Introduction

- **Purpose of the project:** The Online Blood Bank project aims to create a platform for managing and tracking blood donations, connecting donors with recipients, creating & managing blood donation camps and providing real-time information on blood shortages and needs.

- **Objectives:** The primary objectives of the project are to increase the efficiency and accessibility of the blood donation process, improve the matching of donors and recipients, and provide real-time information on blood shortages and needs with a clean UI and experience.

- **Scope of the project:** The scope of the project includes the development of a web-based platform for managing and tracking blood donations, connecting donors with recipients, and providing real-time information on blood shortages and needs. The platform will include both a user-facing interface and an blood bank's interface for managing the data.

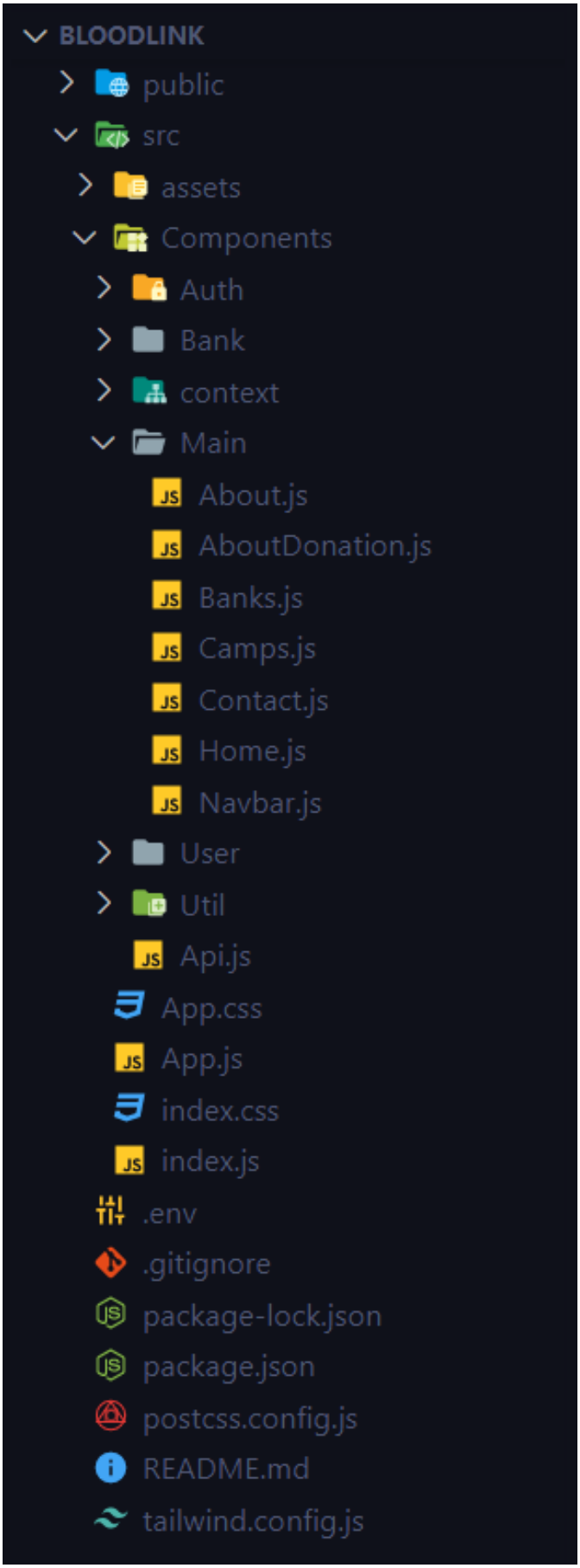
Project Description

- BloodLink will work as a platform for users to register as blood to either request/donate blood and blood banks to manage their stocks by managing the pending donations and request along with scheduling blood camps and managing them. The system will authenticate the user/bank using their username(mobile) and password to further perform other actions.

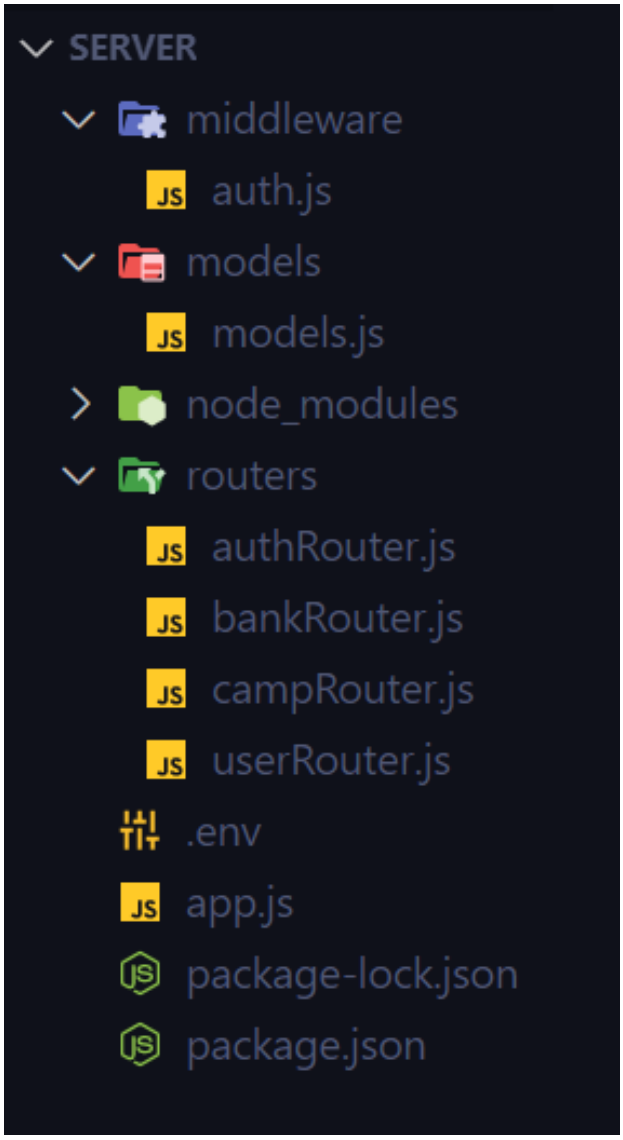
- The milestones for the project include completing the database design, completing the user interface design, completing the back-end functionality, and completing testing.

