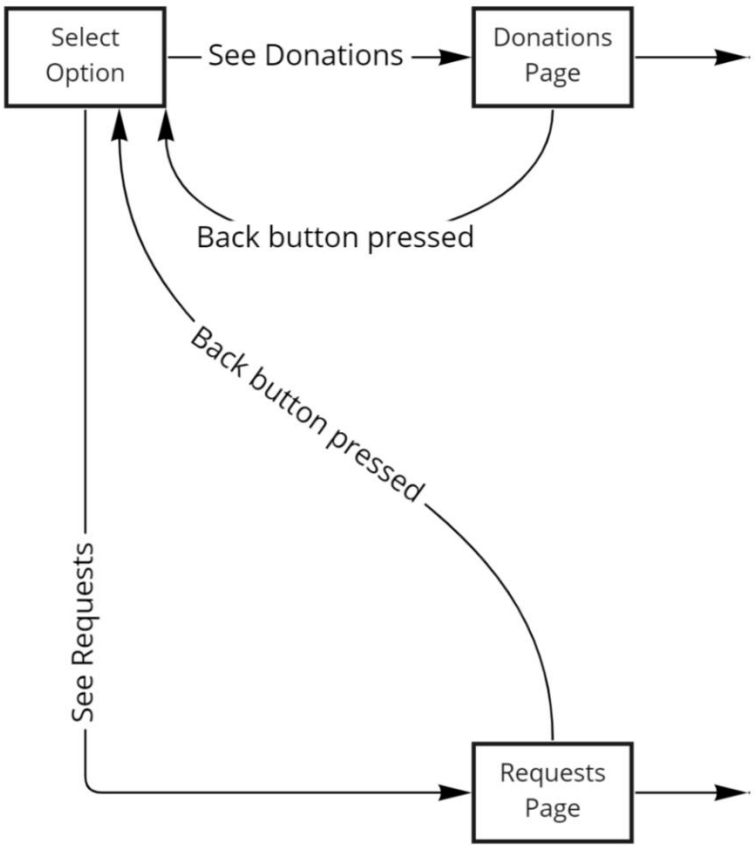

Figure 4: Initial Pharmacist Choice

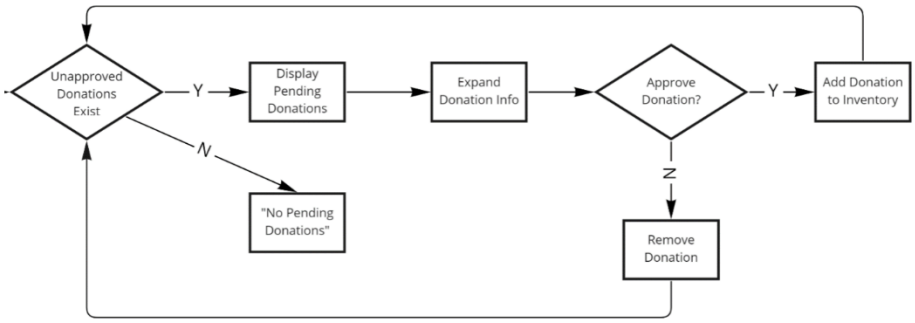

Figure 5: Flow for Approving Donations

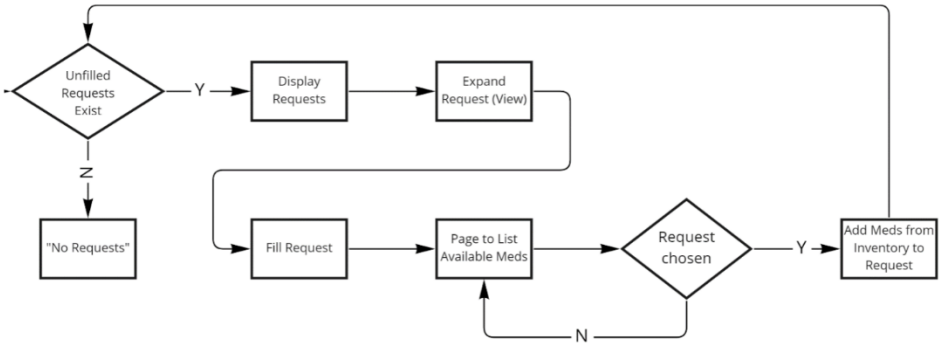

Figure 6: Flow for Filling Requests

UI design:

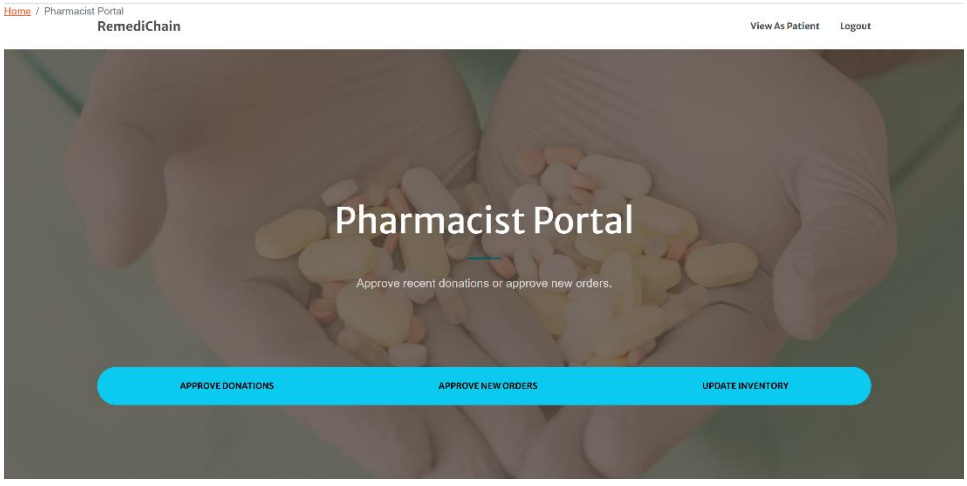

Figure 8: Pharmacist Portal

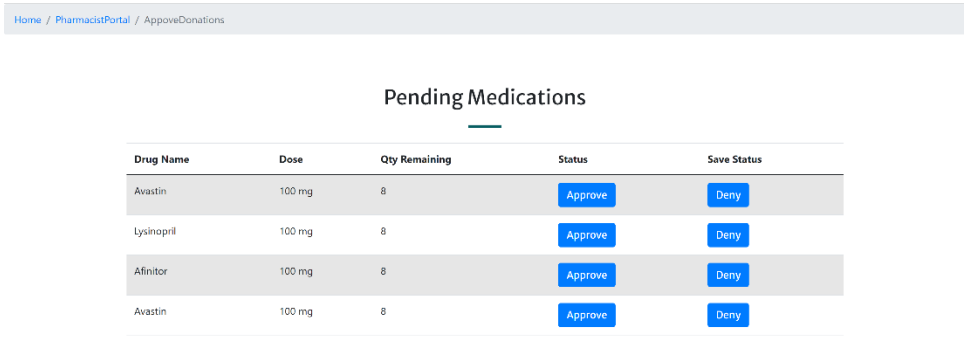

Figure 9: Accept Donations Page

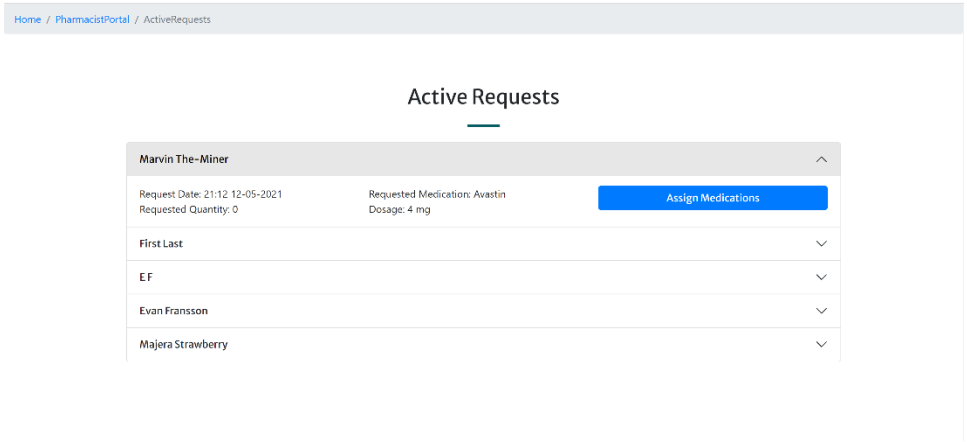

Figure 10: Active Requests Page

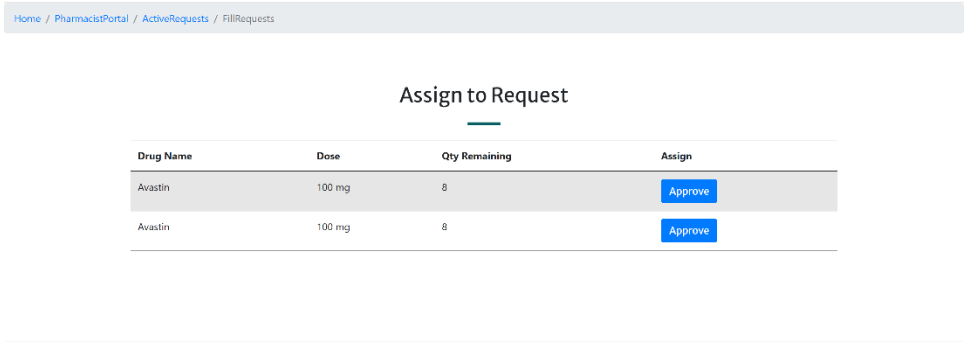

Figure 11: Assign Medications Page

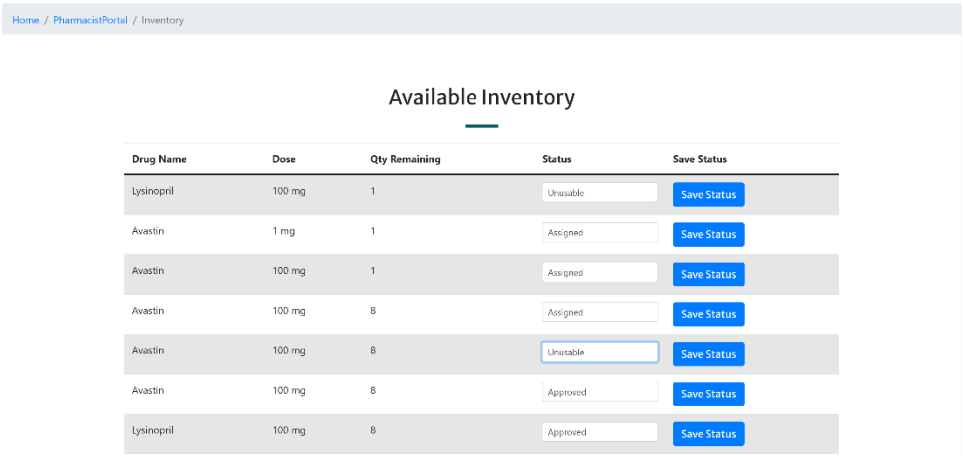

Figure 12: Inventory Page

Technical Design

Status Updates:

The status of an asset represents what stage a medication or request is in currently. A Medications2 asset currently has 3 possible statuses:

- "Pending": When the medication is submitted to be considered for donation.
- "Approved": When the medication is approved to be shipped to RemediChain.
- "Unusable": When the medication cannot be reused (opened packaging, incorrect information, etc.) and should not be shipped to RemediChain.

