Online Blood Bank

Objective:

Online Blood Bank is an online application that brings dignity to Life of people by making Quality blood and blood samples available when needed.

Users of the System:

1. Admin
2. User

Functional Requirements:

- Build an application that User can access and get blood donar details.
- The application should have signup, login, profile, and blood bank page.
- This application should have a provision to maintain a database for user information, blood information.
- Also, an integrated platform required for admin and customer.
- Administration module to include user management and blood bank Management.
- **Automatically remove the blood sample older than 90 days.**

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

- ➢ Filters for donar like near by location.
- ➢ Multi-factor authentication for the sign-in process

Output/ Post Condition:

- ➢ Records Persisted in Success & Failure Collections
- ➢ Standalone application / Deployed in an app Container

Non-Functional Requirements:

Security	• App Platform –UserName/Password-Based Credentials • Sensitive data has to be categorized and stored in a secure manner • Secure connection for transmission of any data
Performance	• Peak Load Performance • Online Blood Bank -< 3 Sec • Admin application < 2 Sec • Non Peak Load Performance • Online Blood Bank < 2 Sec • Admin Application < 2 Sec
Availability	• 99.99 % Availability
Standard Features	• Scalability • Maintainability • Usability • Availability • Failover
Logging &	• The system should support logging(app/web/DB) & auditing at

Auditing	all levels
Monitoring	• Should be able to monitor via as-is enterprise monitoring tools
Cloud	• The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
Browser Compatible	• IE 7+ • Mozilla Firefox Latest – 15 • Google Chrome Latest – 20 • Mobile Ready

Technology Stack

Front End	Angular 7+ Google Material Design Bootstrap / Bulma
Server Side	Spring Boot Spring Web (Rest Controller) Spring Security Spring AOP Spring Hibernate
Core Platform	OpenJDK 11
Database	MySQL or H2

Platform Pre-requisites (Do’s and Don’ts):

1. The angular app should run in port 8081. Do not run the angular app in the port: 4200.
2. Spring boot app should run in port 8080.

Key points to remember:

1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.
2. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.
3. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.
4. Adhere strictly to the endpoints given below.

Application assumptions:

1. The login page should be the first page rendered when the application loads.

2. Manual routing should be restricted by using AuthGaurd by implementing the canActivate interface. For example, if the user enters as http://localhost:4200/signup or http://localhost:4200/home the page should not navigate to the corresponding page instead it should redirect to the login page.
3. Unless logged into the system, the user cannot navigate to any other pages.
4. Logging out must again redirect to the login page.
5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:

1. Basic email validation should be performed.
2. Basic mobile validation should be performed.

Project Tasks:

API Endpoints:

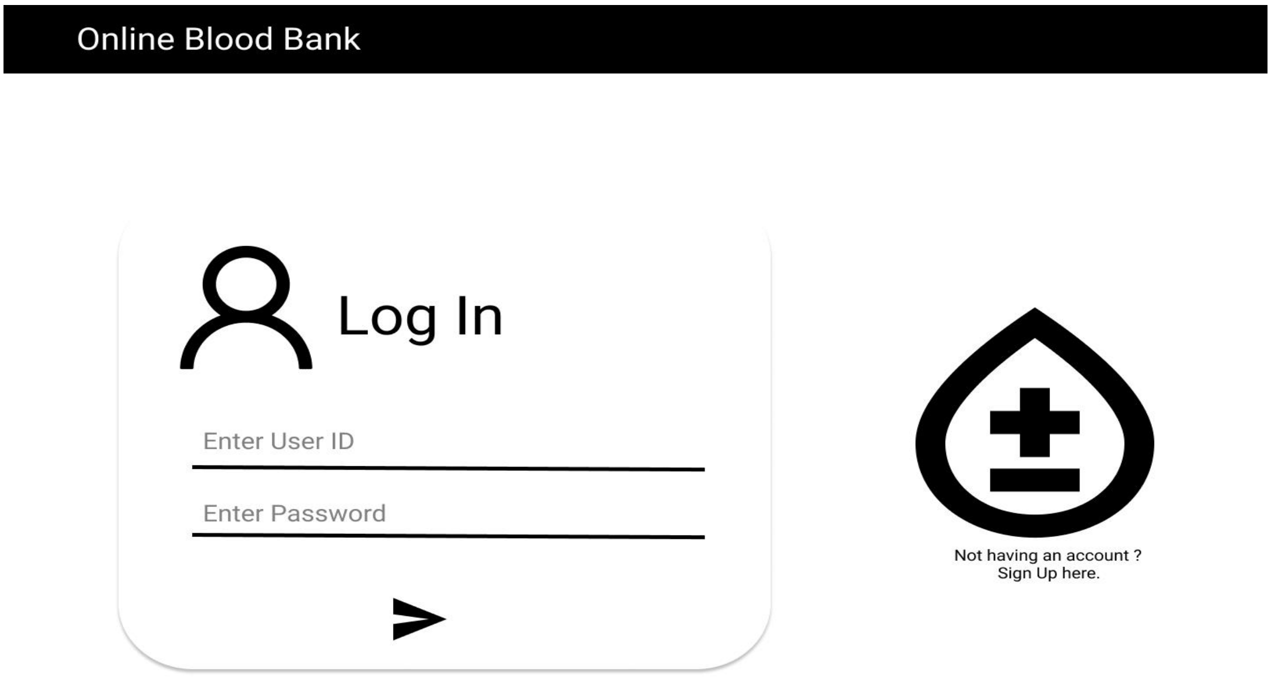
USER			
Action	URL	Method	Response
Login	/login	POST	true/false
Signup	/signup	POST	true/false
Get All Sample	/sample	GET	Array of samples
Get All Donor	/donor	GET	Array of Donors
Get All donor by group	/donor/{group}	GET	List of Donor Details by blood group
Get All Sample by group	/sample/{group}	GET	List of sample Details by blood group
Get Donor	/donor/{id}	GET	Particular Donor
Get Sample Details	/sample/{id}	GET	Particular Sample
ADMIN			
Action	URL	Method	Response
Get All donor	/donor	GET	Array of donor
Get All sample	/sample	GET	Array of sample
Add Sample	/admin/addSample	POST	Sample added
Delete Sample	/admin/sample/{id}	DELETE	Sample deleted
Update Sample	/admin/sample/{id}	PUT	Sample Updated
Update Donor	/admin/donor/{id}	PUT	Donor Updated
Delete Donor	/admin/donor/{id}	DELETE	Donor Removed

