Bilkent University
Department of Computer Engineering

Senior Design Project

Helping-Hand

High-Level Design Report

Mustafa Anıl Taşdan, Metehan Gürbüz, Rola Younis, Hassam Abdullah

SUPERVISOR: Dr. İbrahim Körpeoğlu

Jury Members: Dr. Shervin Arashloo, Dr. Hamdi Dibeklioglu

Dec 24,2021

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2

1. Introduction

As mankind progressed through the ages, the technology we possessed and our understanding of science progressed alongside us. Medicine was one of the fields that saw the most progression. Many medical conditions that may have resulted in fatalities in the past have simple solutions to them today. One big development that proved crucial to the field of medicine were blood transfusions. The first human-to-human blood transfusion was performed in 1795. [1] Since then, blood transfusions have saved countless lives, but the amount of daily blood donations is still very little. Blood transfusions are used for many medical conditions, such as anemia or cancer, but more importantly they are crucial for blood loss resulting from accidents. [2] As donated blood can only be stored for 42 days, and with only a few percent of the population donating their blood[3], a steady supply of blood donations is needed every day. This can be further proven by the fact that Bilkent University periodically sends requests for emergency blood donations. If our community consisting of 14.000 people needs to send this many requests, it is easy to grasp the grimness of the situation when we enlarge the scale to the whole population of Turkey.

This is where we, as HelpingHands, want to come in. We want to provide a mobile based solution for connecting recipients and donors. We want to provide a platform where people in need, or hospitals, can put requests for types of blood donations, and donors fitting the provided criteria can be alerted by notifications, so that they can contact the recipients easily.

1.1 Purpose of the System

1.2 Design Goals

Our design goals are straight-forward. We aim to make a user-friendly and intuitive application that prioritizes functionality over flashiness. The reason behind it is simple, the mobile application is supposed to be a health-critical app and therefore must not be counter-intuitive and clear for users. Here are some goals which are worth consideration related to the system:

1.2.1 Usability

- All text will be larger than or equal to 1rem.
- In case of images not loading, an alt text will appear as a placeholder instead of the images.
- All text will have at least 4:5:1 contrast ratio with the background.
- The app should notify the users of important notifications via the mobiles' notification layer whenever necessary.

1.2.2 Performance

- The system will respond to operations by all users within a span of 2000 ms.
- The web application should be able to handle the traffic of 1,000 concurrent users.

1.2.3 Security

- The GPS information obtained from the user must be used only when necessary and kept confidential and only used with consent.
- All user personal information must be stored encrypted.
- Data must not be shared with third-party applications.

1.2.4 Reliability

- An email verification or phone number verification should exist in the system so that fake accounts are decreased, especially since this is a health critical app.
- Should ensure that donors are notified of nearby donees requests promptly and with accuracy.
- Should ensure that teams/social media/awards functionalities are effective since the app relies on the donors being benevolent and these functionalities act to draw them.

1.2.5 Efficiency

- The mobile application should be less than 100 MB to make it easier to store for users and redundant files via updates should be removed.
- Should work on old smartphones and majority of the popular mobile OS's (Android) and iOS versions since majority of users will be in critical need of blood at any given time.

- Should eliminate user profiles that have not been used for a long time period especially donees(recipients) by sending them a warning email to attain better server capacity efficiency.

1.3 Definitions, Acronyms and Abbreviations

Helping-Hand: The name of the application that our team chose.

Donor: The benevolent person who will be donating his/her blood.

Donee: The person in dire need of blood. In other words, the recipient of the blood given by the doner.

App: Short form for ‘application’. In this report’s context, it relates to a mobile application that will be used by the donor and donees.

Client: Essentially a front interface that is interacted by the users of the mobile application which he/she can use to perform operations related to donor or donee.

Server: The backend of the mobile application that will communicate with the database and apis to store and retrieve user information.

AWS: Amazon Web Services.

OS: Short form for operating system, in this case, the one that android and apple devices use.

iOS: The operating system used by Apple phones.

2. Proposed Software Architecture

2.1 Overview

In this section, the proposed software architecture is presented in detail. This will give insight into the relation of the components and subsystems and how they interact with each other. The decomposition of the two subsystems namely, client(Mobile application) and server will be detailed in the following sections. In addition, certain technologies that will be used in the project subsystems will be delved into.

2.2 Subsystem Decomposition

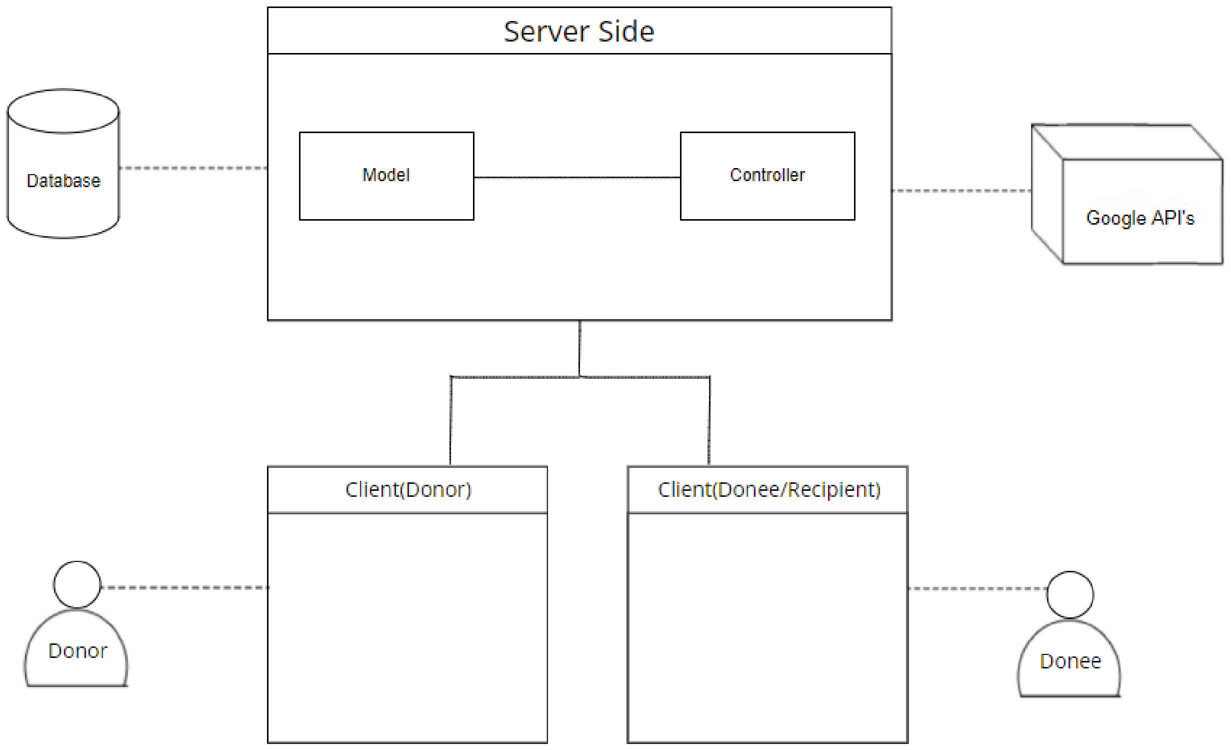


Figure 1: Subsystems

2.2.1.1 Model

The model subsystems will be responsible for interacting with the database connection and all database operations such as updation, deletion and addition will be through it. Therefore, it will act as a means to store data and provide relevant functionalities pertaining to the blood donation process.

2.2.1.2 Controller

The controller subsystem will handle the main logic and functionalities of the server side and will therefore act as a bridge between the user interface and data. The functionalities that the Controller subsystem will deal with include functionalities such as locating the user's location via the interacting with the Google Geolocation API, locating nearby blood banks via interacting with the Google Maps API and directing nearby blood donee (Recipient) requests to potential blood donors.

2.2.2 Mobile Application

The mobile application will be a client-side subsystem that both blood recipients (donees) and blood donors will use. Therefore, depending on the account logged in, it will provide core functionalities such as making/modifying blood requests, view nearby blood banks, view blood journey, creation of teams etc. Hence, this subsystem will communicate with the Controller subsystem on the Server side and is also one of the view sides of the MVC architecture.

2.3 Hardware-Software Mapping

The Client-side of the system is separated into two parts namely the Client(Donor) and Client(Donee/Recipient) side. The first client side will deal with providing the functionality pertaining to the donor while the second client-side will provide the functionality pertaining to the donee. Both sides will run on mobiles smartphones with GPS and will run on both android and ios since it will be developed using react native. The server side will be developed using Spring and a NoSQL database will be used to store the data, MongoDB. Furthermore, the interaction between client and server side will be handled via the Express framework.

2.4 Persistent Data Management

In Helping-Hand, we need to store comprehensive user data such as: name, blood type, contact information, connections to other users and blood banks; blood bank data such as location, connected blood requests, donor-recipient information and much more. So, we need to store this data in such a way that it is permanently and properly stored and readily available. For this purpose, we plan to use MongoDB, which is an object-oriented NoSQL database. We plan to use MongoDB because it complements the object models we plan to use nicely and it also is simple to connect to our backend-frontend data flow, in which we plan to use JSON objects. Lastly, we plan to host our database and backend on a cloud hosting service, such as AWS, to achieve permanent availability.

2.5 Access Control and Security

In Helping-Hand, we plan to have three types of accounts:

- User: Creates and responds to blood requests.
- Blood Bank: Manages related users and blood requests.
- Administrator: Has access to server management tools.

After account creation, a user account will be able to access their own information stored on the database. They will be shown their name, blood type, interaction history and interaction-related statistics. With a blood request creation, basic information of the responding donors and recipients will be shared between these users. They will have the ability to communicate through a text-based service provided by our platform. Upon confirmation by the recipient, extra information will be shared between the two parties, such as location and contact information; this information will also be shared with the blood bank that the recipient wants the donation to happen in. Access control and sanitization of queries and data flow will be controlled by the backend application. It is critical that only the intended amount of information is shared with the mentioned parties, as the application will store vital personal information.

User authentication can be achieved by both Google Authentication or Helping-Hand's own authentication system. The user will be able to choose between the two. Additional external platforms may be implemented in the future.

2.6 Global Software Control

Helping-Hand will use event-driven software control for both the client and server-side operations. The client will generate events upon user input, such as account creation, data requests, blood requests and so on. The server will handle events that are created upon data transmission from the client, and will decide on what to send back upon confirmation with the database. The system will stay in the same state if there are no inputs generated by any of the currently active users. The client may be updated upon events generated by other clients, but the server by itself does not have the ability to change its state.

2.7 Boundary Conditions

2.7.1 Initialization

As the server and the database will be hosted on a cloud based service, they will always be initialized, barring events such as maintenance, software updates or failures.

The clients will first have to download the mobile application from a hosting platform that is not yet determined. When they launch the application for the first time, they will be prompted with an account creation interface. There are three possibilities, the user may create an account, the user may login using a Google account, which will be verified with the Google Authentication API, or if the user is representing a blood bank, they may fill out a form to request a special account creation from our team. If the user falls into the first two categories and successfully completes these steps, they will be granted access to their personalized interface.

If the user is already registered with the platform, they may login using their login information or go through Google Authentication. If the verification is successful by the server, they will be sent an access token and will be able to access the application.

2.7.2 Termination

Closing the application is not expected to have any complicated effects on the application. If the user does not allow it, their login information will not be stored on the application and they will be automatically logged out, so they will have to go through authentication each time they launch the application. The user can also log out using the user interface, which will cause all personalized data stored on their device to be deleted.

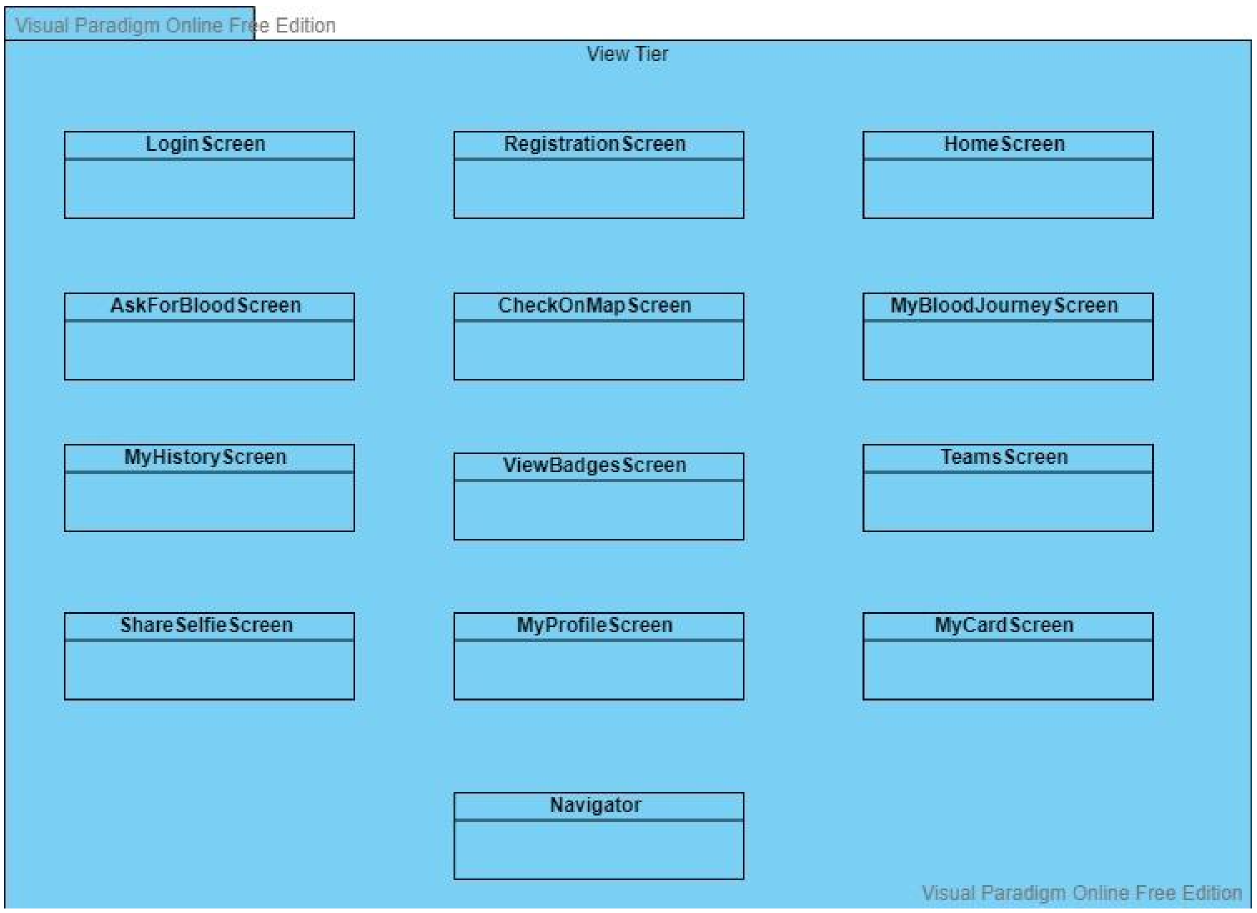
2.7.3 Failure

The application will try to exit gracefully upon any kind of perceived failure, such as failed authorization, data connection issues or server issues. The user will be prompted with a message if there is a failure. Changes made by clients will be immediately sent to the server, so no data loss is expected if a problem occurs on the client side. Nevertheless, developmental errors may be introduced during the coding of the application and the user may face a sudden crash, we aim to eliminate these errors as they arise.

3. Subsystem Services

3.1 Client

3.1.1 View Tier

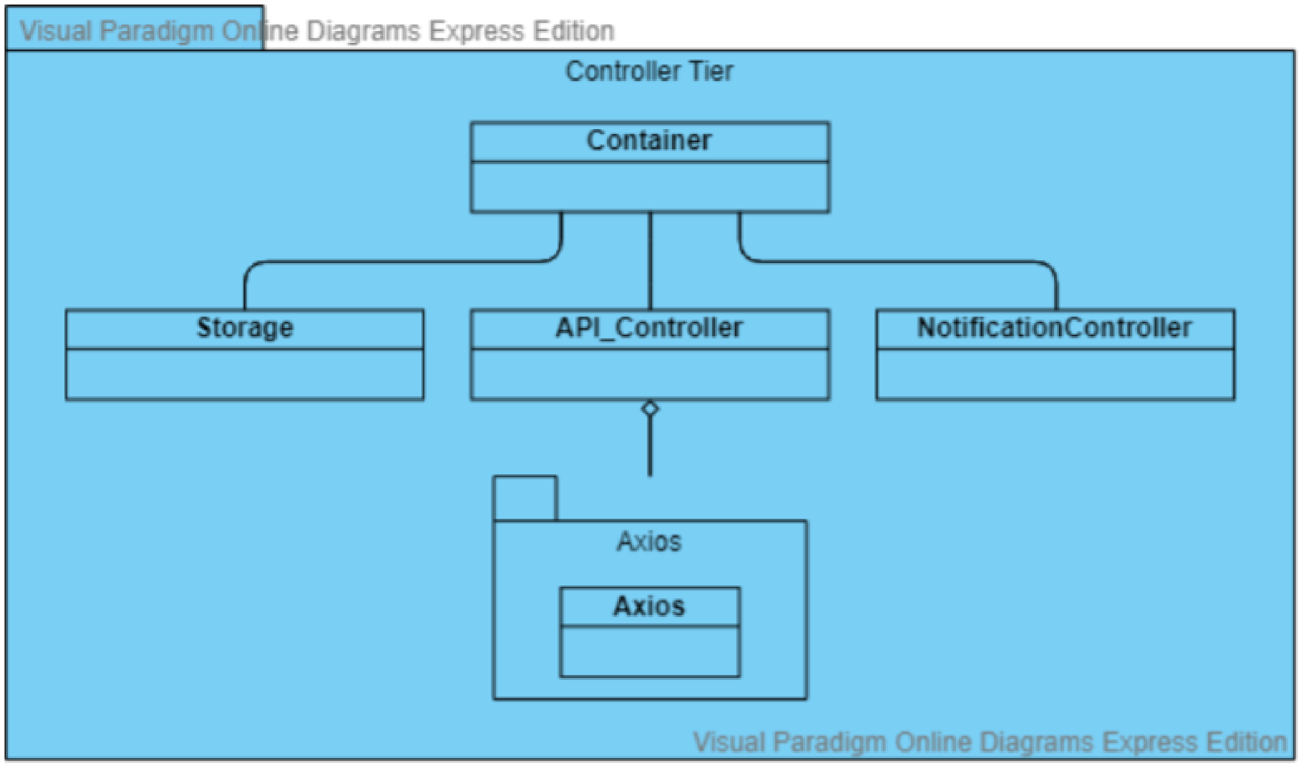


View tier is the subsystem for the UI components.

Class	**Description**
Navigator	This class is the navigation system of the application. It is responsible for the switches the user makes among screens.
LoginScreen	This class provides the login components to the user.
RegistrationScreen	This class provides the registration components to the user.
HomeScreen	This class provides the main menu to the user where he/she can go to other screens.
AskForBloodScreen	This class provides the option for blood reservation.
CheckOnMapScreen	This class provides the option for blood donation.
MyBloodJourneyScreen	This class provides the blood journey of the blood sample for the donor.
MyHistoryScreen	This class provides users their own blood history.
ViewBadgesScreen	This class provides users with badges as how much they interact with the application.

TeamsScreen	This class provides users the ability to create and join teams.
ShareSelfieScreen	This class provides the users to take a selfie from the app and share it on social media.
MyProfileScreen	This class provides the users to change their profile information.
MyCardScreen	This class provides the users their own and unique blood cards.

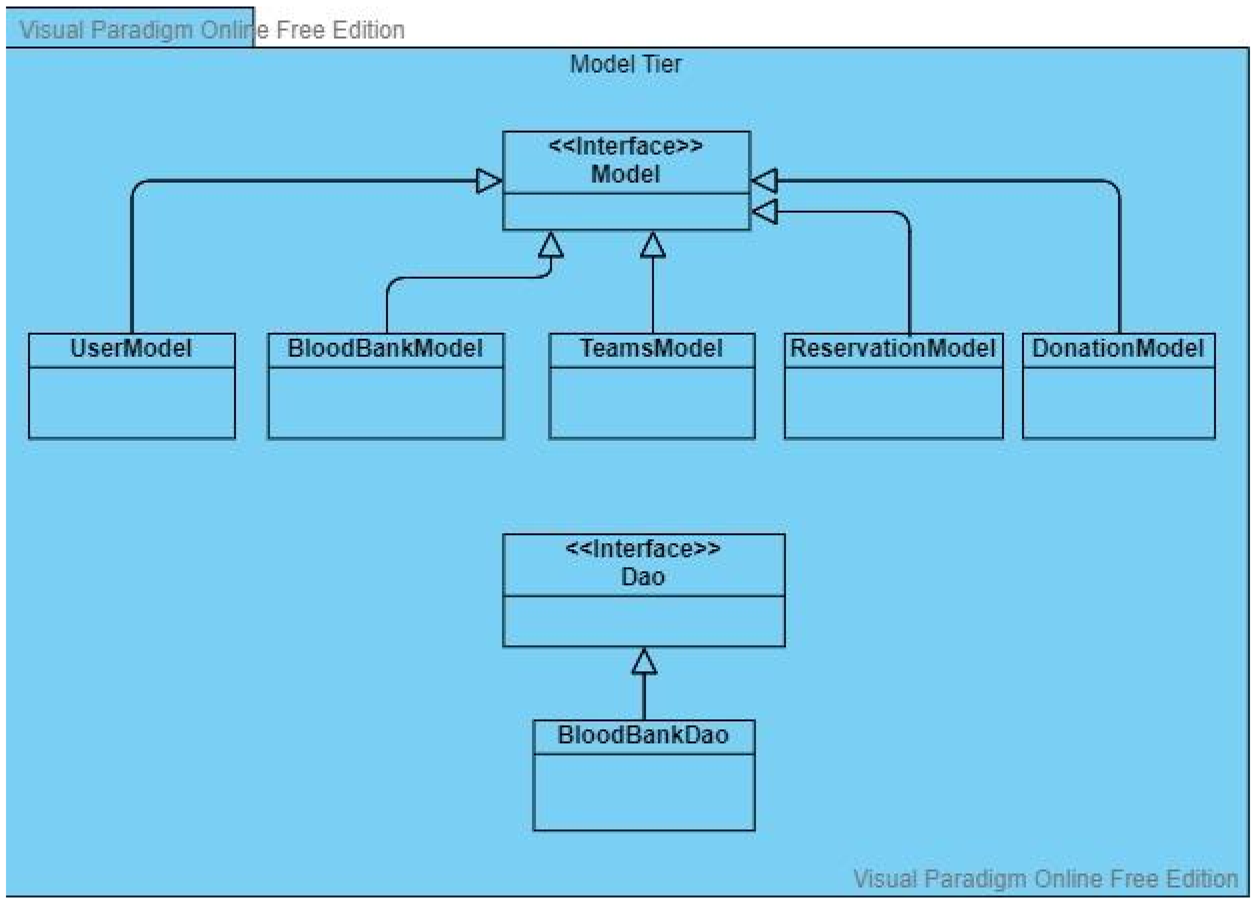
3.1.2 Controller Tier


Visual Paradigm Online Diagrams Express Edition
Controller Tier
Container
Storage
API_Controller
NotificationController
Axios
Axios
Visual Paradigm Online Diagrams Express Edition

Controller tier is the subsystem in order to handle events, requests, and responses.

Class	**Description**
Container	This class provides UI classes access to local storage, a server connection, private state, and a notification listening service.
Storage	This class provides access to the device's local storage. Any sort of data can be encrypted and preserved.
API_Controller	This class provides sending http requests -get, post, put, delete- to the server via 25 instances of Axios class.
NotificationController	This is a singleton class that manages the app's notifications.
Axios	This class is part of the Axios third-party package and is able to send http requests to a defined URL and port [4].

3.1.3 Model Tier



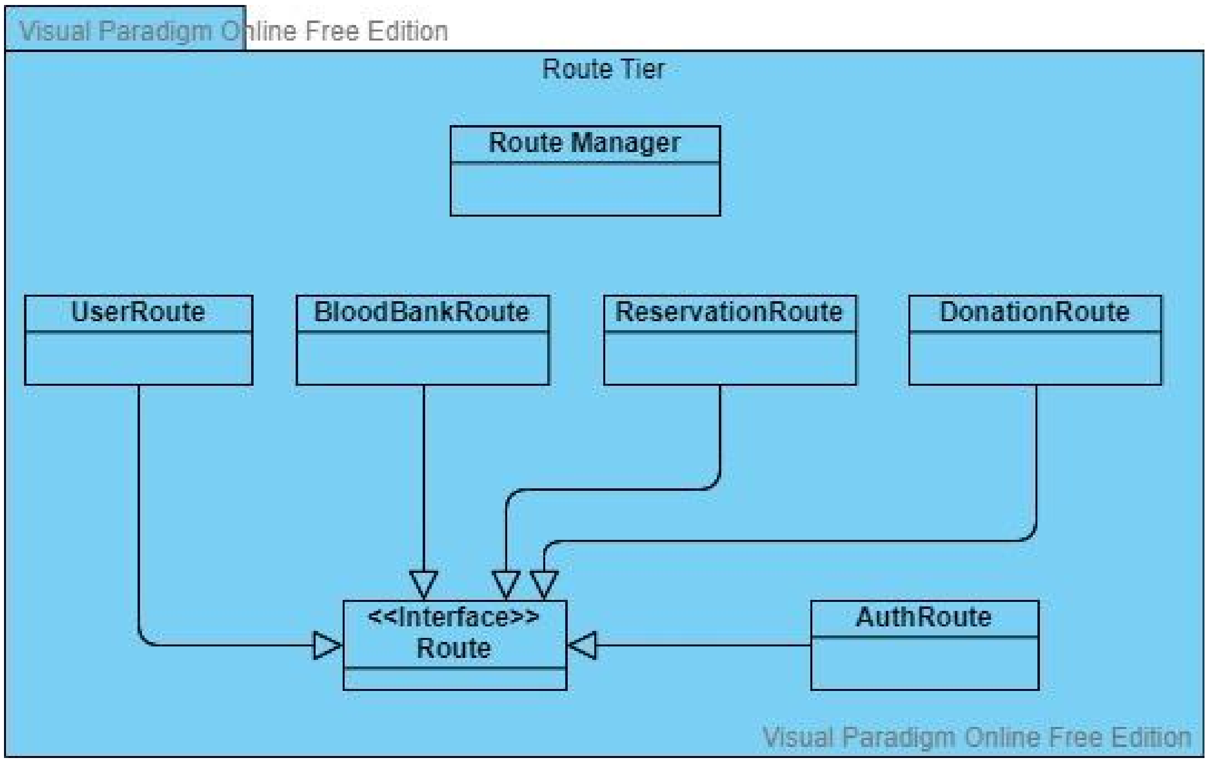
Model is the subsystem for data elements in order to communicate easily and reliably.

Class	**Description**
Dao <<interface>>	Dao: Data Access Object. It isolates the layer of business from a relational database by providing abstract API [5].

BloodBankDao ~ *Dao*	This class provides a Dao for BloodBankModel class.
Model <<Interface>>	This interface contains get and set methods.
UserModel ~ *Model*	This class contains a model of a User object.
BloodBankModel ~ *Model*	This class provides a model of a BloodBank object.
TeamsModel ~ *Model*	This class provides a model of a teams object.
ReservationModel ~ *Model*	This class provides a model of a blood reservation object.
DonationModel ~ *Model*	This class provides a model of a blood donation object.

3.2 Server

3.2.1 Route tier

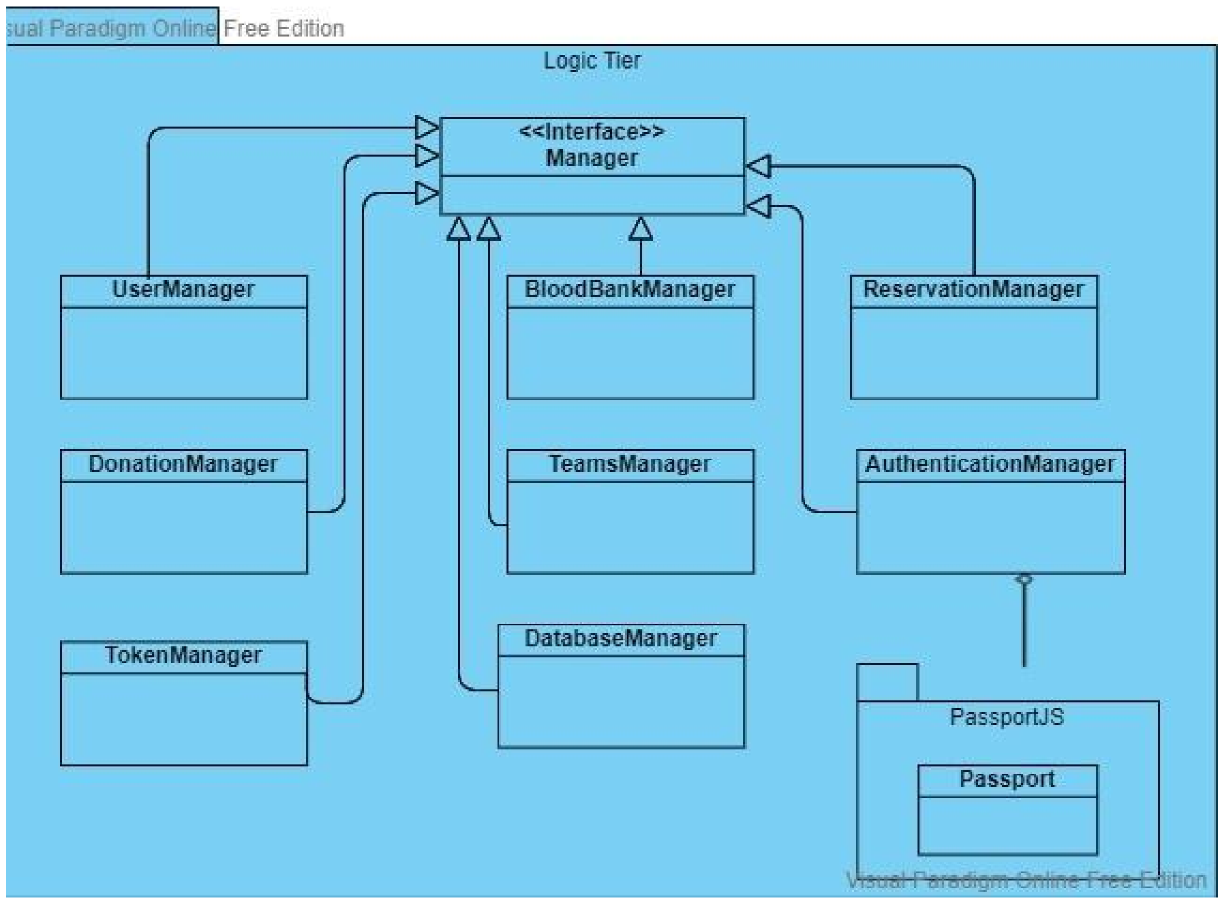


Route tier is the subsystem for the route elements in order to provide the concept of rest api.

Class	**Description**
Route<<Interface>>	This class manages http requests which are post, get, delete, and put.

UserRoute ~ *Route*	This class manages http requests to the path “/user/*”
BloodBankRoute ~ *Route*	This class manages http requests to the path “/bloodBank/*”
ReservationRoute ~ *Route*	This class manages http requests to the path “/reservation/*”`
DonationRoute ~ *Route*	This class manages http requests to the path “/donation/*”
AuthRoute ~ *Route*	This class manages http requests to the path “/auth/*”
RouteManager	This class provides the server management of the routes.

3.3.2 Logic Tier

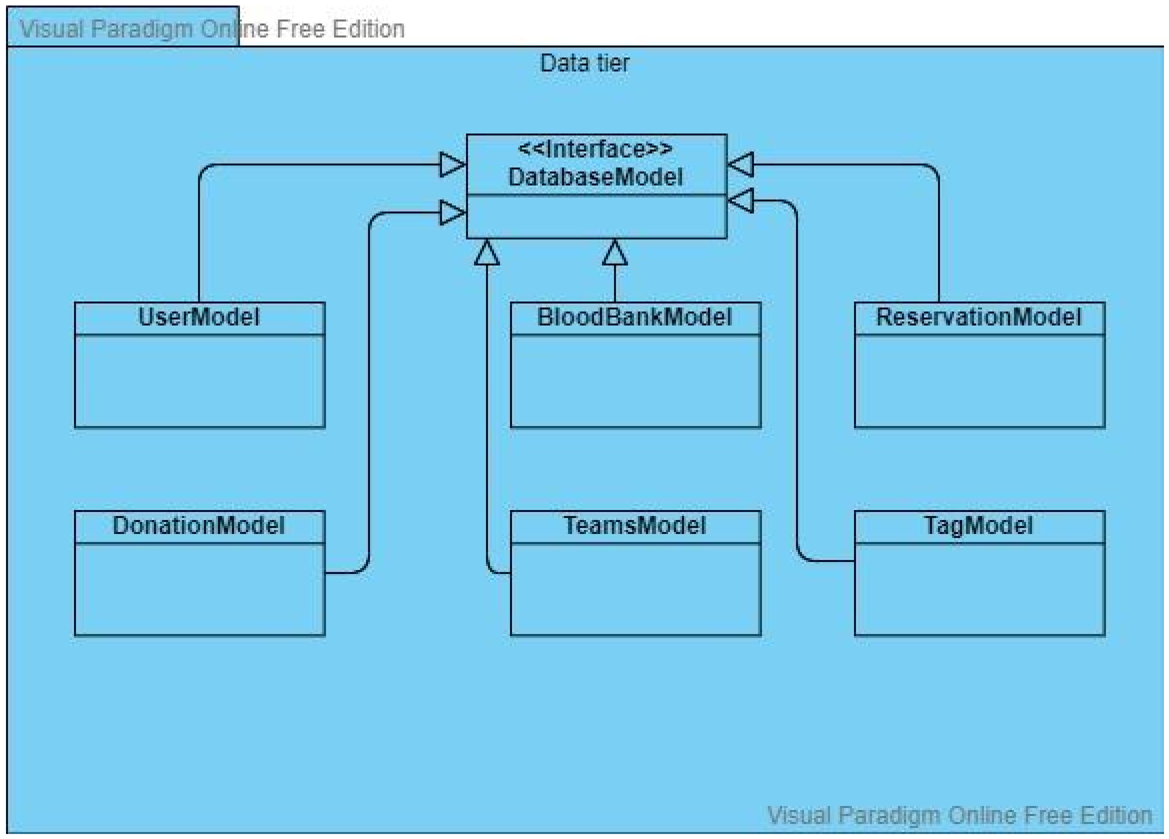


Logic tier is the subsystem for management elements in order to provide the business logic.

Manager<<interface>>	This interface is responsible for the dataflow with the DatabaseManager class.
UserManager ~ *Manager*	This class provides the business logic behind the user operations.

BloodBankManager ~ *Manager*	This class provides the business logic behind the blood bank operations.
ReservationManager ~ *Manager*	This class provides the business logic behind the reservation operations.
DonationManager ~ *Manager*	This class provides the business logic behind the donation operations.
TeamsManager ~ *Manager*	This class provides the business logic behind the teams' operations.
AuthenticationManager ~ *Manager*	This class provides the business logic behind authentications operations.
DatabaseManager	This class is in control of the SQL database.
Passport	This class is given by the third-party package PassportJS and is used by Express to construct middlewares and authentication protocols for the web service [6].

3.3.3 Data tier


Visual Paradigm Online Free Edition
Data tier
<<Interface>> DatabaseModel
UserModel
BloodBankModel
ReservationModel
DonationModel
TeamsModel
TagModel
Visual Paradigm Online Free Edition

Data tier is the subsystem for Data elements for them to communicate easily.

Class	Description
DatabaseModel<<Interface>>	This interface is responsible for the get and set methods.

UserModel ~ *DatabaseModel*	This class provides a model for the User system in the database.
BloodBankModel ~ *DatabaseModel*	This class provides a model for the BloodBank system in the database.
ReservationModel ~ *DatabaseModel*	This class provides a model for the reservation system in the database.
DonationModel ~ *DatabaseModel*	This class provides a model for the donation system in the database.
TeamsModel ~ *DatabaseModel*	This class provides a model for the teams system in the database.
TagModel ~ *DatabaseModel*	This class provides a model of the Tag entity in the database.

4. Consideration of Various factors in Engineering design

4.1 Public Health

Perhaps the most important factor to consider, we do not think that Helping-Hand will be constrained by public health. While it is true that our project aims to provide a solution to a health problem, in all aspects it is a supplementary application. It is more of a social media app aimed to connect donors and recipients, it holds absolutely no authority in any medical domains. When a donor and recipient is connected, any medical operations deemed necessary will be performed by hospitals or blood banks, so we do not foresee any negative health effects caused by our application and we do not foresee any negative effects caused on our project by such concerns.

Level of Effect: 2/10

4.2 Public Safety

Public safety is an important factor for our project, as we aim to have individuals meet up in real life. Upon confirmation from the user, their name, contact information and location will be shared with a complete stranger. While we will hold the two parties' information, we cannot verify if this information is completely correct. We will try to combat these problems in two ways: First, we will share a person's desired amount of information upon their request, and we will not share live location data, as in we will only share the target hospital/blood bank's location data. After these precautions, it will mostly fall on the user to ensure their safety. Since some sort of blood donation request systems already exist (e-mails/announcements), we do not think our project will introduce any new safety concerns that do not already exist.

Level of Effect: 5/10

4.3 Privacy

Privacy is another extremely important factor to consider, because Helping-Hand will hold vital personal information, including but not limited to: Name, Surname, Phone Number, E-Mail, Blood Type and such. While we are not set on how much will be stored on the mobile client, all information of all participants will be stored on our server. Therefore, it is important that this information is kept securely and is not shared with other people without the said individual's confirmation.

Level of Effect: 10/10

4.4 Public Welfare

HelpingHands will be completely free to download and use. Also, the application will not contain any paid services. All a person will need is a compatible mobile phone and an internet connection. Therefore, Helping-Hand is not constrained by public welfare.

Level of Effect: 0/10

4.5 Economic Constraints

As the entire development team consists of CS 491 students with their own equipment, we do not foresee any costs during development. For now, we are using GitHub Pages to host our project website, which is free. To distribute our application, we can use our website, which will eliminate any distribution costs, otherwise, if we use platforms such as Google Play Store or Apple App Store, we will have some costs. Finally, we plan to host our server in a cloud service, for which we are considering Amazon Web Services. While we cannot foresee the potential costs for using the service, our Innovation Expert explained to us that it is free for some amount of usage time or storage amount. To conclude, we do not foresee any costs in the context of CS 491/492, but we may come across potential costs if we decide to continue the project after that.

Level of Effect: 0/10

4.6 Social Constraints

Helping-Hand will not concern itself with gender, race or any other social aspects. The only consideration we will have to make is that we have to make sure that participants are over age 18, which we plan to verify during registration.

Level of Effect: 1/10

Factor	Level of Effect (out of 10)
Public Health	2
Public Safety	5
Privacy	10
Public Welfare	0
Economic Constraints	0
Social Constraints	1

Any other factors	0

5. Teamwork Details

5.1 Contributing and functioning effectively on the team

Because a project has a specific goal, it's essential that everyone involved understands what that goal is. A project member can make better individual judgments and decrease confusion and rework by understanding the target outcome. Furthermore, each individual is an important component of the overall project structure. Everyone should be aware of their own and others' responsibilities. Each member of the group should communicate with the others by asking questions and providing updates on the parts they are in charge of. Furthermore, each group member must adapt to the changes that may occur during the project's various phases. As a result, they can make the most of their contribution to the project. Apart from that, the project's tasks should be separated in such a way that everyone can work efficiently. This can be accomplished by splitting the duties into areas where project members have extensive understanding.

5.2 Helping creating a collaborative and inclusive environment

To create a collaborative and inclusive environment in the project, all team members should be aware of each other's strengths and shortcomings and respond accordingly. They should endeavor to compensate for their team members' weaknesses so that the project's flow is not disrupted. Each group member should be given the opportunity to be open-minded about their views and behaviors in order to approach each scenario from a different angle. This will have a favorable impact on the team's overall performance.

5.3 Taking lead role and sharing leadership on the team

To encourage everyone to take part in leadership, we divide the project into work packages and assign each person to lead one of them. Each team member is assigned to all work packages so that they all have the same amount of work. As a result, every project member will be informed of the project's processes. To execute duties in a timely manner, all team members will be in contact with one another at all times during the project.

6. References

1)*What is the history of blood transfusion?* Latest Medical News, Clinical Trials, Guidelines - Today on Medscape. (2020, December 6). Retrieved October 10, 2021,from https://www.medscape.com/answers/434176-183004/what-is-the-history-of-blood-transfusion#:~:text=The%20earliest%20known%20blood%20transfusions,Blundell%20in%20London%20in%201818.

2)U.S. Department of Health and Human Services. (n.d.). *Blood transfusion*. National Heart Lung and Blood Institute. Retrieved October 10, 2021, from https://www.nhlbi.nih.gov/health-topics/blood-transfusion.

3)*US Blood Supply Facts*. Facts About Blood Supply In The U.S. | Red Cross Blood Services. (n.d.). Retrieved October 10, 2021, from https://www.redcrossblood.org/donate-blood/how-to-donate/how-blood-donations-help/blood-needs-blood-supply.html.

[4] Axios. “Axios/Axios.” GitHub, 7 Mar. 2020, github.com/axios/axios

[5] Baeldung. “The DAO Pattern in Java.” Baeldung, 21 Mar. 2020, www.baeldung.com/java-dao-pattern.

[6] “Passport.js.” Passport.js, www.passportjs.org/