- **Project structure & Architecture:**

Frontend

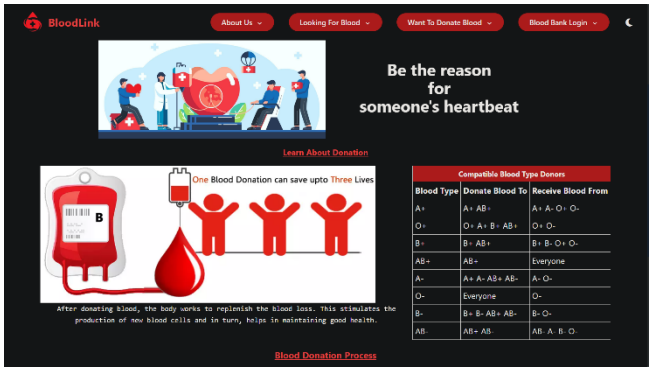


Backend



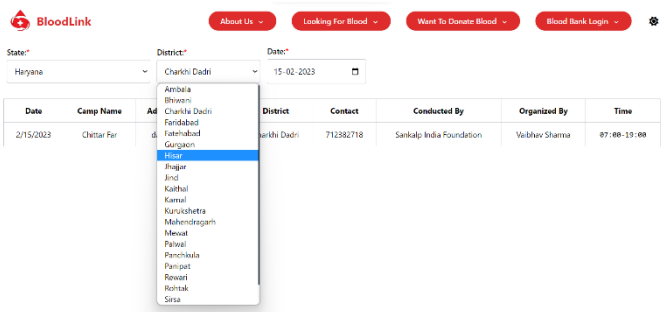
- **Home:**

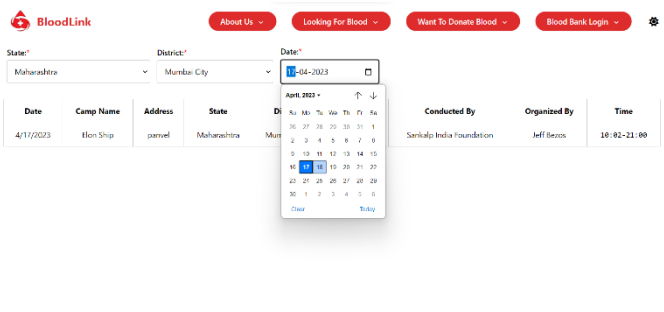
The home page of the website displays the navbar with position fixed at the top of the webpage with options available for the user to navigate across other pages with option available to switch between dark and light theme.

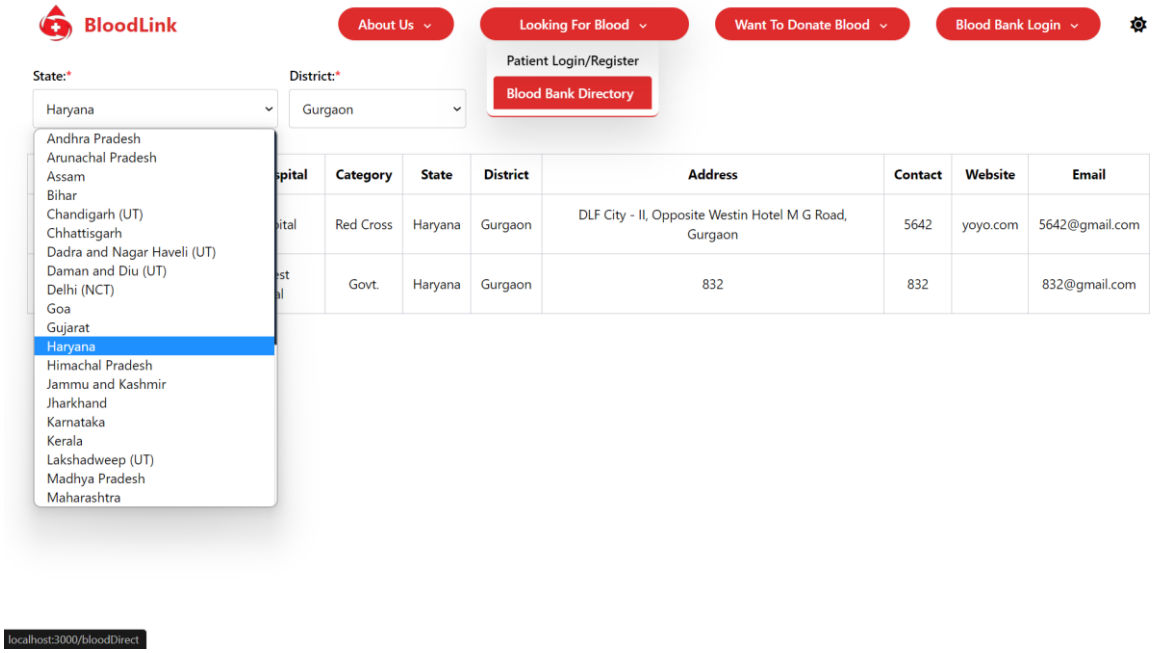


- **<u>About, Contact, About Blood Donation, Blood Bank Directory & Blood Donation Camps:</u>**

  These are the options in the submenu of the navbar for user to know about us, the system, the process of blood donation and to get access to the list of the blood donation camps scheduled by different blood banks that can be filtered by either state, district and the date scheduled. The Blood Bank Directory option lets the user to see the list of all the blood banks registered and the details on the system.