Frontend:

User:

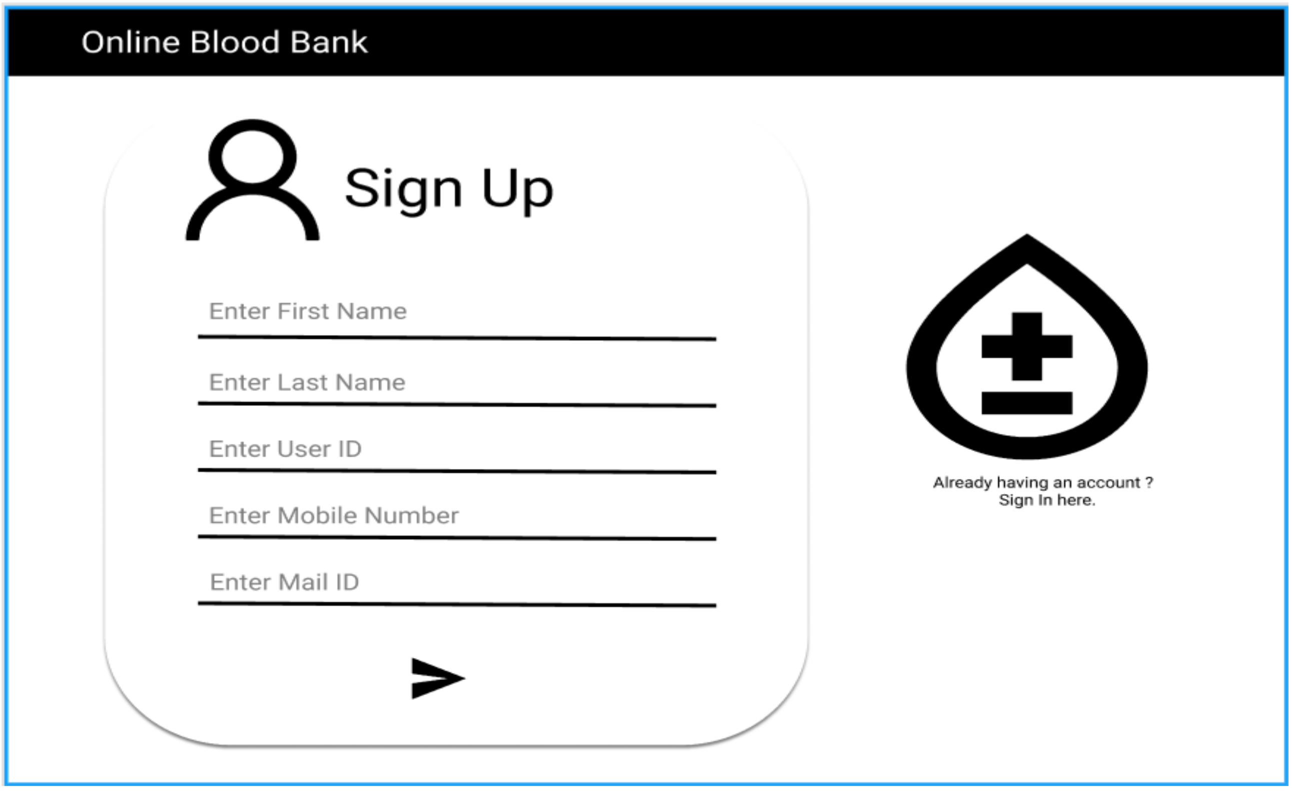
Login:

Output Screenshot:



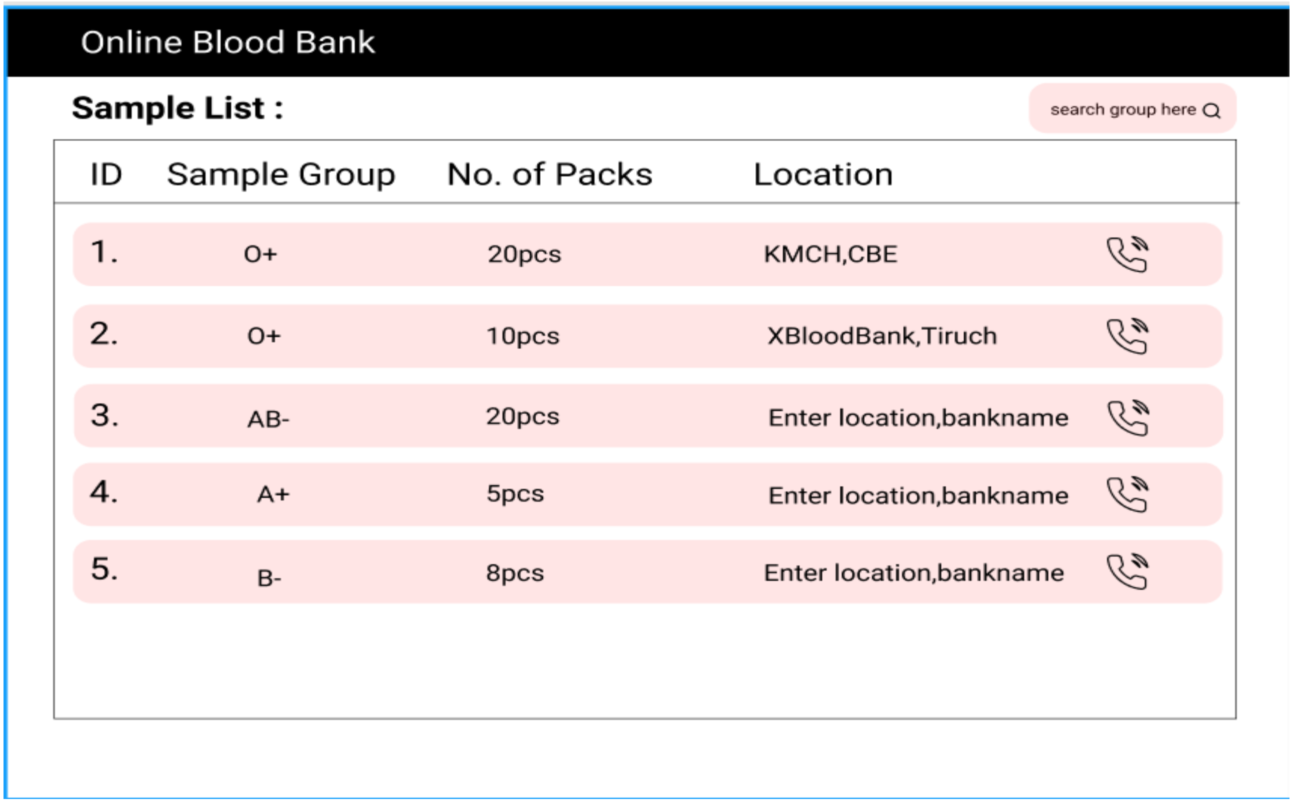
Signup:

Output Screenshot:



Blood Sample:

Output Screenshot:



Online Blood Bank

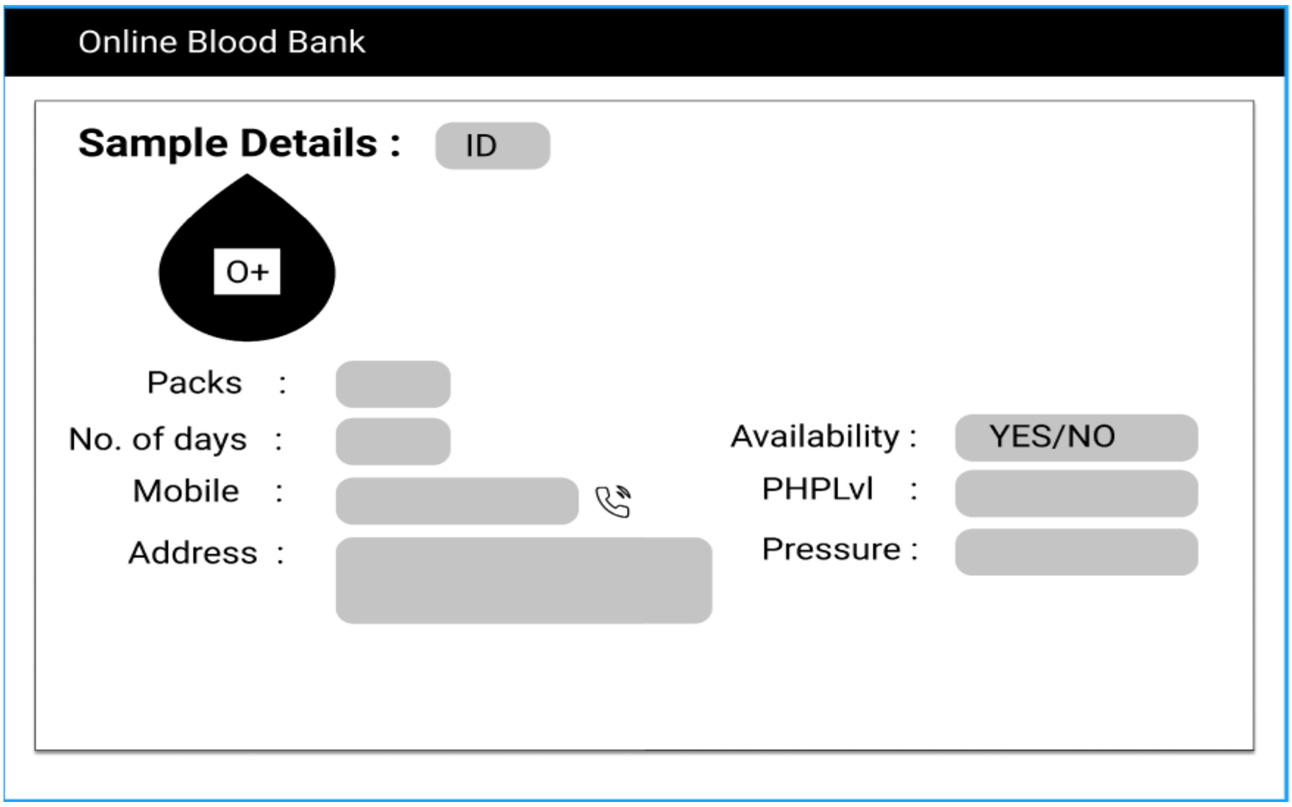
Sample List :

