SOFTWARE DESIGN DOCUMENT

Cardinal: Blood Bank Communication Software

SWE 3313 | Group #1

Andujar Brutus | Pamir Ahmad | Audrey Allen | El Arbi Belfarsi | Thomas Breitung

Version 1.0

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of Cardinal, a software application for real-time cataloging of blood bank resources as well as connecting banks to the medical professionals and donors that work with them. Cardinal's interested parties and future users may use this document to learn of the intended design of the application.

1.2 Scope

Cardinal is intended to establish real-time resource tracking and communication between blood banks and the people who utilize their services. Donors will be able to view which bank is the best option for them, based on factors such as distance and need for resources. The blood banks themselves will use the program to catalog their resources, as well as keep track of the expiration dates of those resources. When the medical professionals using Cardinal request blood or plasma, they will be prioritized by order of expiration as to utilize as many resources as possible. This software will be used exclusively for blood banks; Cardinal is not planned to be applied to other medical facilities or means of resource management.

By reaching the objectives listed above, Cardinal will help blood bank administrators utilize their donations in the most efficient way possible. Medical professionals will be able to find banks quickly and easily with the resources they need for medical procedures. Donors will also be able to see where they can most easily donate, as well as where their services are most needed. By accomplishing these goals and encouraging more people to donate to these facilities, Cardinal aims to combat the effects of low donation numbers.

1.3 Overview

This design document shows how Cardinal will be implemented through its architecture, class and object structure, database usage, and human interface design. The system architecture section is broken down into smaller subsections to help detail the types of technologies used, how they are designed, and why they have been chosen. The detailed design and human interface design sections rely on diagrams to demonstrate the class and object structure of the application, as well as basic examples of what users may see while utilizing the interface. Lastly, the requirements matrix serves to refresh readers on the desired requirements for this application, as well as to show how those requirements involve users, functions, and use/test cases.

1.4 Reference Material

https://careerfoundry.com/en/blog/ux-design/what-is-a-wireframe-guide/

https://maze.co/blog/ui-vs-ux/#what-is-ui-design

https://project-management.com/requirements-traceability-matrix-rtm/

https://www.slideshare.net/DerekCunningham1/requirements-management-matrix-done1-42896960

1.5 Definitions and Acronyms

2D: Two Dimensional

API: Application Programming Interface

DBMS: Database Management System

GUI: Graphical User Interface

MS: Microsoft

SQL: Structured Query Language

2. SYSTEM OVERVIEW

After launch, Cardinal will be used by three primary parties: blood bank administrators, medical professionals seeking resources, and donors seeking to donate. Cardinal is intended to function on any operating system, but it will be a priority to optimize it for systems that are already being used in banks in other medical facilities so as to make launch and initial use as simple as possible. Blood bank administrators will have the ability to update information regarding their resources in real time through use of a database. Using the information from this database, donors and medical professionals will be able to enter their information into a GUI and receive a list of banks sorted by factors such as proximity, need, and resource amount, depending on the user. Unique profiles will be created for each user. The locations of facilities and donors will also be recorded, and an algorithm within the application will provide travel time estimates to each facility.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

SUBSYSTEMS:

3.1.1. Donation center locator

Role: will contain necessary information such as the nearest locations of donation centers, phone numbers, hours of operation, and blood volumes available. This information is displayed on a page using a Map/center locator API.

3.1.2. Blood availability identifier

Role: allows users and centers alike to know the blood availability of any participating center. This information is stored on a public database. Can be viewed from center locator API.

3.1.3. Donation center priority detector

Role: features that will help individual users or organizations find the centers that require more donations of a specific type. The necessary information is stored on a public database. Can be viewed from center locator API.

3.1.4. Personal information database

Role: Common personal data is stored for each user to be queried for donation center prioritization. This information is stored in a private database due to HIPAA regulations.

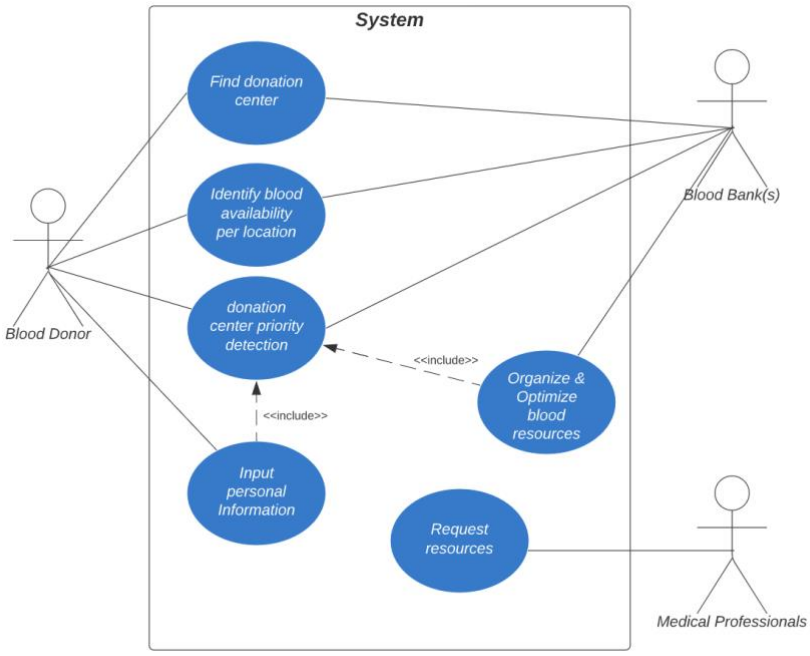
3.1.5. Blood resource organizer/optimizer

Role: Organize a donation center's current resources. This information is stored on a private database to protect from sensitive information leaks. Parts of the private database will be queried for blood center prioritization.

3.1.6. Resource request

Role: Request a donation center's current resources. The participating medical professionals and centers exchange information pulled from public or private databases depending on security clearances and information sensitivity.

(cont.)



3.2 Technical design

JavaFX: it is a Java based technology that supports building reliable desktop applications for different operating systems using powerful APIs written in native Java code. Since those APIs are built using native Java code, they are runnable over the JVM making them multi-OS friendly.

Java Swing: for the GUI, we will be using Java Swing GUI toolkit because of the more sophisticated graphical components it offers over the classical Abstract Window Toolkit.

MS SQL Server: dealing with multi-row relational databases, we will be using SQL Server to perform data retrieval and storage requested by the Java desktop application.

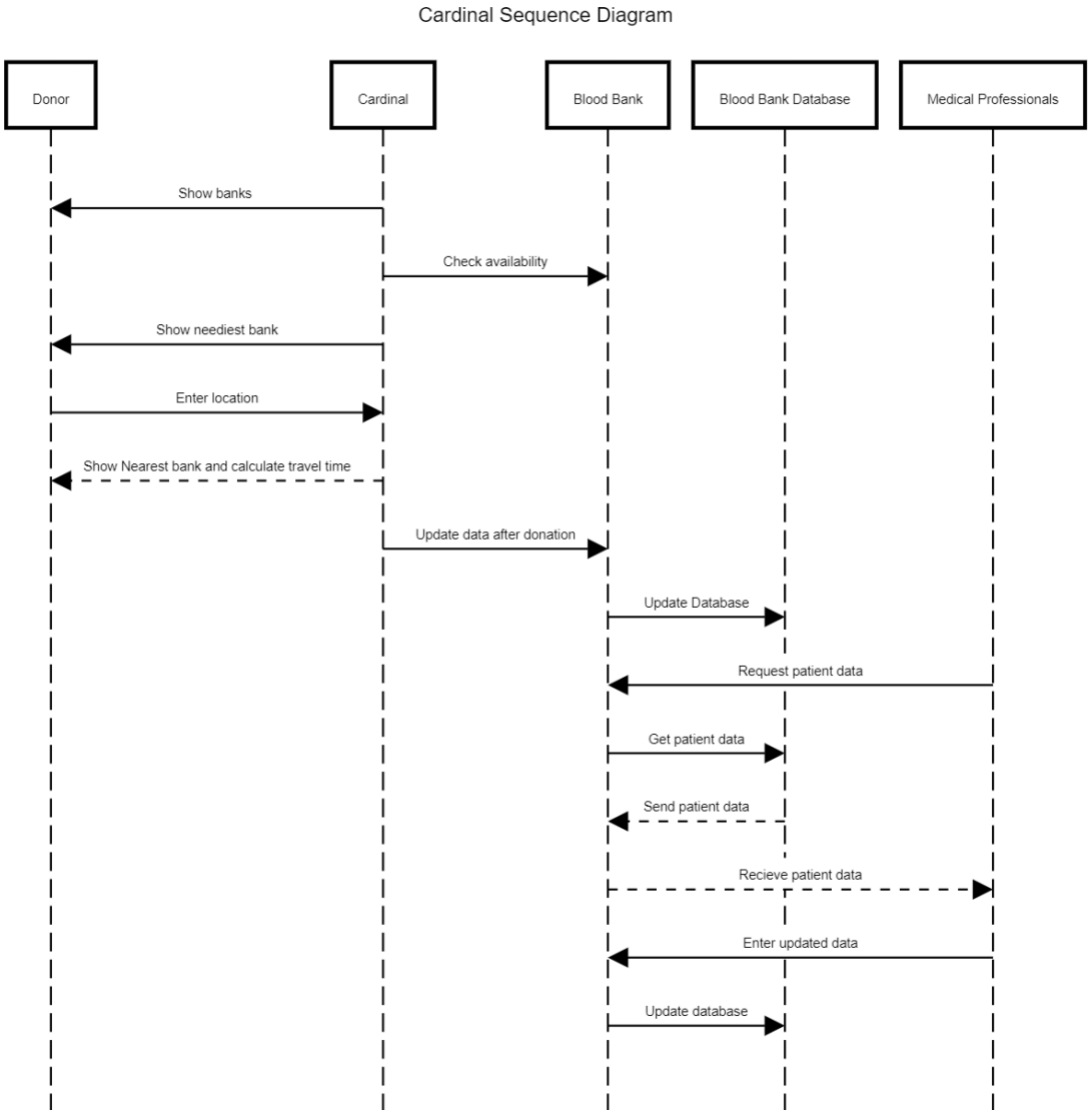
3.3 Design Rationale

The closest architectural pattern to our architecture is a combination of the Client-Server and the Data-centric architectures. In Cardinal, we have different clients interacting with the server due to the nature of the need it is responding to. Also, we will be following a data-centric approach as the database is heavily interacted with by different user groups. When it comes to technical design, we chose those technologies for different reasons including ease-of-use, reliability, and performance.

4. DETAILED DESIGN

4.1 Class diagrams

4.2 Other diagrams



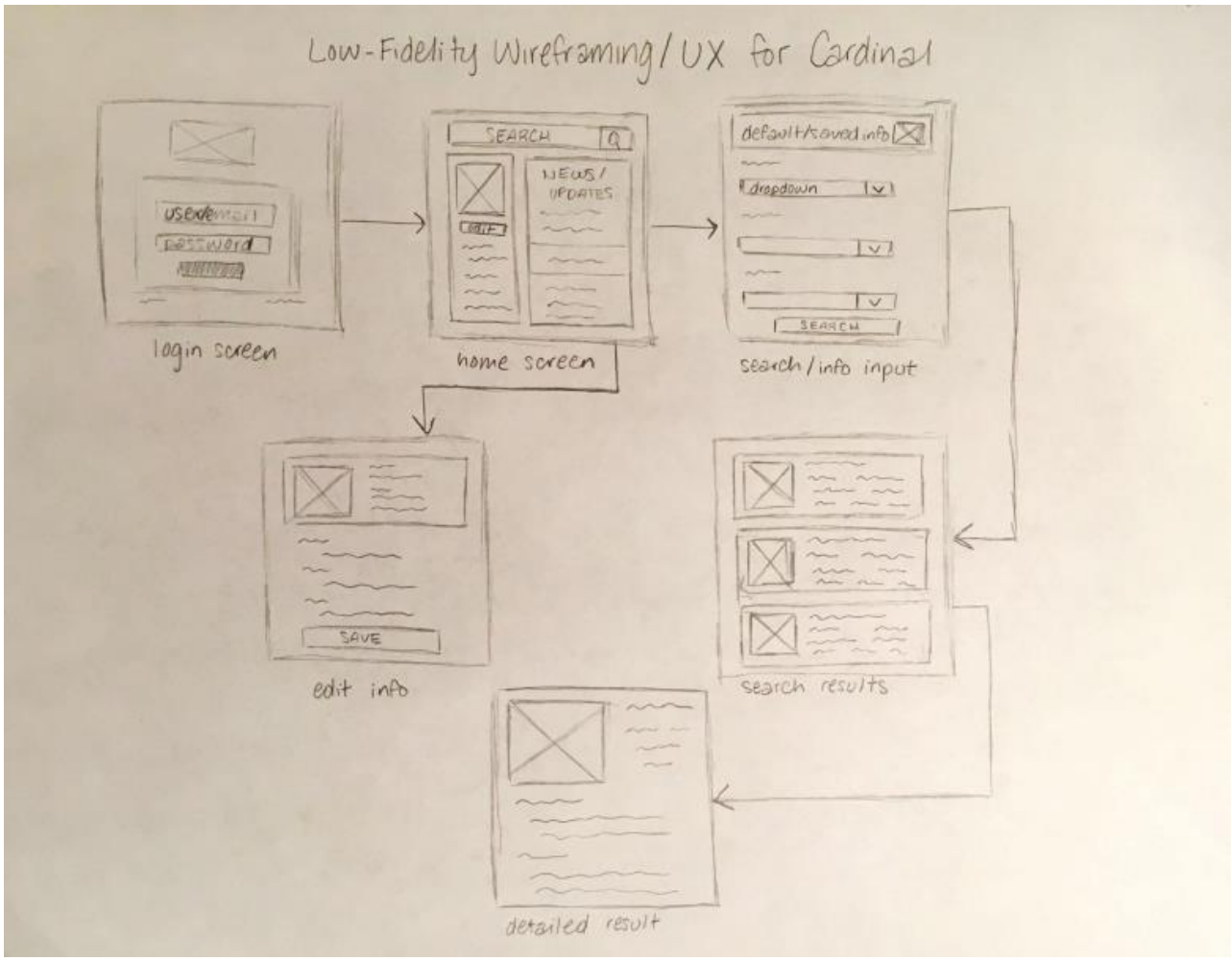
5. DATA (DATABASE) DESIGN

The main data structures used will be arrays and linked lists. Arrays will mainly be used due to its efficiency, and since you can quickly grab data and make a table (when considering 2D arrays) to help send into our Database. After storing it into our 2D array we can then store it into a DBMS (Database Management System) easily since it is already formatted well. It will be complete chaos if we do not properly set up a good DBMS to manage all the data since the main purpose of our app is to be the "middleman" and distribute information (data) that we have collected. We will need to store hospital names, blood types, members, dates, and plenty more. The DBMS needs to be carefully set up to where it is efficient to use since the most important functions of the web app will need to pull information out of the database to present to the client. We will be using MYSQL to store our data in from the

hospitals and use it to retrieve data to our clients. We will be using the Relational Model as it saves data in a table format, and this is what the group is mainly familiar with, and it also works the best with our given data.

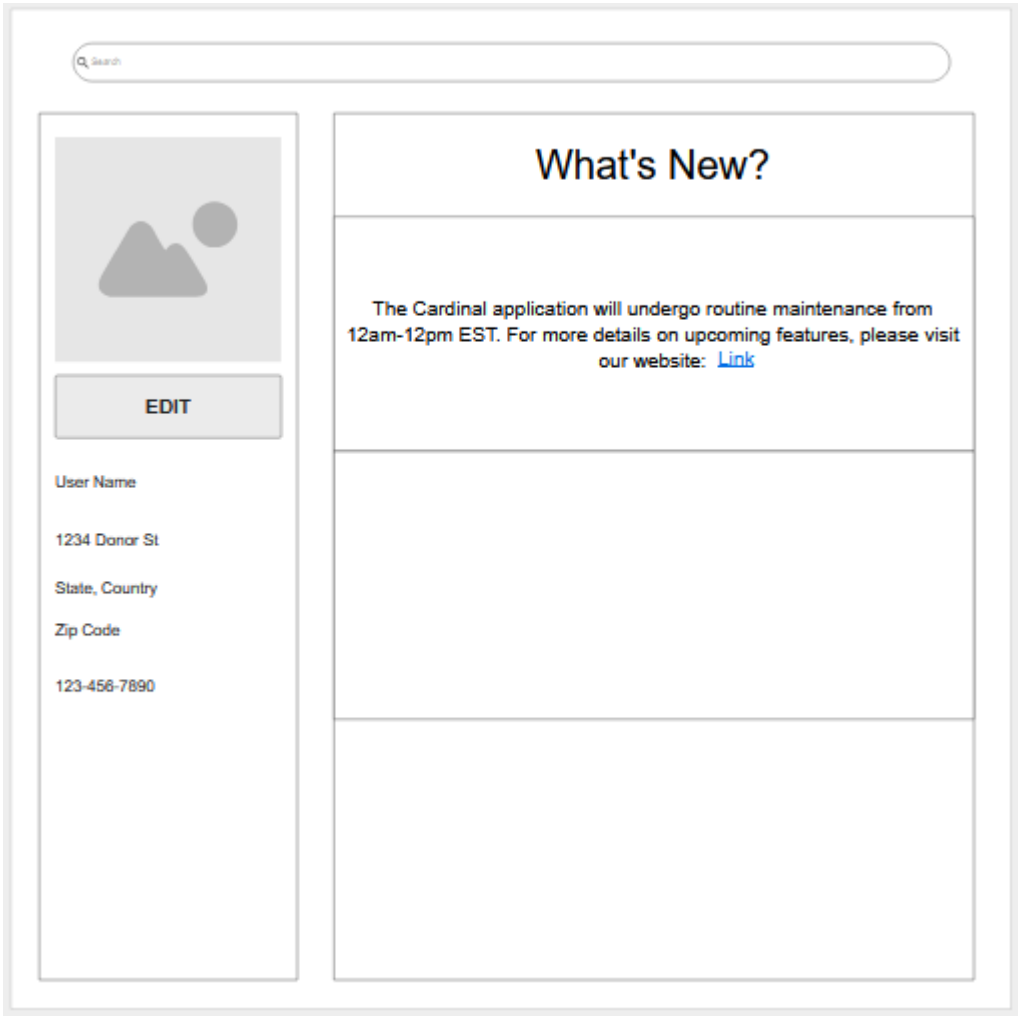
6. HUMAN INTERFACE DESIGN

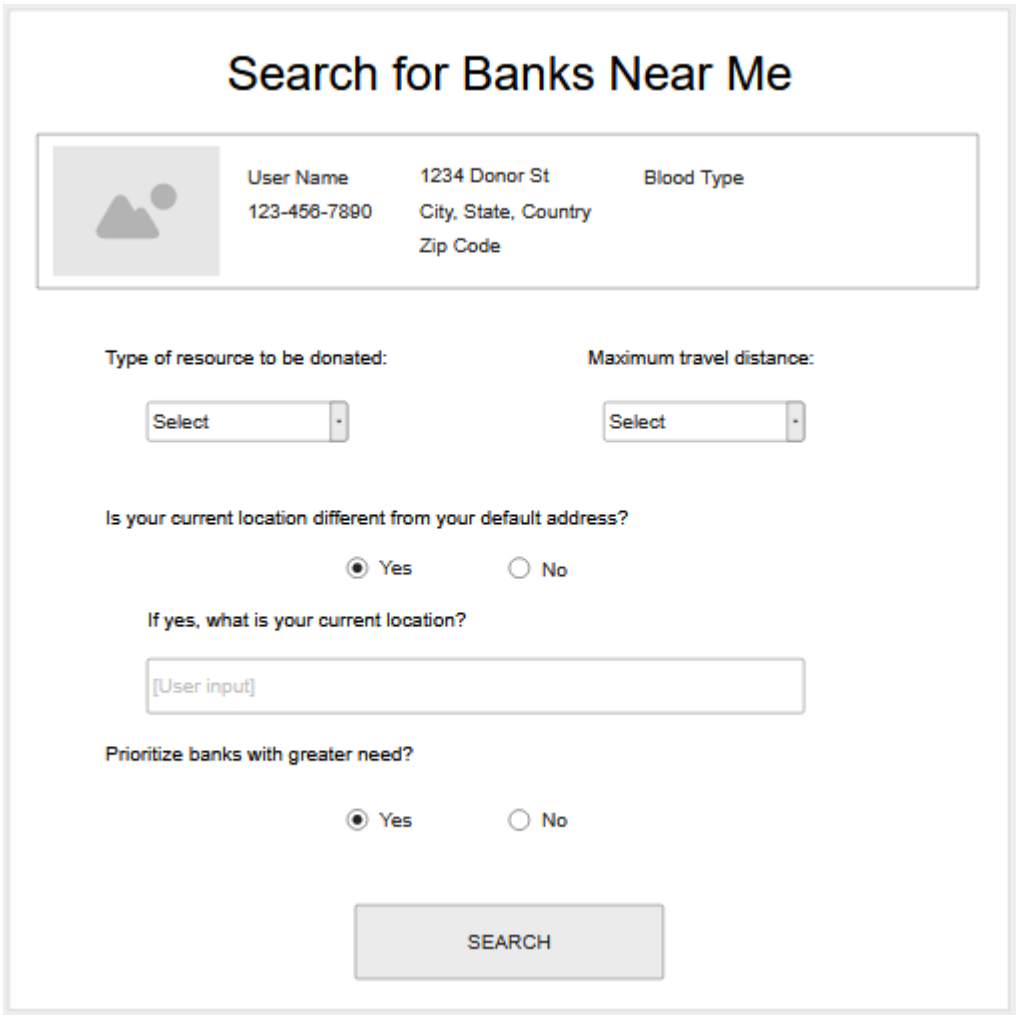
6.1 UX design



[Wireframe/UX figure showing Cardinal's pages and flow of use.]

6.2 UI design


Search
EDIT
User Name
1234 Donor St
State, Country
Zip Code
123-456-7890
What's New?
The Cardinal application will undergo routine maintenance from 12am-12pm EST. For more details on upcoming features, please visit our website: Link



[Examples of UI for Cardinal's home and search pages for a donor-type user.]

7. REQUIREMENTS MATRIX

ID	User Type(s)	Requirements Description	Requirements Objectives	Use Case	Test Cases
1	Donors, Blood Banks, Medical Professionals	Center locator	Helps Locate donation center on map API	Find donation center	TC_1
1.1		Map API	Center locator	Find donation center	TC_1.1
2	Donors, Blood Banks, Medical Professionals	Blood availability per location	Helps users identify centers most in need of overall donations	Identify blood availability per location	TC_2
2.1		Private Database	Stores donation supply information and medical records	Identify blood availability per location	TC_2.1
3	Donors, Blood Banks, Medical Professionals	Donation center prioritization	Helps users find centers in need of specific donation	Donation center priority detection	TC_3
3.1		Public Database	Stores information for center/blood prioritization	Donation center priority detection	TC_3.1
4	Donors, Blood Banks, Medical Professionals	Personal information input	Common personal data is stored for each user	Input personal information	TC_4
4.1 (2.1)		Refer to ID 2.1	Stores donation supply information and medical records	Identify blood availability per location	TC_2.1
5	Blood Banks	Organize resources	Organize a donation center's current resources	Organize and Optimize blood resources	TC_5
6	Medical Professionals	Resource request	Medical organizations request a donation center's current resources	Request resources	TC_6

8. Appendices