Figure 1 depicts an example asset with the status “Pending” (sensitive information is blacked out):


```
{
    "hash": "1df12e585db6283d305ed5ff439c76d7976f11ba94098806ad1d0464b8e27d1f",
    "timestamp": {
        "$date": "2022-06-06T21:43:05.400Z"
    },
    "type": "asset",
    "operation": "create",
    "owners": [
        [redacted]
    ],
    "signer": [redacted]
    "signature":
        [redacted]
    "dictionary": "Medications2",
    "asset": {
        "drug_name": "DB1",
        "dosage": 200.00,
        "quantity": 3,
        "expiration_date": {
            "$date": "2223-01-01T00:00:00.000Z"
        },
        "NDC": "110",
        "form": "123",
        "manufacturer": "j&j",
        "lot": "10",
        "monetary_value": 1000.00,
        "status": "Pending",
        "dosage_unit": "mg",
        "donor_id": "donor_id",
        "recipient_id": "recipient_id"
    },
```


Figure 1: Example Medications2 Asset

Similarly, a Prescriptions asset currently has 3 possible statuses:

- “Pending”: The request is unfilled and can have medication assigned to it.
- “Approved”: The requested medication has been assigned to the user.
- “Denied”: The request is unable to be filled.

To update the status of an asset, an API call is made, the body of which is changed to reflect the change in status. When a pharmacist approves a donation, a “Pending” donation is changed to “Approved”, and an ownership transfer (which will be explained shortly) takes place. When a pharmacist denies a donation, a “Pending” status is changed to “Unusable”.

Ownership Transfer:

A Medications2 asset is owned by different users depending on its status and physical location. In our implementation, when a donation is initially submitted for consideration, it is owned by the donor's account, represented by the presence of the donor's public ID in the "owners" field of the asset. Throughout the workflow of a pharmacist user, the owner of a medication asset changes according to the timeline laid out in Figure 2.

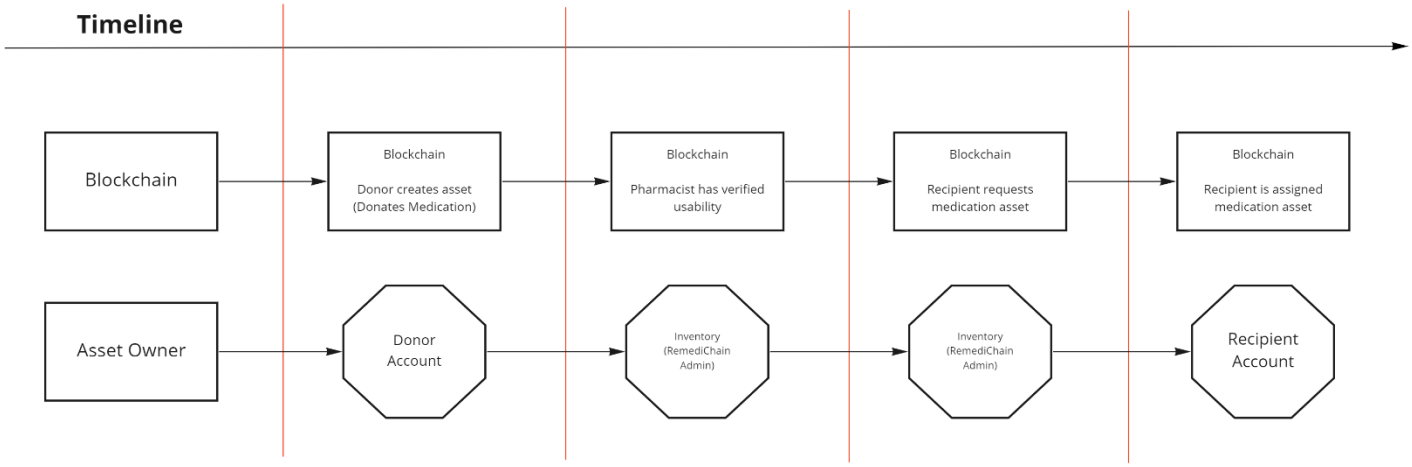

Figure 2: Timeline of medication asset ownership

When a medication is approved for donation, ownership of the asset is transferred to the inventory. The inventory is represented by an administrator public ID that remains constant. In the current implementation of the ownership transfer function in this case, ownership is removed from the previous owner (the donor) of the medication. If a medication is denied for whatever reason, ownership of the asset is changed. Only the status is updated.