search group here

ID	Sample Group	No. of Packs	Location
1.	O+	20pcs	KMCH,CBE
2.	O+	10pcs	XBloodBank,Tiruch
3.	AB-	20pcs	Enter location,bankname
4.	A+	5pcs	Enter location,bankname
5.	B-	8pcs	Enter location,bankname

Blood Sample ID:

Output Screenshot:



Online Blood Bank

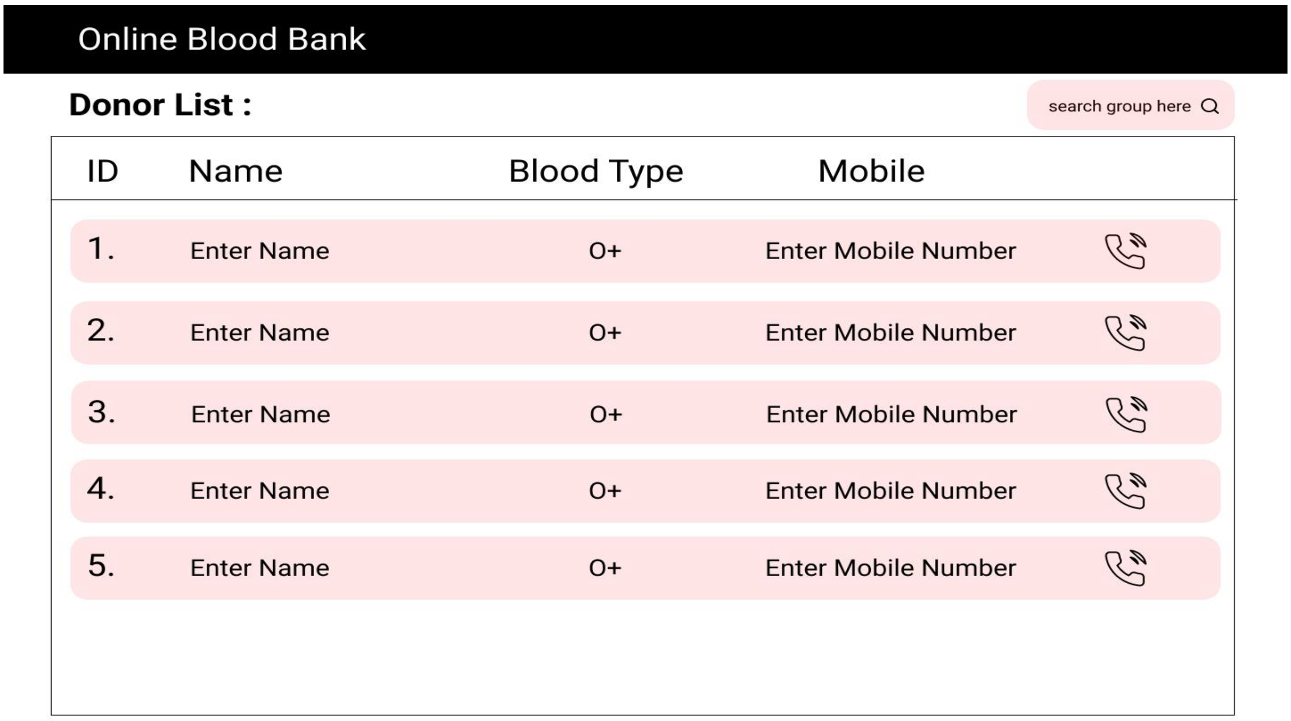
Sample Details : ID

O+

Packs :			
No. of days :		Availability :	YES/NO
Mobile :		PHPLvl :	
Address :		Pressure :	

Donors:

Output Screenshot:



Online Blood Bank

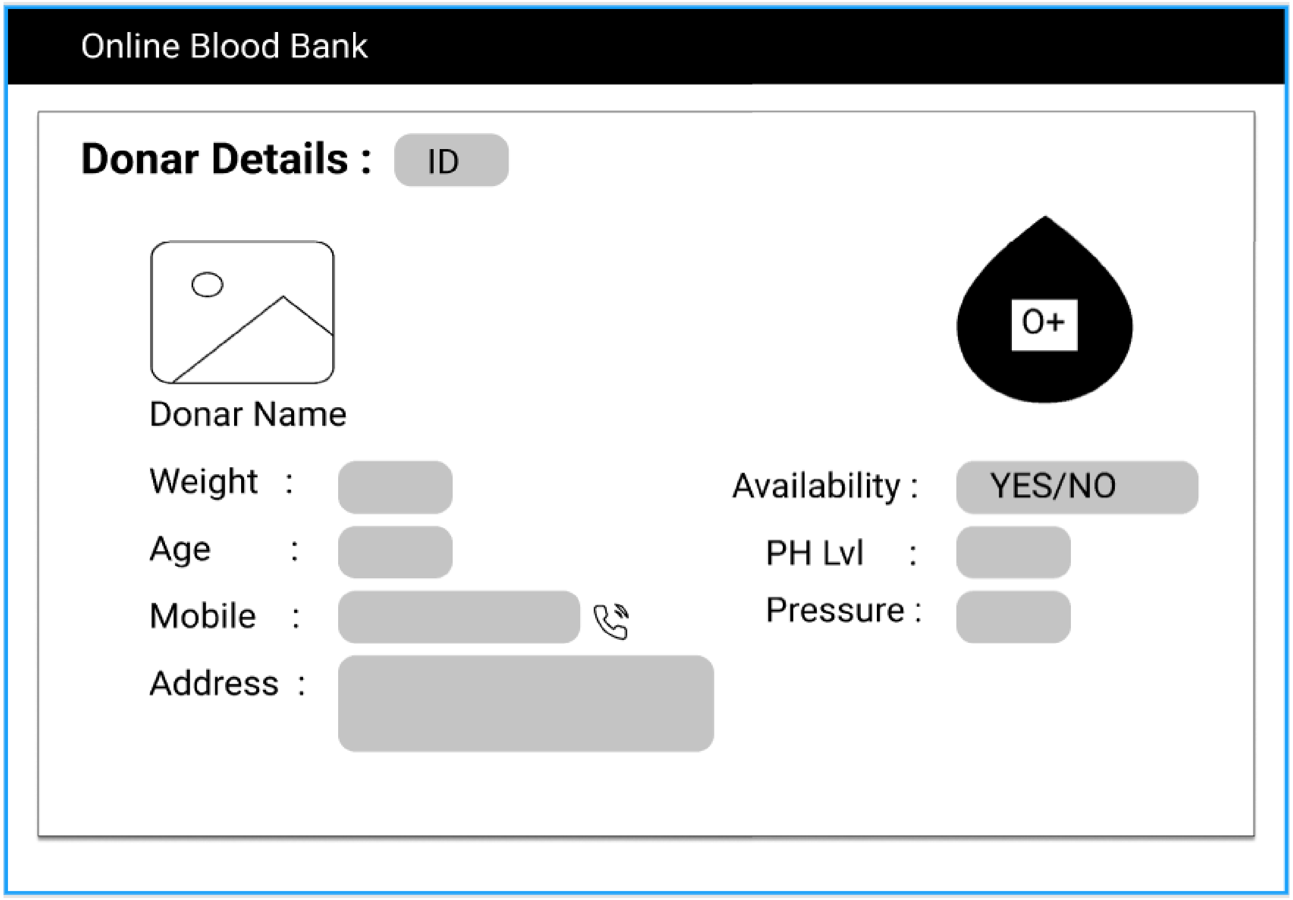
Donor List :

search group here

ID	Name	Blood Type	Mobile
1.	Enter Name	O+	Enter Mobile Number
2.	Enter Name	O+	Enter Mobile Number
3.	Enter Name	O+	Enter Mobile Number
4.	Enter Name	O+	Enter Mobile Number
5.	Enter Name	O+	Enter Mobile Number

Donor By Id:

Output Screenshot:



Online Blood Bank

Donar Details : ID

O+

Donar Name

Weight :

Age :

Mobile :

Address :