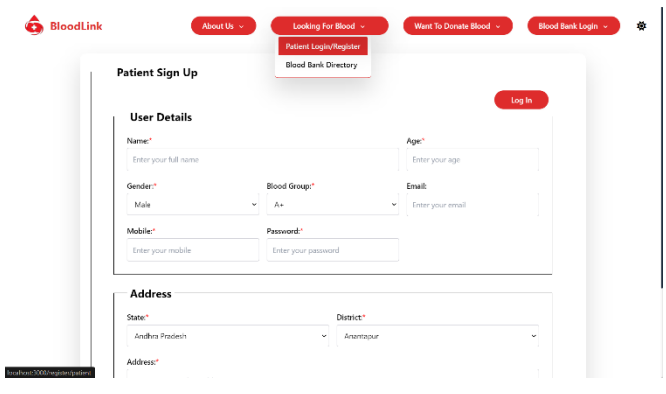


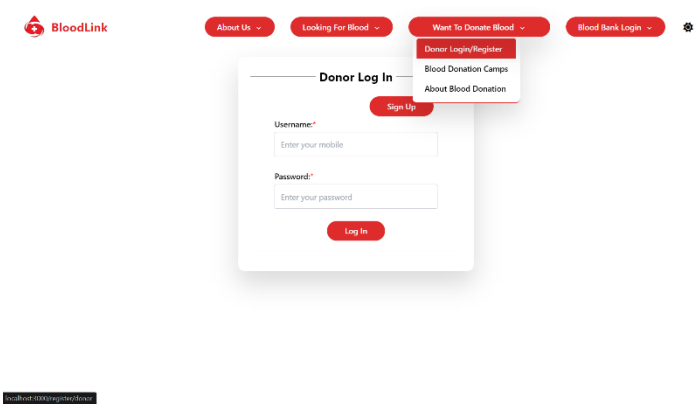




- **<u>Patient/Donor Registration & Login:</u>**

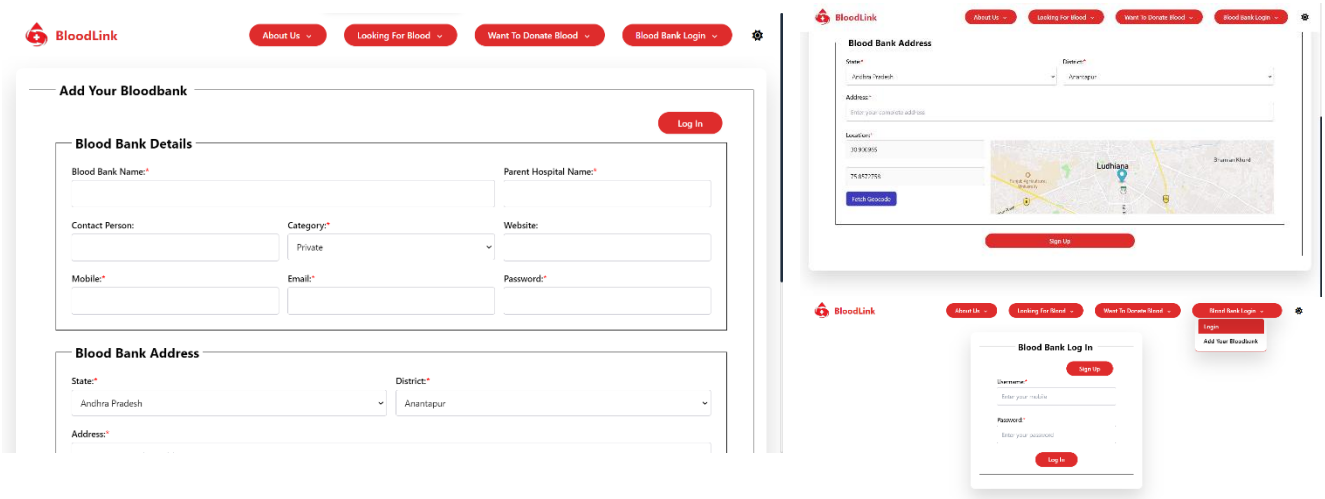
  Either donor or the one who is requesting for blood can register as a user using the Donor or Patient SignUp/Login page available by entering all the essential details like Name, Age, Gender, Blood Group, Email, Mobile, Password & Address Details.





- **Blood Bank Registration & Login:-**

  New Blood Banks can register on the portal using `Add your Bloodbank` option available under `Blood Bank Login` in Navbar and the already registered can login using `Login` available under the same.
  Blood Banks can register by entering the essential info including BloodBank Name, Parent Hospital Name, Contact Person, Category, Website, Mobile, Email, Password & Address details. Here, the address will include the state, district and complete address including the geographic location co-ordinates of the bank which can be fetched at that time using `Fetch Geocode` option available.



- **Authentication:**

- The authentication of the users during registration and logging in has been done using JWTs (Json Web Tokens), a JWT will be signed everytime any user/bank registers or logs into the website which will be stored as the cookie in the user's browser with a max expiration age scheduled after which user will need to log in again.
- This JWT token will be a random string of characters which will get verified at the backend and authenticate the user. To check the authenticity of the user logged in, a request will be sent to the backend using a api route which will return a Boolean value after verifying the JWT token of the user basis on which the logged in bank/user will be able to access all services.

- **Security:**

- For security purpose, the password saved at the server will be a encrypted hash value generated after the user's registration which will neither be known to database administrators nor the system managers and the backend. The

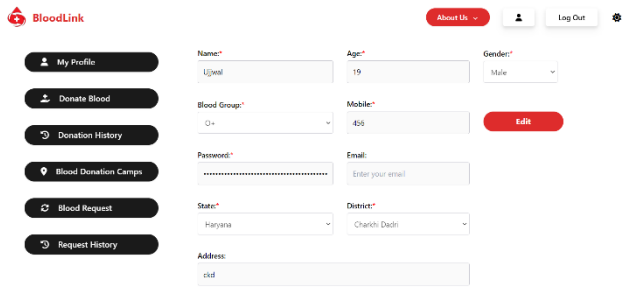
middle-ware will match the hash value of the entered password by user on logging in with the hash value stored in the database to authenticate the user.

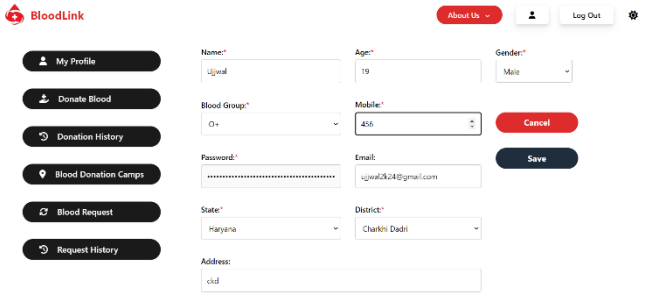
- Further, complete authentication has been done on the system with securing all routes which shouldn't be accessible to the user who is not logged in.

User Side

➢ Main:

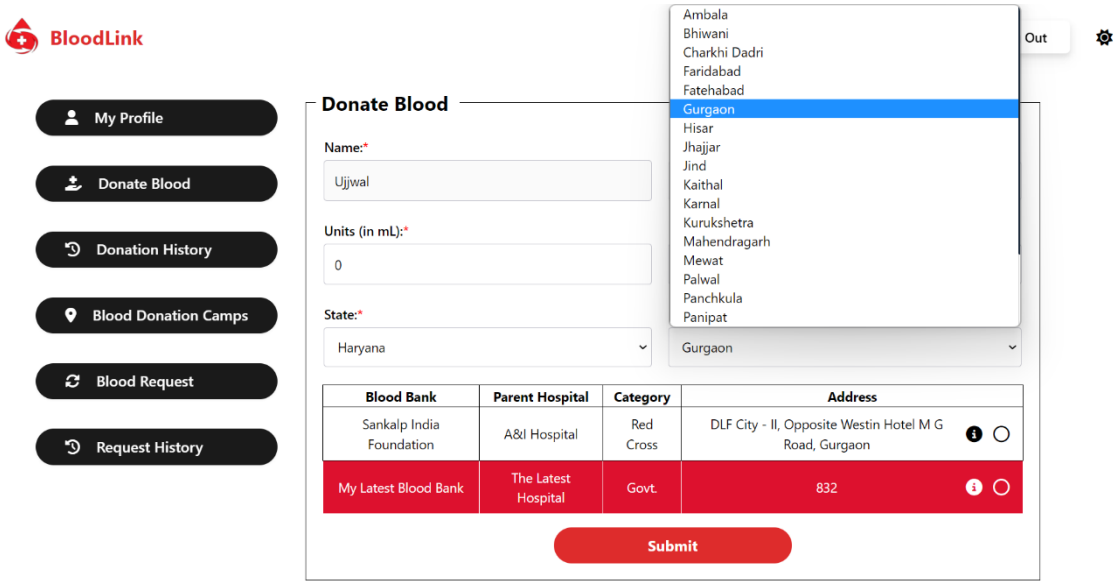
- The user logged in will have an access to different options for editing his/her profile, donate blood, request for blood, see donation/requests history and to register for a blood donation camp.
- The navbar will be changed for the logged-in user for navigating to the about, contact us, my profile page and to logout.



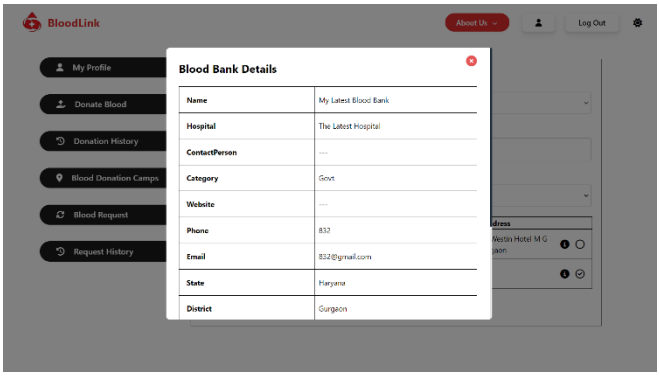


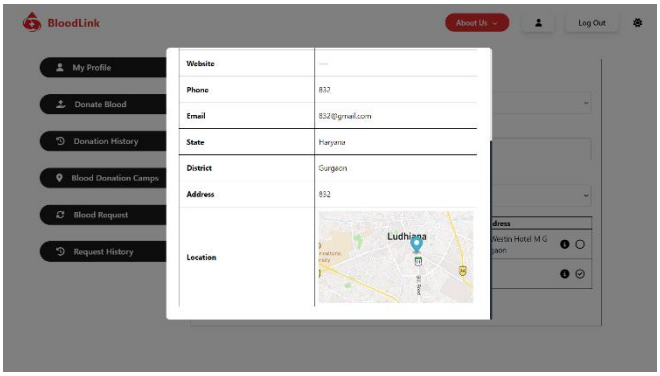
➢ Donate Blood:

➢ The logged-in user can donate blood by entering the units they want to donate and any disease if they have and can choose blood bank by state and district where they want to donate. Name & age will get auto-filled as per the details of the current logged-in user.

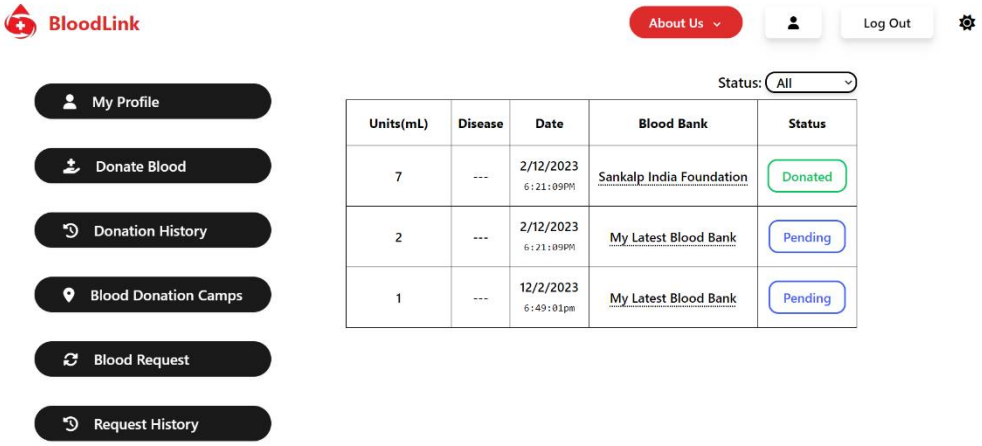


- There will be an extra option also available for the user to see the complete details of the blood bank.

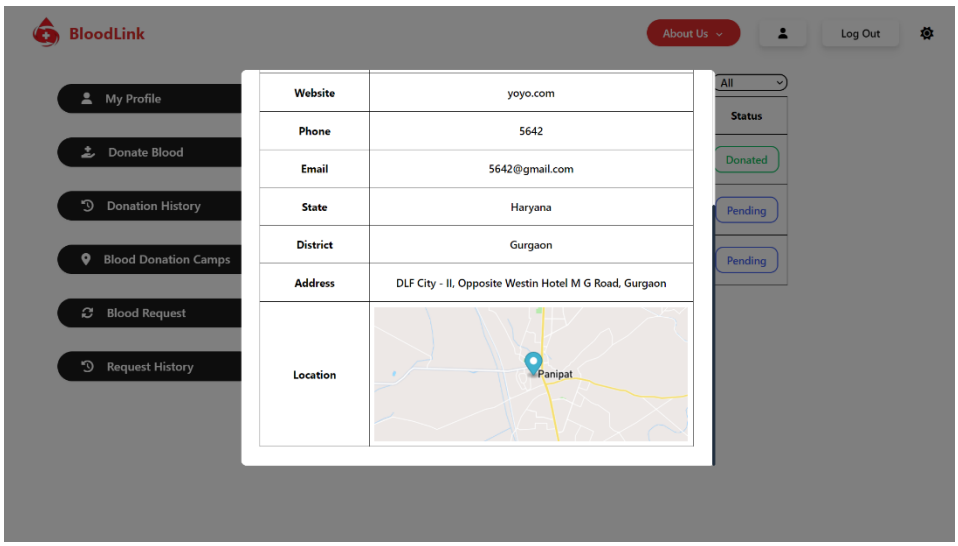




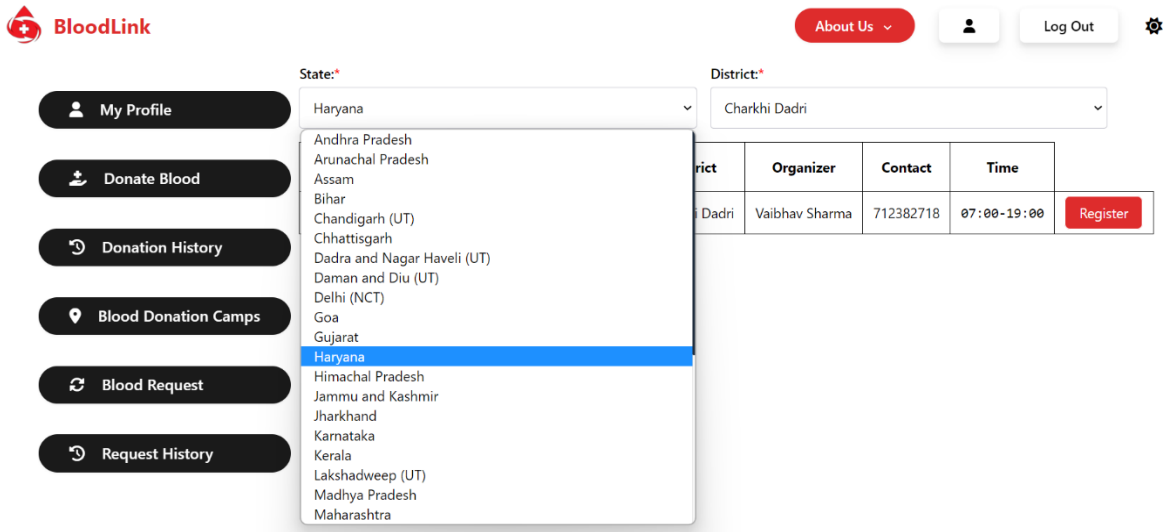
- **<u>Donation History:</u>**
- The user will have an access to see his past donation history and their status (Pending/Approved/Donated/Denied)



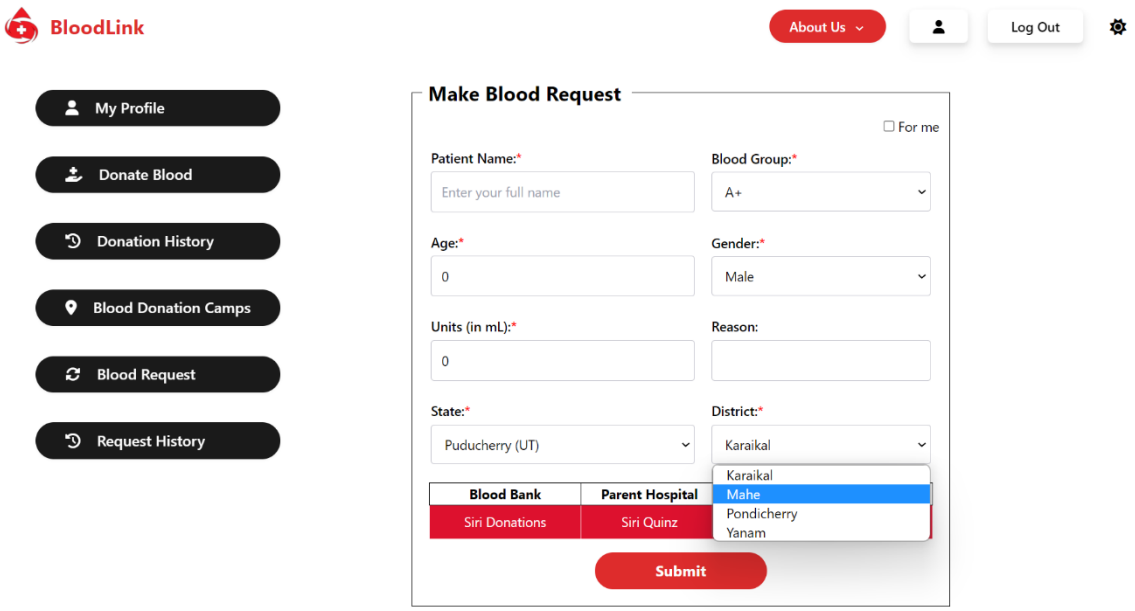
- The user can click on the bank name to see the complete bank details also.



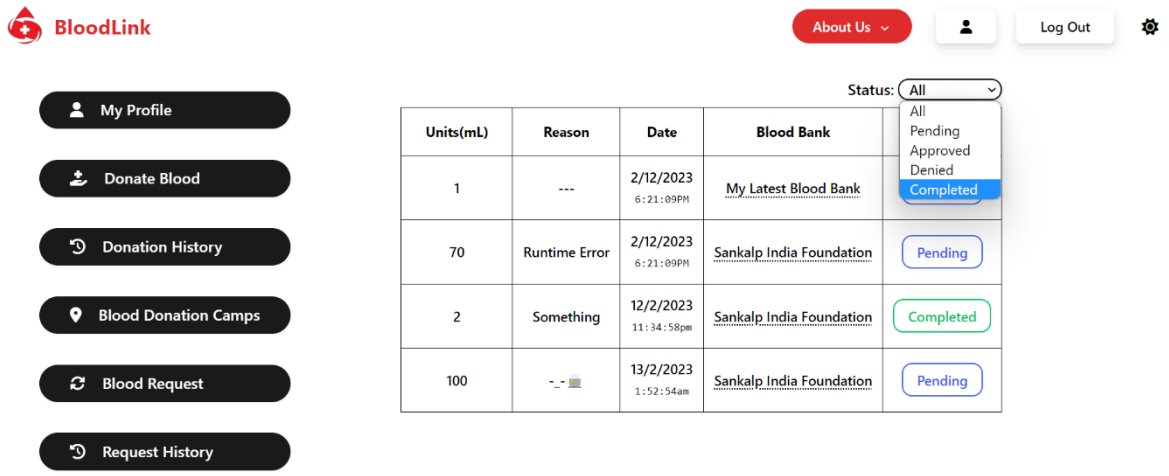
- **Blood Donation Camps:**
- User can register for the blood donation camps scheduled by different blood banks and can also filter them by the state & district.



- **Blood Request:**
- Users can request blood either for himself or any other person, if he will be requesting blood for himself, the name, age, blood group & gender values will get auto-filled accordingly and the users will need to enter the units required, reason and selecting the bank by filtering them all by state and districts.
- The user can see the complete bank details using the `i` button again same as on the donate blood panel.

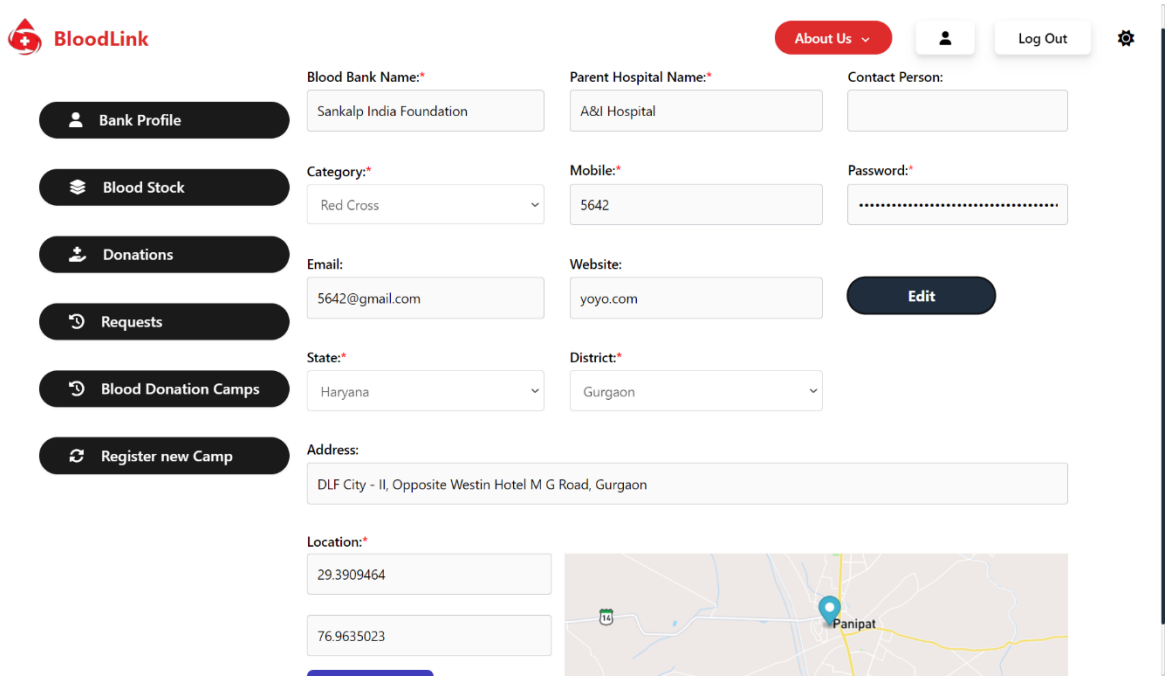


- **Request History:**
- The user will have an access to see his past requests history and their status (Pending/Approved/Completed /Denied)
- The option to see the complete details of the bank by click on bank will be available here also.



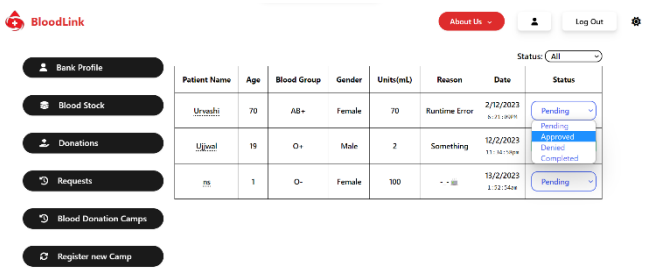
Blood Bank Side

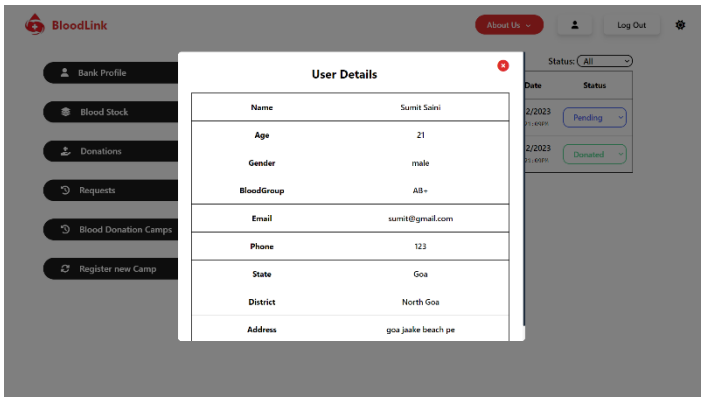
- **Main:**
- The blood-bank logged in will have an access to different options for its profile, see blood stock, donations, requests, blood donation camps and can register a new blood donation camp.
- The navbar will be changed for the logged-in bank for navigating to the about, contact us, my profile page and to logout.



- **Blood Stock:**
- `Blood stock` will show the current available blood stock at the current logged-int blood bank.

- **Donations/Requests:**
- These options will let the logged-in blood bank to see and manage the blood donation requests and the requests for blood by updating their status accordingly by the user details.
- The complete details of the user can be seen here by clicking on the user's name underlined.

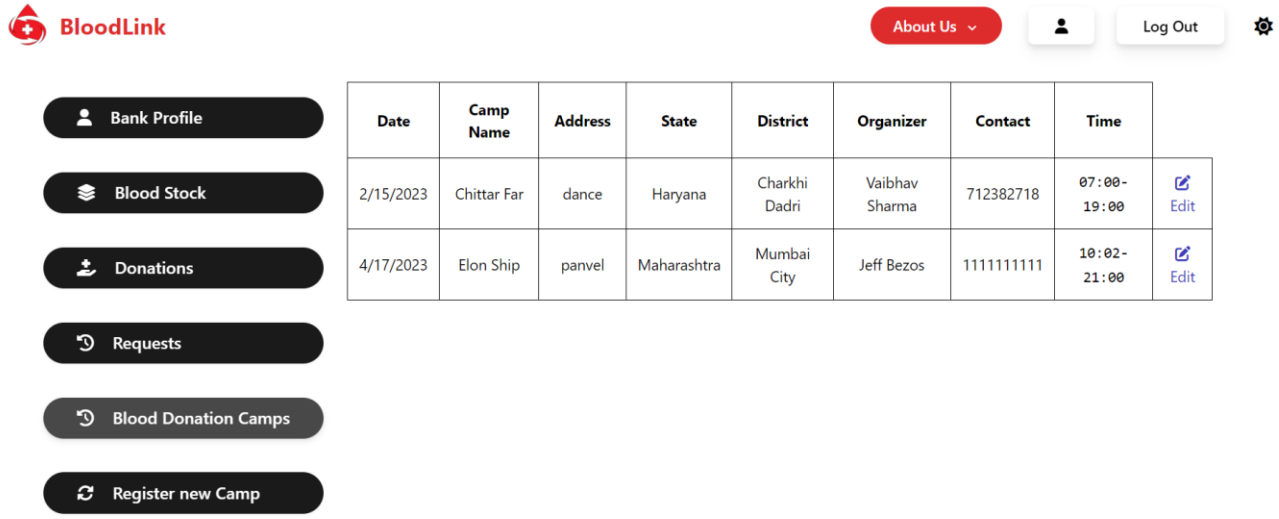


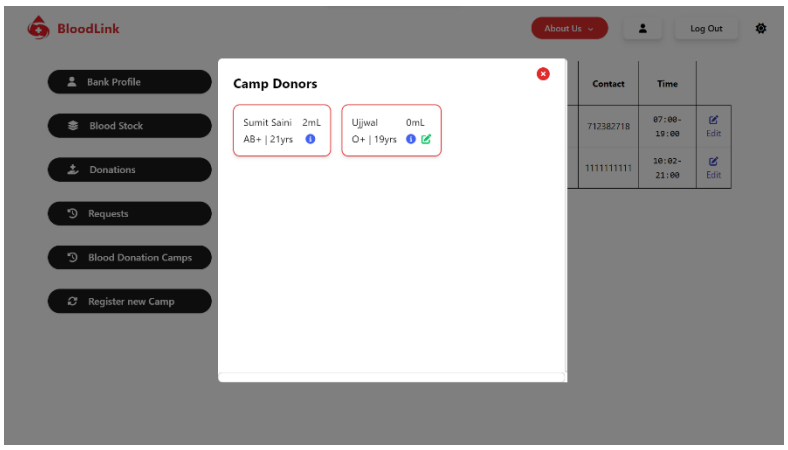


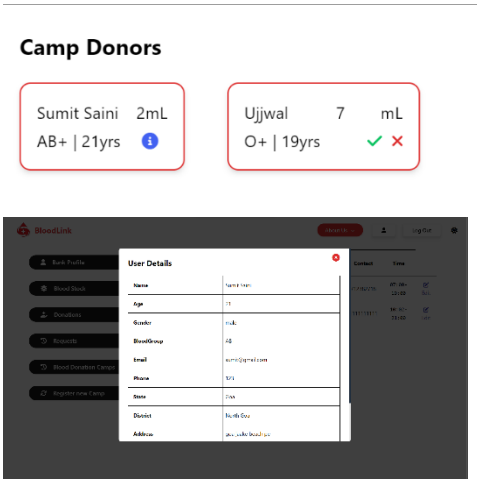
- The option to update the status of the request will get automatically disabled after the status gets updated to "Donated" for donations and "Completed" for the requests.

- **Blood Donation Camps:**
- Bank currently logged-in can see all of his registered blood donation camps and can edit using the "Edit" button provided.
- On clicking the "Edit" button, a new window will be popped-up showing all the camp donors registered for the camp. Further clicking on the "i" button mentioned under every donor will open another pop-window and will show the complete details of the donor.

- Bank can here update the donation value after the donation completion at the blood camp by the donor, any new user at the camp can register for the same by signing up directly from the options mentioned previously.

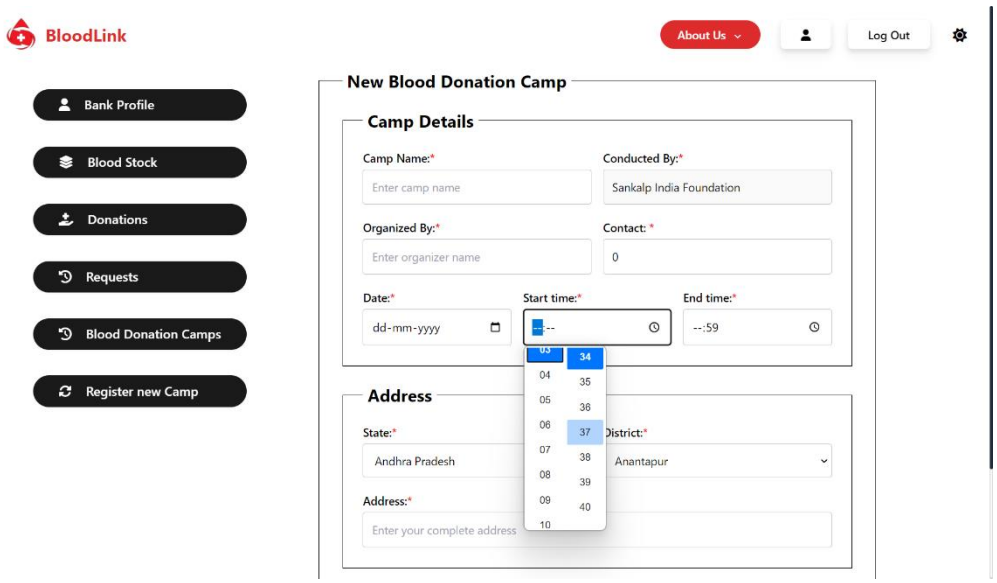






- **<u>Register new camp:</u>**

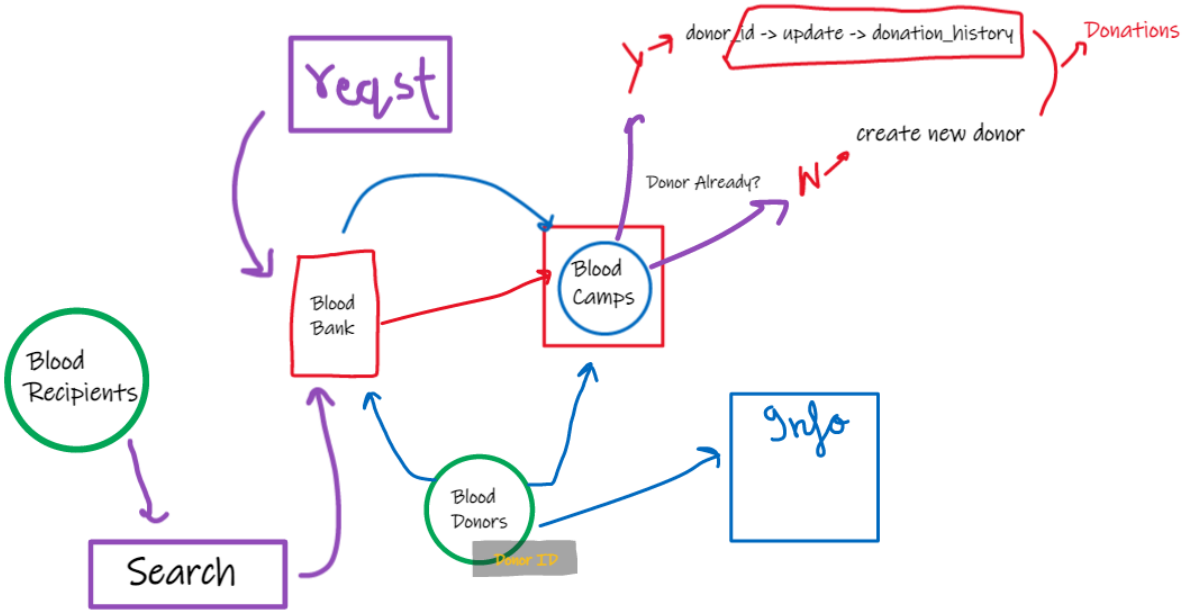
  Bank can create/register a new blood donation camp by `Register new Camp` option available by specifying all details with the date & time duration of the camp.



Technical Specification

- **Front-end:** React.js
- **Back-end:** Node.js, Express.js
- **Database:** MongoDB

Database Design & Schema



The database used for the BloodLink project is MongoDB, a NoSQL database.

The following collections are being used to store and organize the data for the project:

Users, BloodBanks, Donations, Requests, Camps

➢ Users:

```
// ------- User Model -------

// Create schema for Users
const userSchema = new mongoose.Schema({
    name: { type: String, required: true },
    age: { type: Number, required: true },
    gender: { type: String, required: true },
    bloodGroup: { type: String, enum: bloodGroups, required: true },
    email: { type: String },
    phone: { type: Number, unique: true, required: true },
    password: { type: String, required: true },
    state: { type: String, required: true },
    district: { type: String, required: true },
    address: { type: String },
});

// Create model for Users
const User = mongoose.model('Users', userSchema);
```

➢ BloodBanks:

```
// Create schema for Blood Banks
const bloodBankSchema = new mongoose.Schema({
    name: { type: String, required: true },
    hospital: { type: String, required: true },
    contactPerson: { type: String },
    category: { type: String, required: true },
    website: { type: String },
    phone: { type: Number, required: true },
    email: { type: String, required: true },
    password: { type: String, required: true },
    state: { type: String, required: true },
    district: { type: String, required: true },
    address: { type: String, required: true },
    latitude: { type: Number, required: true },
    longitude: { type: Number, required: true },
    requests: [{
        requestId: { type: mongoose.Schema.Types.ObjectId, ref: 'Requests' },
    }],
    donations: [{
        donationId: { type: mongoose.Schema.Types.ObjectId, ref: 'Donations' },
    }],
    stock: {
        'A+': { type: Number, default: 0 },
        'A-': { type: Number, default: 0 },
        'B+': { type: Number, default: 0 },
        'B-': { type: Number, default: 0 },
        'AB+': { type: Number, default: 0 },
        'AB-': { type: Number, default: 0 },
        'O+': { type: Number, default: 0 },
        'O-': { type: Number, default: 0 }
    }
});

// Create model for Blood Banks
const BloodBank = mongoose.model('BloodBanks', bloodBankSchema);
```

➢ Donations:

```
// ------- Donations Model -------

// Create schema for Donations
const bloodDonations = new mongoose.Schema({
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'Users', required: true },
    bankId: { type: mongoose.Schema.Types.ObjectId, ref: 'BloodBanks', required: true },
    units: { type: Number, required: true },
    date: { type: String, required: true },
    disease: { type: String },
    status: { type: String, required: true,
              enum: ['Pending', 'Approved', 'Denied', 'Donated'],
              default: 'Pending'
            },
});

// Create model for Donors
const Donations = mongoose.model('Donations', bloodDonations);
```

➢ Requests:

```
// ------- Requests Model -------

// Create schema for Patients
const bloodRequests = new mongoose.Schema({
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'Users', required: true },
    bankId: { type: mongoose.Schema.Types.ObjectId, ref: 'BloodBanks', required: true },
    name: { type: String, required: true },
    age: { type: Number, required: true },
    gender: { type: String, required: true },
    bloodGroup: { type: String, enum: bloodGroups, required: true },
    units: { type: Number, required: true },
    date: { type: String, required: true },
    reason: { type: String },
    status: { type: String,
              enum: ['Pending', 'Approved', 'Denied', 'Completed'],
              default: 'Pending'
            }
});

// Create model for Patients
const Requests = mongoose.model('Requests', bloodRequests);
```

- **Camps:**

```
// Create schema for Camps
const campSchema = new mongoose.Schema({
    name: { type: String, required: true },
    date: { type: Date, required: true },
    address: { type: String, required: true },
    state: { type: String, required: true },
    district: { type: String, required: true },
    bankId: { type: mongoose.Schema.Types.ObjectId, ref: 'BloodBanks' },
    organizer: { type: String, required: true },
    contact: { type: Number, required: true },
    startTime: { type: String, required: true },
    endTime: { type: String, required: true },
    donors: [{
        _id: { type: mongoose.Schema.Types.ObjectId, ref: 'Users', unique: true },
        units: { type: Number, required: true, default: 0 },
        status: { type: Number, enum: [0, 1], default: 0 }
    }]
});

// Create model for Camps
const Camp = mongoose.model('Camps', campSchema);
```

- **Description:**

- The `Users` collection stores complete details of any registered user.
- `BloodBanks` collection stores complete details of the registered blood banks,
    - The `requests` field here is an array of objects having requestID referring to the `Requests` collection.
    - The `donations ` field here is an array of objects having donationID referring to the `Donations` collection.
    - The `stock` field is an object with all the blood groups as keys and value of the units of the stock of each blood group available.
- `Donations` collections stores every donation request made by the user, with userId referring to the `Users` collection and bankId referring to the `BloodBanks` collection with the status of the request,units to be donated.

- `Requests` collections stores every request made for blood by the user, with userId referring to the `Users` collection and bankId referring to the `BloodBanks` collection with the status of the request, units requested,.
- `Camps` schema stores the details of any camp scheduled by any blood bank where bankId is referring to `BloodBanks` collection and donors field is an array of objects with _id referring the `Users` collection, the units donated and the status.
- All the fields referring to other collections will get populated using the populate() in mongoose accordingly to get the complete data.

------- Thank You