When a medication is assigned to a recipient, ownership of the asset is transferred from the admin public ID to the recipient's public ID. This represents that the medication has been reserved and is no longer available to be assigned to another request.

Quality Assurance

Unit Testing

The functions at the core of the project are those that handle ownership transfer and status updates. The following functionality was verified using the Postman and Insomnia API clients that:

- When the ownership transfer function was called, the owner of an asset was updated.
- When the status change function was called, the status of an asset was updated.

User Interface Testing

The UI must be appealing and navigable. It was verified that:

- Where to click to reach each service was clear.
- Styling was consistent with that of the existing pages created by previous sessions.

This was done through simple visual inspection.

Integration Testing

Interacting with certain pages must result in certain actions on the back end. It was verified that:

- Clicking on any button on the “approve donations” page resulted in asset updates.
- Changing status of a medication in the inventory page using a drop-down menu and submitting changes updated asset status.
- Assigning medications to an order resulted in ownership transfer.

This was done through manual interaction with webpages and breakpoints in the developer tools of a browser.

Code Reviews

To allow for the code to remain understandable, all members of the team reviewed the code that implemented major changes. This was done in person.

Testing Table:

Test Purpose	Description	Tools	Definition of Acceptable	Edge Cases	Results of Testing
Ensure that medication status can be changed in the inventory page.	A medication asset’s status will be changed from every status to every other status	VSCode Live Server Extension, Postman/Insomnia, BusrtIQ API	The asset is able to be changed from any status to any other status, and this is verified with the API	A medication asset is changed from an Approved status to another status.	Change was successful across every possible status change.
Ensure that the UI looks as it should on a variety of screen sizes	Several different resolutions, screen sizes, and browsers used to access the website.	Chrome Firefox Edge Vivaldi Opera	The UI doesn’t overlap itself and is generally pleasing to look at on every tested screen.	A user may be using a very old browser without support for some modern features.	Website displays all important attributes when each page is compressed, expanded, or displayed on an older browser.
Ensure that the BurstChain can be queried, and the results can be displayed on various pages.	The BurstChain will be queried using BurstIQ’s API and the results will be verified. The Inventory, Requests, and Donations page	Postman/Insomnia Web Browser Console User Interface BurstIQ Framework	Every asset queried (several of every type) returns all data. There shall not be any missing assets that are supposed to appear.	-Query invalid statuses. -Change asset status and query very quickly after -Querying duplicate assets.	No results return when querying via an invalid status. When querying for valid statuses, all entries under the status show up. When an

	will be visually inspected to ensure that information populates correctly.			-Querying non-existent assets -Querying assets with invalid authentication	asset status is changed and the assets are queried again, the query is updated. If there are duplicate assets, both are shown. If the asset name does not exits, nothing is queried, and if the user does not have credentials, then the BurstChain throws a 400 error signifying a bad request.
Ensure that a Pharmacist can change donation status to approved or unusable.	Both a valid and invalid donation will be submitted through the donations page. One of these donations will be accepted, and one will be denied.	VSCode Live Server Extension Chrome Developer Tools	The approved donation is visible in inventory, the rejected donation is not visible in inventory, both donations are no longer visible on the pending donations page.	User changes the status and then changes it back to pending.	The status of every tested medication asset was successfully updated to “Approved” upon a button press. Status was successfully updated using the drop-down menu on the inventory page.
User Interface Ease of Use Ensuring that a Pharmacist can navigate to every necessary page.	An attempt will be made to reach every page relevant to a pharmacist’s workflow. The result of clicking on navigation buttons will be noted. Redundant navigation options will be searched for.	VSCode Live Server Extension	All pages that a pharmacist will need to see can be reached.	Multiple roles leading to loss of privileges.	The pharmacist’s workflow was demonstrated in practice. Every page was accessible, and it was clear (to our team) how to get to each page.