Availability : YES/NO

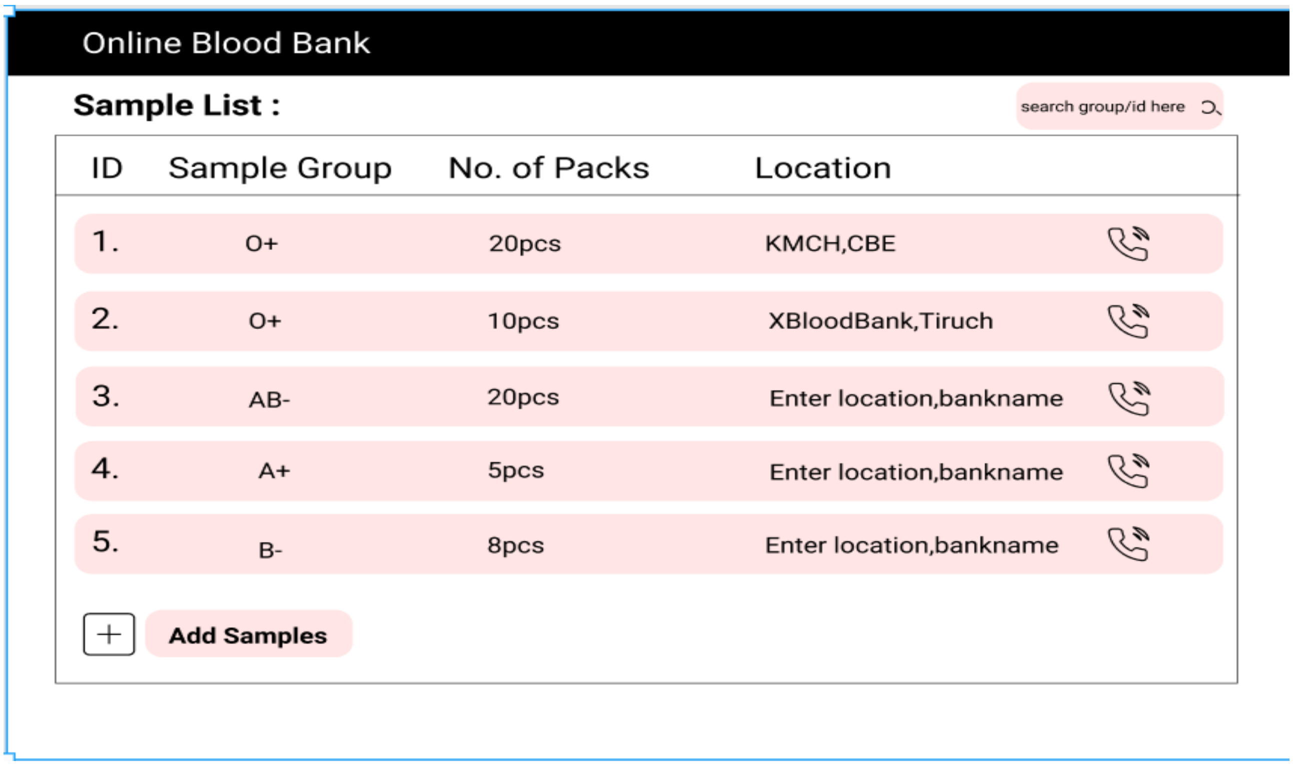
PH Lvl :

Pressure :

Admin:

All Sample:

Output Screenshot:



Online Blood Bank

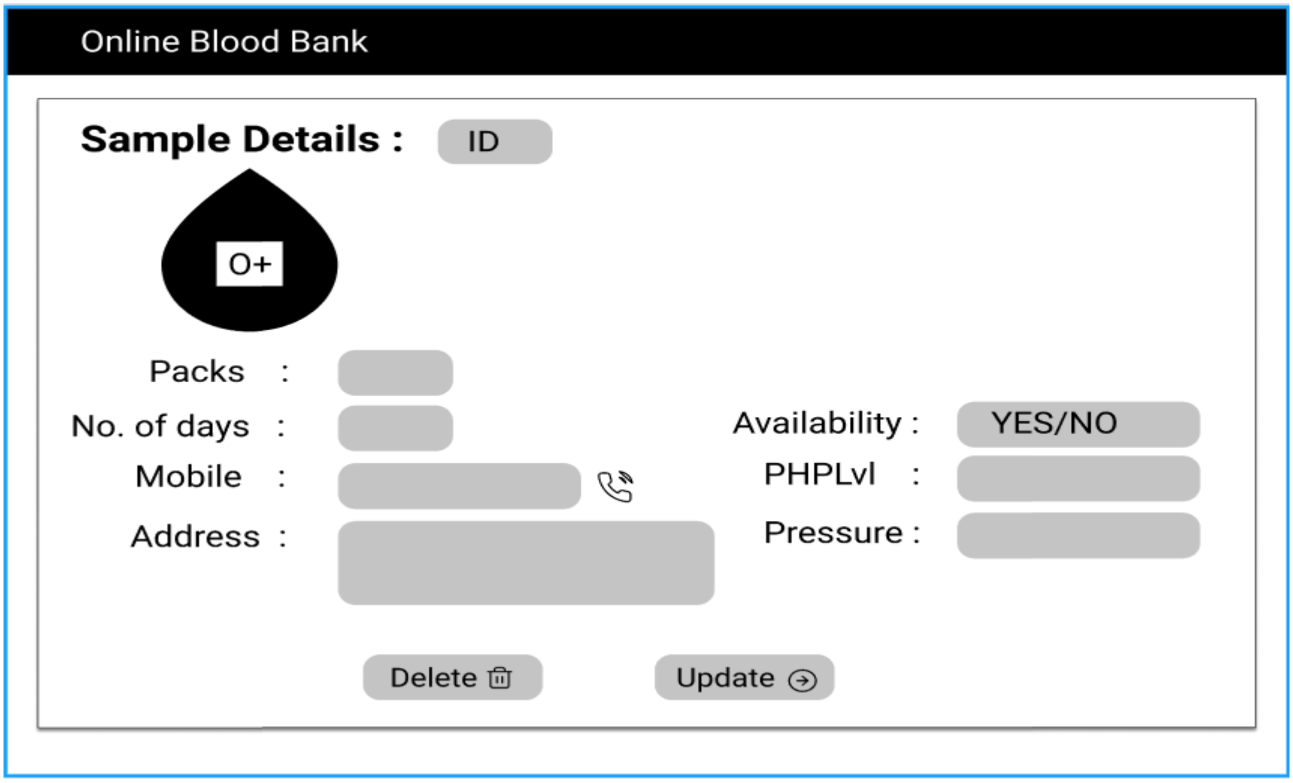
Sample List :

search group/id here

ID	Sample Group	No. of Packs	Location
1.	O+	20pcs	KMCH,CBE
2.	O+	10pcs	XBloodBank,Tiruch
3.	AB-	20pcs	Enter location,bankname
4.	A+	5pcs	Enter location,bankname
5.	B-	8pcs	Enter location,bankname

Add Sample:

Output Screenshot:



Online Blood Bank

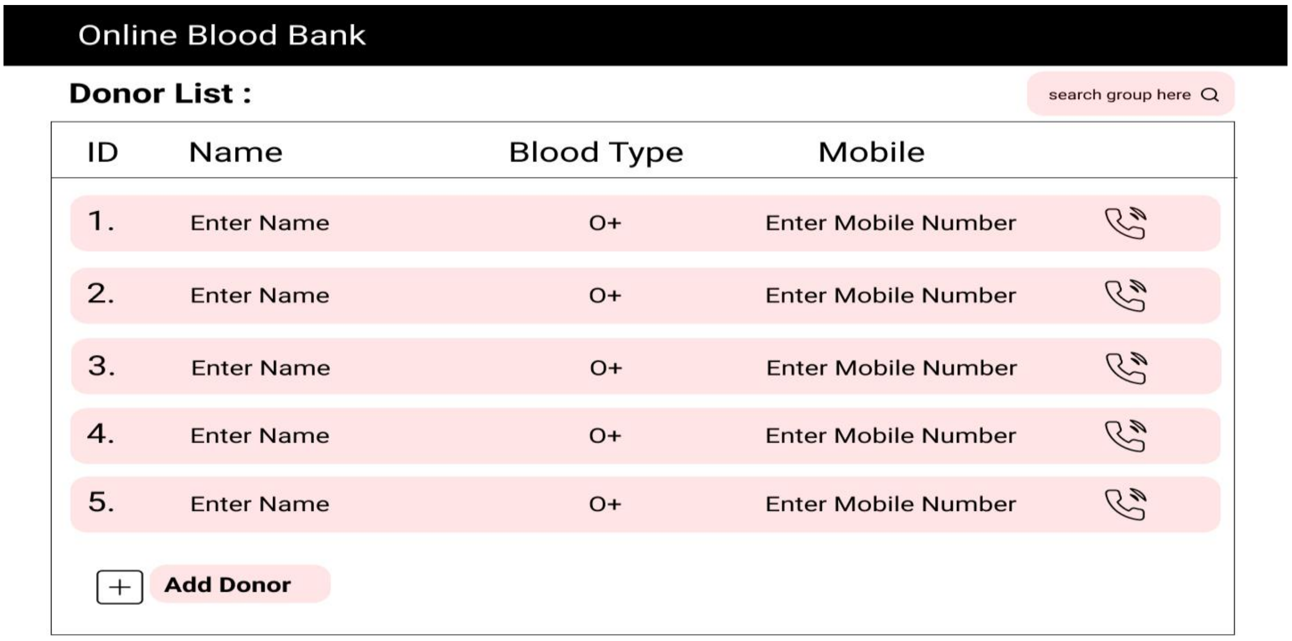
Sample Details : ID

O+

Packs :			
No. of days :		Availability :	YES/NO
Mobile :		PHPLvl :	
Address :		Pressure :	

All Donors:

Output Screenshot:



Online Blood Bank

Donor List :

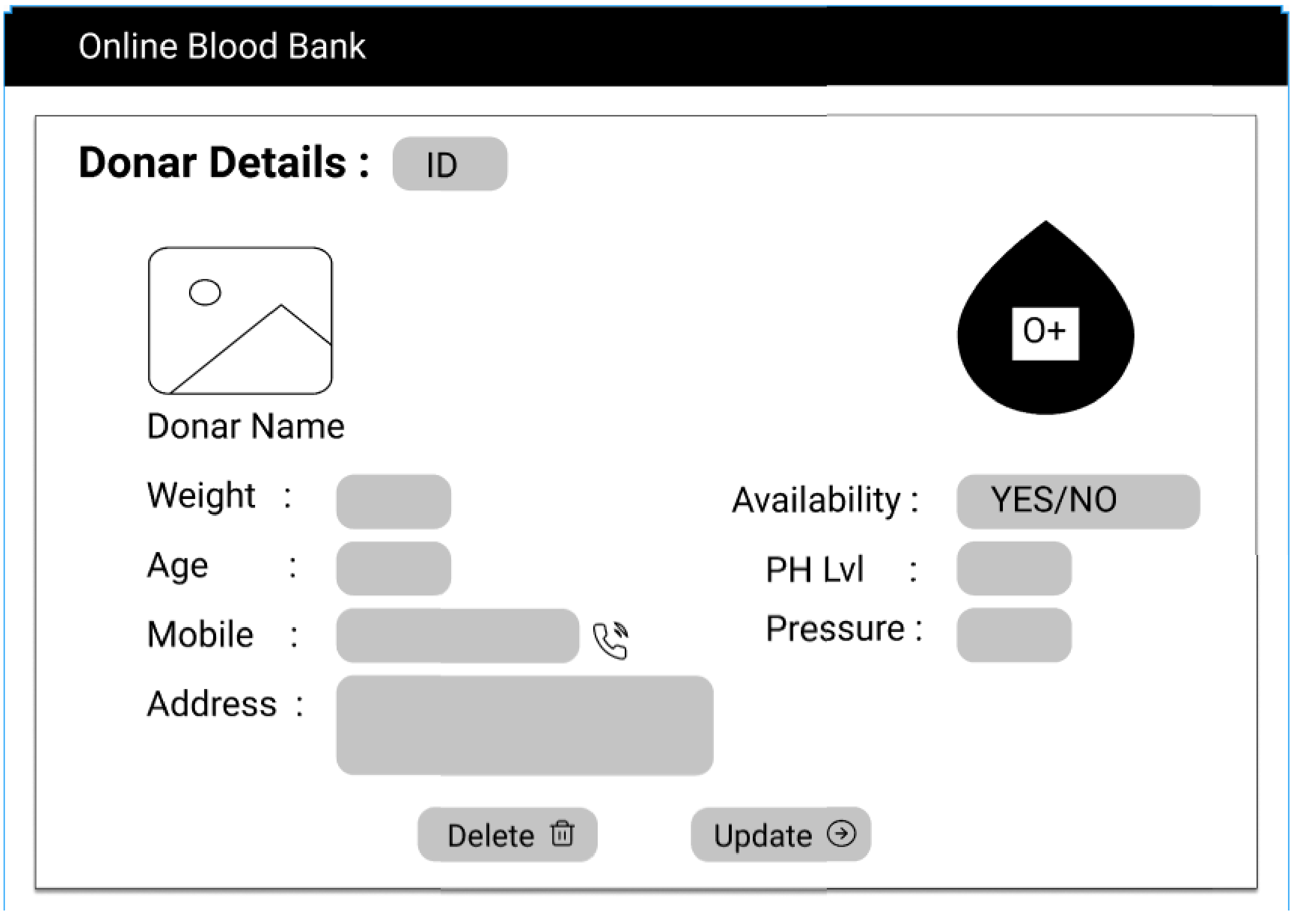
search group here

ID	Name	Blood Type	Mobile
1.	Enter Name	O+	Enter Mobile Number
2.	Enter Name	O+	Enter Mobile Number
3.	Enter Name	O+	Enter Mobile Number
4.	Enter Name	O+	Enter Mobile Number
5.	Enter Name	O+	Enter Mobile Number

Add Donor

Update Donor:

Output Screenshot:



Online Blood Bank

Donar Details : ID

Donar Name

O+

Weight :		Availability :	YES/NO
Age :		PH Lvl :	
Mobile :		Pressure :	
Address :			

Delete

Update

Backend:

Class and Method description:

Model Layer:

1. UserModel: This class stores the user type (admin or the User) and all user information.
    a. Attributes:
        i. email: String
        ii. password: String
        iii. mobileNumber: String
        iv. active: Boolean
        v. role: String
    b. Methods: -
2. LoginModel: This class contains the email and password of the user.
    a. Attributes:
        i. email: String
        ii. password: String
    b. Methods: -
3. BloodDonarModel: This class stores the details of the Blood Donor.
    a. Attributes:
        i. Id: String
        ii. donarName: UserModel
        iii. bloodGroup: String
        iv. PHLevel: String
        v. bloodPressure: String
        vi. active: Boolean
    b. Methods: -
4. BloodBankModel: This class stores the Blood Bank details.
    a. Attributes:
        i. bloodBankID: String
        ii. bloodGroup: String
        iii. bloodPressure: String

    iv. PHLevel: String
    v. Quantity: int
  b. Methods: -

<u>Controller Layer:</u>

5. SignupController: This class control the user signup
   a. Attributes: -
   b. Methods:
      i. saveUser(UserModel user): This method helps to store users in the database and return true or false based on the database transaction.
6. LoginController: This class controls the user login.
   a. Attributes: -
   b. Methods:
      i. checkUser(LoginModel data): This method helps the user to sign up for the application and must return true or false
7. DonorController: This class controls the add/edit/update/view products.
   a. Attributes: -
   b. Methods:
      i. List<DonorModel> getDonor(): This method helps the users to fetch all Donors from the database.
      ii. DonorModel getDonorByID(String id): This method helps to retrieve a donor from the database based on the user id.
      iii. DonorModel getDonorByBloodGroup(String group): This method helps to retrieve a donor from the database based on the blood group.
      iv. updateDonar(DonorModel data): This method helps to update a donor and save it to the database.
      v. addDonar(DonorModel data): This method helps to add a new donor to the database.
      vi. removeDonor (String id): This method helps to delete a donor from the database.
8. BloodBankController: This class helps in adding blood sample to the database, deleting the blood sample from the database, updating sample in the database.
   a. Attributes: -
   b. Methods:
      i. addBlood(String id): This method helps the admin to add the blood sample to the database.
      ii. List<BloodBankModel> showBloodSample(): This method helps to view all blood sample.

iii. removeBloodSample (String id): This method helps to delete a blood sample by Id.

iv. updateBloodSample (BloodBank data): This method helps to update a blood sample details.