Table 1: Table Describing the Testing Process

Results

Implemented:

- A function for transferring ownership of any asset to a new owner, optionally allowing for the current owner to be kept.
- A function for changing the “status” field of a medication asset.
- A “pharmacist portal” allowing a pharmacist user to reach all relevant pages.

- A page for the pharmacist to view, as well as approve or deny pending donations.
- A page for the pharmacist to view open requests.
- A page for the pharmacist to assign medications to a request.
- A function for querying information about inventory to display on the admin dashboard.
- A partial admin dashboard page that displays very basic information about medications in inventory.

Unimplemented/Incomplete:

- Polish on website styling and total UI consistency.
- Displaying all information about donations and order that the client desires.
- A sign-up page.
- A "sort-by" feature to order the donations and requests tables.
- A full-fledged admin dashboard that displays in-depth information about medications in inventory.

Future Work

A major consideration when dealing with medical data that was stressed is information security. It is important that only those who are authorized can view the data present in the RemediChain system. The current implementation of API calls within functions leaves the admin's private ID exposed. If this were available for anyone to see, then they could make authorized API requests to access sensitive data. Further work should be done to ensure that this security risk is addressed.

A more comprehensive format of displaying data on donated medication, medication in inventory, and open requests could be implemented to help make a pharmacist's work clearer. Currently, a minimal amount of information is conveyed to the pharmacist about a donation. Alterations to the display function could be made to allow for a unique identifier to be displayed, for example.

Due to the nature of having three separate field sessions working on this project, there is some friction between the styling of pages made by each group. A general overhaul of the UI would likely make the user experience of the pharmacist as well as a potential donor or recipient more enjoyable.

Currently, the statuses that are available for a medication asset are "Pending", "Approved", and "Unusable". According to the client, a more comprehensive system of statuses is desired to allow for more stages of the donation and assignment process to be represented ("Shipped", "Expired", "Assigned"). This should be considered when adding functionality for accounting for donation shipment.

Lessons Learned

- Organization is vital. We found that having more structured in-person collaboration sessions helped us get work done more quickly and made sure we all understood what had been completed. We were able to discuss changes and determine the best course of action for every change and addition. In

addition, we used the app ClickUp for a scrum managing app, and it helped keep all of us on task and reporting to each other when we finished our individual work.

- It's important to determine specific requirements early. We missed out on some development time trying to figure out specifically what to implement, and there was some early confusion about what specific functionality needed to be in place to meet requirements. Once we organized our work and began to discuss this, specific tasks became clearer.

- Only a single member of our group had any experience with web development. Everyone gained experience with not only the technologies used for this project (web APIs, HTML, CSS, JavaScript) but learning new technologies during a project as well. It is very likely that during our careers we will have to both learn new technologies and use them simultaneously. This gives us experience with that process. Stack overflow, contrary to our initial perceptions, is quite commonly used in the industry to gain further insight and learn.

- We got lots of new experience with web API development. We had to form post, get, and put requests to be able to interact with a database that BurstIQ had developed. We learned that this is how most websites interact with each other, and how data is transferred on the internet: both through HTTP communication and through Webhooks.

- We gained experience with the Bootstrap CSS framework, as well as the security risks associated with running a webpage that is manipulable by JavaScript and/or injection.

- Working with a blockchain was a major selling point of this project for all of our team members. Most of our collective knowledge on blockchain was through cryptocurrency, which uses a different type of blockchain than what is used for medical data. With BurstIQ we learned about this type (private permissioned) and what makes it secure. We also learned about the principle of a distributed ledger: a database that is shared and synchronized across multiple sites and is accessible by multiple users that have access to any one of the nodes that is being communicated with in the database

Team Profile

Members: