

CSE471: System Analysis and Design

Project Report

Project Title: DigiCare

Table of Contents

Introduction.....	1
Functional Requirements	1
User Management:	1
Health Data and Goal Tracking:	2
Health Calculators and Blood Donation:	2
Blood Donation and Admin:	3
User Manual.....	4
Opening an Account / Sign Up / Registration:	4
Log in.....	7
Forgetting Password / Reset password	8
Profile Setup.....	11
Edit Profile	12
Log out.....	13
Set Goals:	15
See All Goal Status:	17
Health Trends / Statical Overview	18
Health Data Tracking / Add Health Data.....	20
See All health data	22

Medications / Add Medication.....	23
See all medications.	24
Calculators / BMI Calculators	25
Calculator / BMR Calculator	27
Calculator / Body Fat Calculator	28
Need Blood / Blood Donation / Blood Bank	30
Notifications.....	33
Ban / Unban from Admin Pannel.....	36
Backend Development	37
Register	38
Login, Logout	38
Setup Profile.....	39
Forget Password, Reset Password.....	40
Goal Model	41
Med Model.....	42
HealthData Model	43
There are also database models of User.....	43
Goal.....	44
Med	44

HealthData	45
DonationRequest.....	45
Blood Donor Routes	46
Notifications.....	46
Statistics	47
Calculators	48
BMI	48
BMR.....	49
Body Fat Index.....	49
Fitness Evaluation	50
Admin Panel.....	51
Frontend Development.....	52
Database Schema	60
Technology (Framework, Languages)	61
Conclusion	62

Introduction

Digicare is a dynamic health management platform designed to empower users in tracking their health progress and receiving personalized assistance. Developed on Flask backend complemented by HTML, CSS and JavaScript frontend, this platform offers an array of features catering to various aspects of health management and assistance.

Functional Requirements

User Management:

- **Home Page:** The home page serves as a central hub, providing users with an overview of Digicare's functionalities.
- **Sign Up:** This feature enables users to register for a Digicare account, granting access to the platform's suite of features.
- **Log In and Log Out:** Users can securely access their accounts through the login feature and log out when the session is complete.
- **Password Recovery:** In the event of forgotten passwords, users can recover access via a secure process involving security question authentication.
- **Profile Setup:** Users can set up their profile data on the platform.
- **Editing profile:** Users can customize their profiles, facilitating personalized health tracking within the platform.

Health Data and Goal Tracking:

- **Set Health Data, Goals, and Medications:** This feature empowers users to input and manage personal health metrics, goals, and medication details.
- **View Health Records:** Users gain access to a comprehensive overview of their health data, goals, and medications, facilitating easy tracking.
- **Goal and Medication Completion:** Users can mark goals or medications as completed, with real-time updates reflecting their progress.
- **Progress Comparison:** Digicare provides comparative progress charts, allowing users to compare their individual achievements with aggregate user data.

Health Calculators and Blood Donation:

- **Health Calculators:** Interactive BMI, BMR, and Body Fat calculators provide users with insightful health metrics and feedback.
- **Blood Donor Network:** Users can connect with potential blood donors within Digicare, fostering a supportive community for blood donation.
- **Request Blood Donation:** This feature allows users to request blood donation, providing necessary contact information for potential donors.
- **Blood Request Notifications:** Real-time notifications keep users informed about incoming blood donation requests and acceptances.

Blood Donation and Admin:

- **Blood Request Management:** A notification panel empowers blood request receivers to efficiently manage incoming requests.
- **Accept/Reject Blood Requests:** Users can make informed decisions on blood donation requests, accepting or rejecting based on their availability.
- **Admin Panel:** An exclusive access point for administrators to oversee user management and platform activities.
- **User Management:** Administrators have the capability to view details of all users registered on Digicare.
- **Ban/Unban:** The admin can apply bans or lift them as necessary, ensuring a mechanism to address unusual or unethical user activities.

User Manual

Opening an Account / Sign Up / Registration:

In the system, after setting it up, users will be directed to the Home page. Here, they need to either open an account or register by clicking the Sign-up button, leading them to the registration page. On this page, users must fill in the form with accurate details. It's crucial to remember the email, password, and security question answers for future logins. While the Digicare system provides a password reset option, remembering the security question answers is necessary for successful access. So, to make it simplify for a successful registration, these are the steps to follow:

1. Upon system setup, navigate to the Home page.
2. Click on the Sign-up button to create an account.
3. Fill in the required details on the registration page.
4. Take note of the email, password, and security question answers for future logins.
5. With the registered information, you can access the Digicare system. If needed, there's an option to reset the password, but remembering security question answers is essential for

this process.

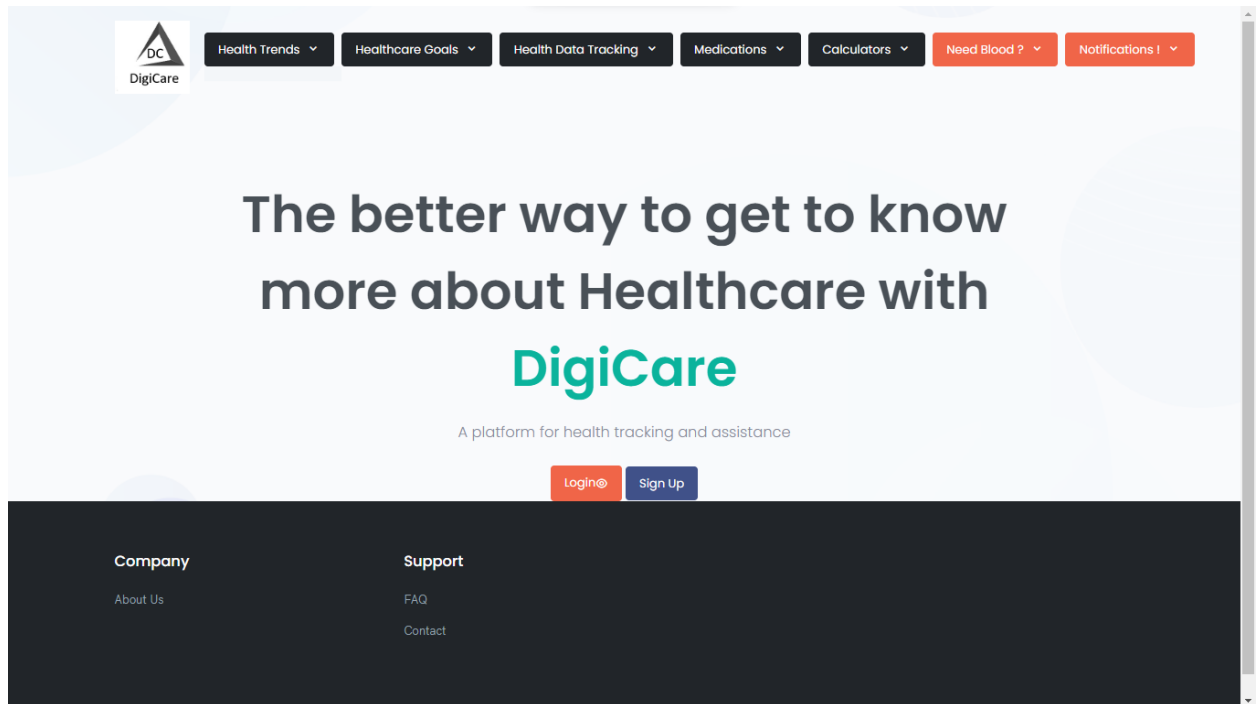


Figure 1 The Homepage of DigiCare

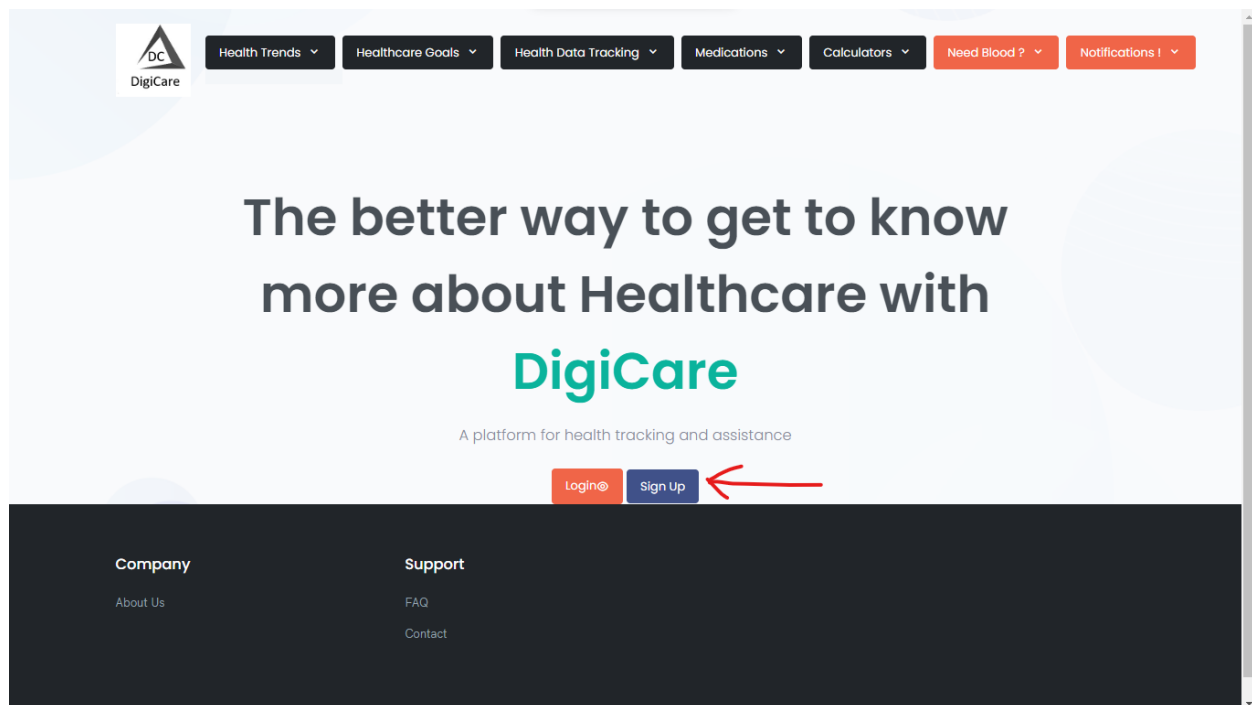
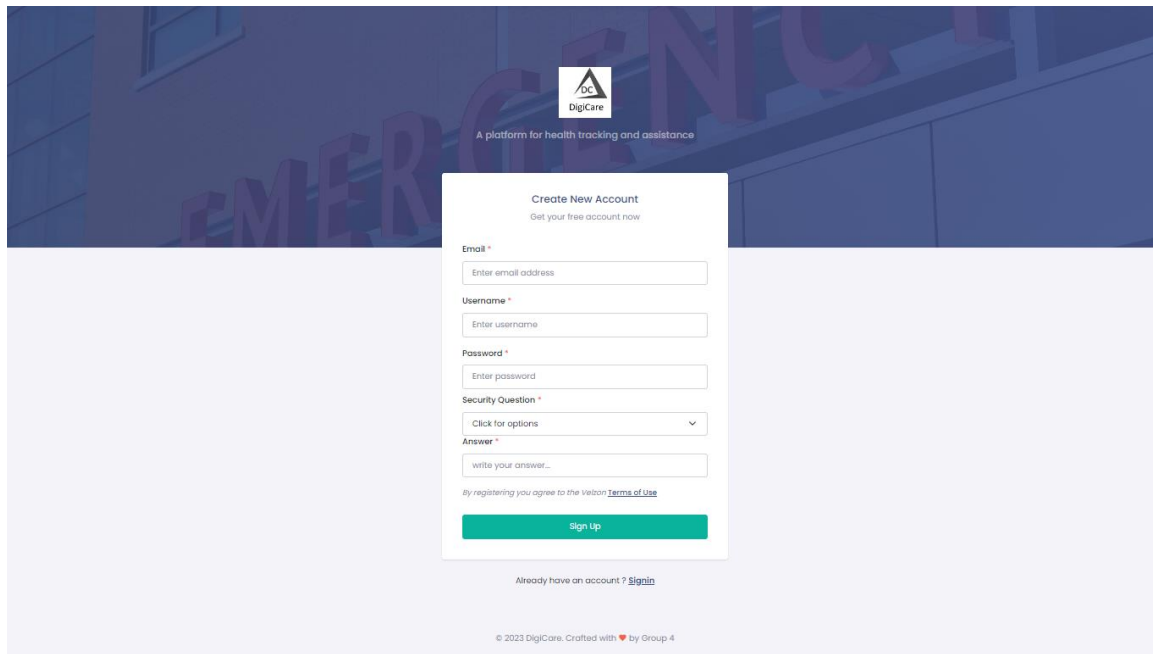


Figure 2: Log in and Sign-up button in Homepage



The image shows a web page for DigiCare, a platform for health tracking and assistance. The background is a dark blue gradient with the word "EMERGENCY" in large, stylized letters. The DigiCare logo is at the top center. Below the logo, the text "A platform for health tracking and assistance" is displayed. The main content is a white registration form titled "Create New Account" with the subtitle "Get your free account now". The form contains the following fields: "Email" (with a red asterisk), "Username" (with a red asterisk), "Password" (with a red asterisk), "Security Question" (with a red asterisk and a dropdown menu), and "Answer" (with a red asterisk). Below the form, there is a link to the "Terms of Use" and a "Sign Up" button. At the bottom, there is a link to "login" for users who already have an account. The footer text reads "© 2023 DigiCare, Crafted with ❤️ by Group 4".

bc
DigiCare

A platform for health tracking and assistance

Create New Account
Get your free account now

Email *
Enter email address

Username *
Enter username

Password *
Enter password

Security Question *
Click for options

Answer *
write your answer...

By registering you agree to the Veiton [Terms of Use](#)

Sign Up

Already have an account ? [login](#)

© 2023 DigiCare, Crafted with ❤️ by Group 4

Figure 3: Registration / Sign-up page of DigiCare

Log in

To access the DigiCare system, locate and click the "Log in" button. This action will direct the user to the login page, where the user needs to enter the email and password associated with their account. Upon correctly inputting the credentials, you will successfully log in to your account.

So, to make it simplify for a successful log in, these are the steps to follow:

1. Find and click the "Log in" button in the DigiCare system.
2. Upon clicking, the user will be redirected to the login page.
3. Input the email and password linked to your account.
4. Ensure the accuracy of the entered email and password.
5. If the credentials are correct, the user will be successfully logged into your DigiCare account.

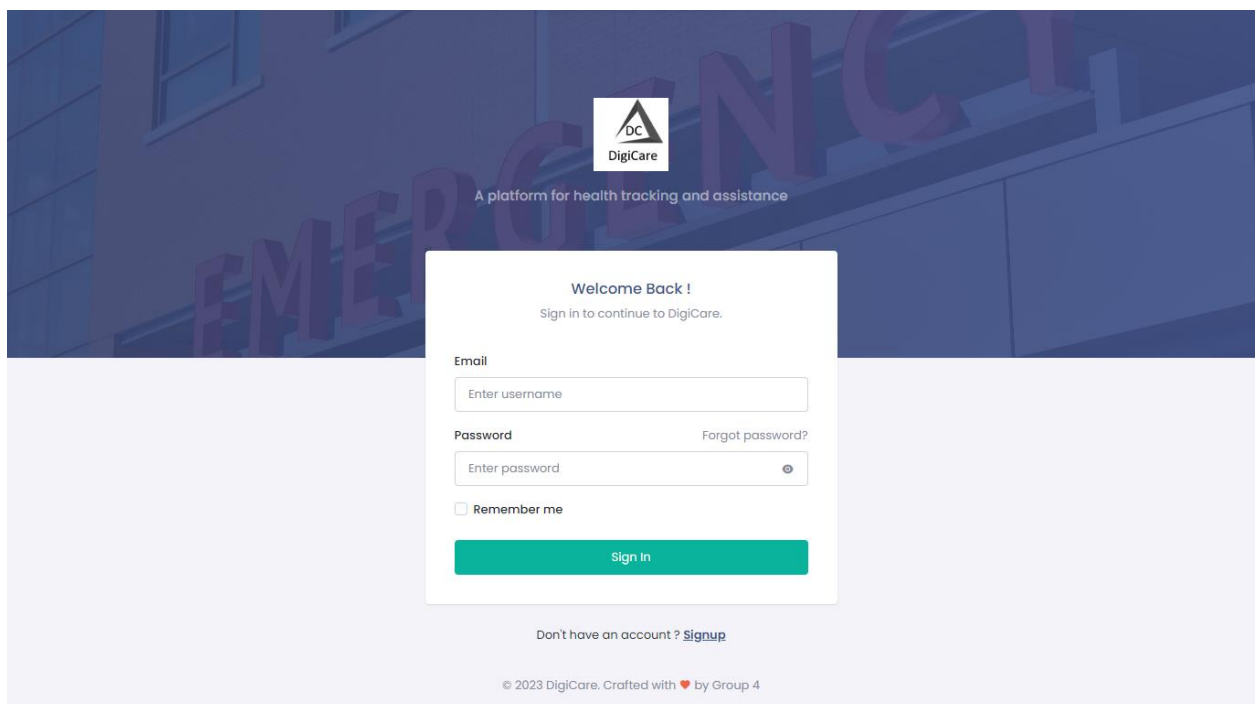


Figure 4: Log in Page

Forgetting Password / Reset password

In instances where users encounter password-related issues such as forgetting or mistyping, the DigiCare system provides a "Forget Password?" feature for resolution. To initiate the password reset process, users should locate the "Forget Password?" button on the login page. Upon clicking, they will be directed to the password reset page. Here, users need to enter their email, confirm the accurate security question used during sign-up, and provide the correct answer. If all

the information aligns accurately, the reset page will appear, allowing users to set a new password. So, to make it simplify these are the steps to follow:

1. On the login page, locate and click the "Forget Password?" button.
2. The click will redirect you to the forgotten password page.
3. Input your email and verify the security question and its answer used during sign-up.
4. Ensure that all information entered is correct and matches the records.
5. If the details are accurate, the reset page will appear, allowing you to set a new password.

Signup'. At the very bottom, small text reads '© 2023 MediAid. Crafted with ❤️ by Group 4'."/>

Oh no!
Forget Password?

Email
Enter username

Security Question *
Click for options

Answer *
write your answer...

Submit

Don't have an account ? [Signup](#)

© 2023 MediAid. Crafted with ❤️ by Group 4

Figure 5: Forget Password page

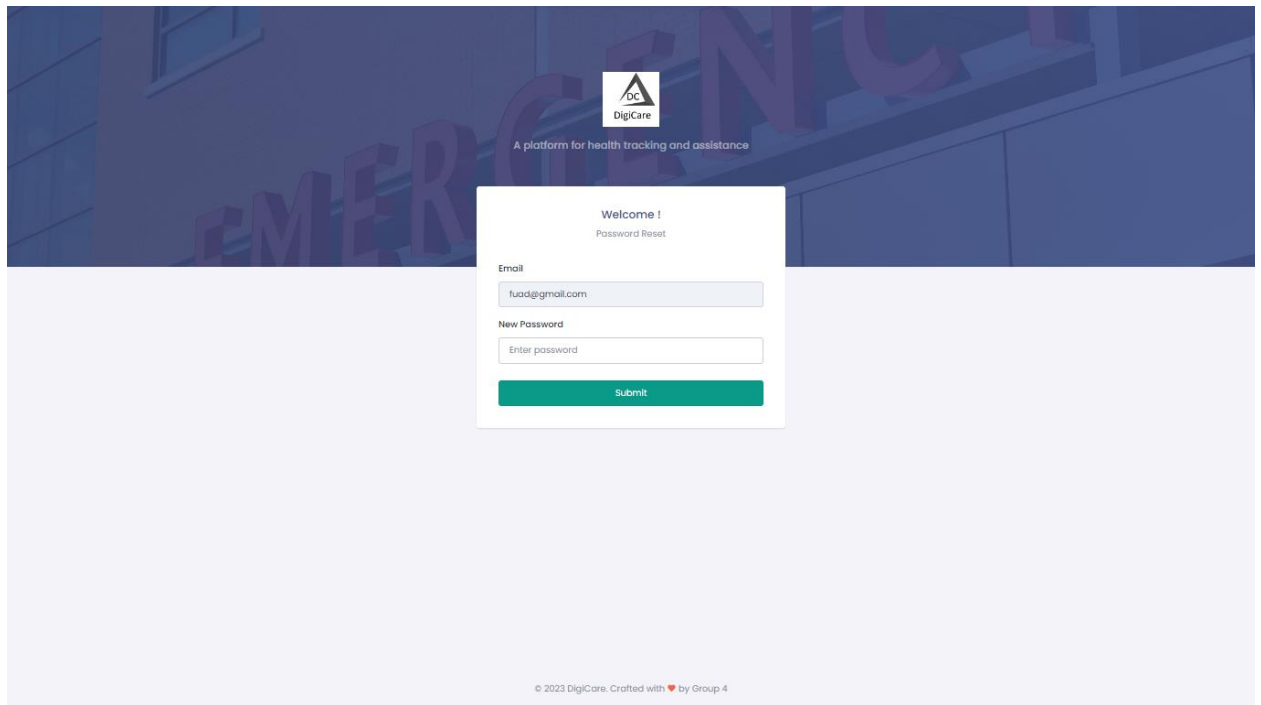


Figure 6: Reset Password page.

Profile Setup

Upon initial login, users will be automatically directed to the profile setup page. Subsequent logins will present an option labeled "Profile," which allows users to both view and input their information.

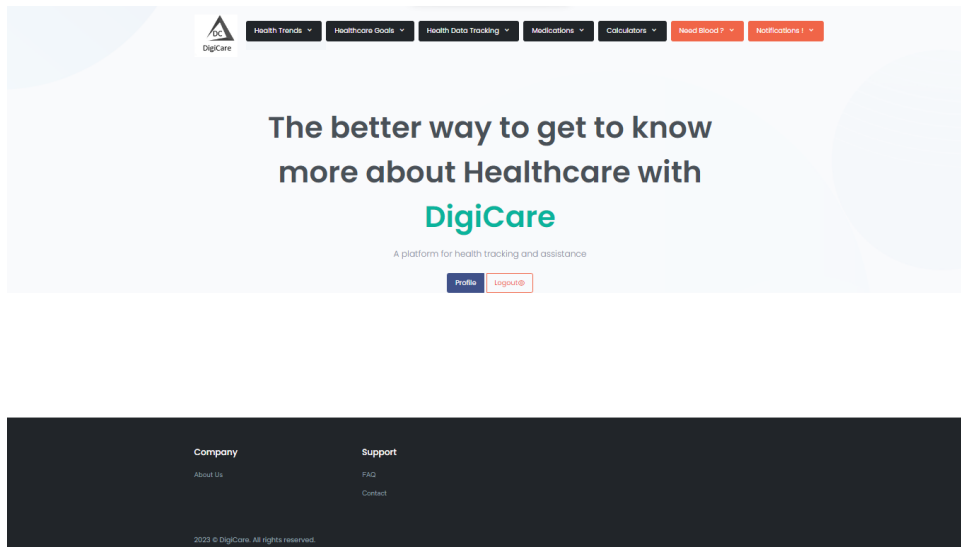


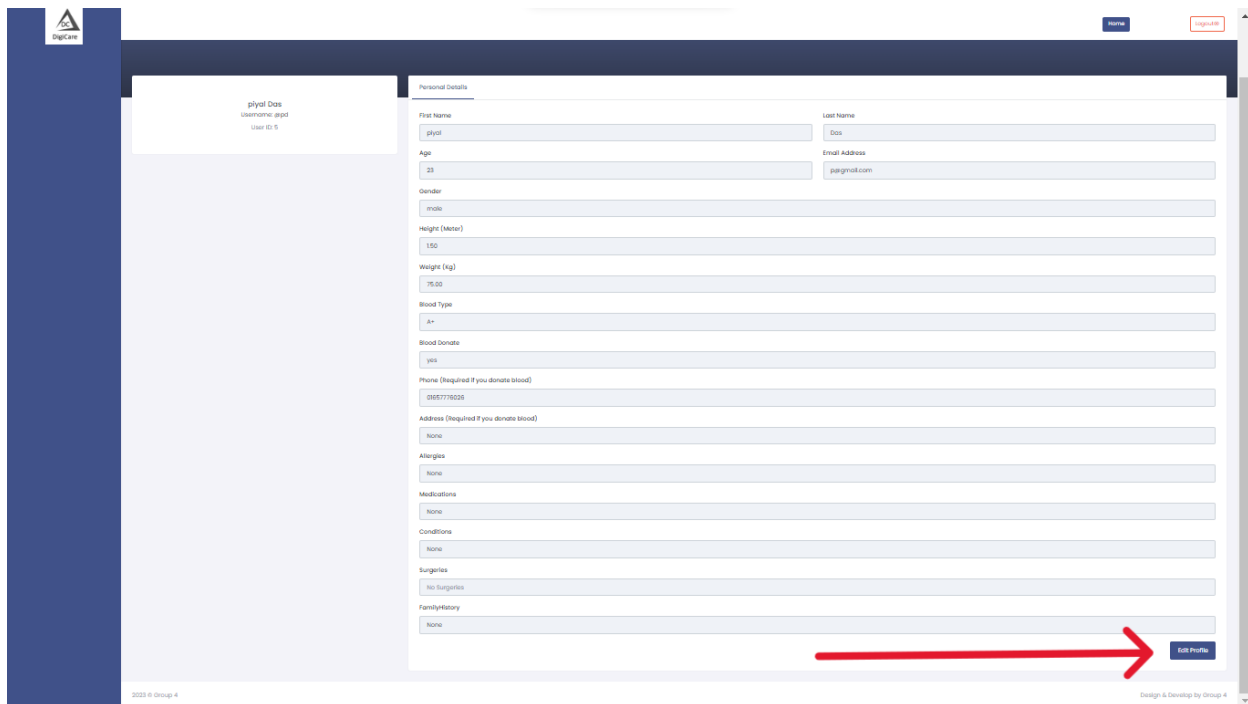
Figure 7: Profile button on the Home page

The screenshot shows the Profile Information setup page. On the left, there is a sidebar with the DigiCare logo and a user profile section for "pysal Das" (username: ap01, user ID: 1). The main content area is titled "Personal Details" and contains a form with the following fields: First Name (pysal), Last Name (Das), Age (28), Email Address (pysal@gmail.com), Gender (male), Height (180), Weight (75.00), Blood Type (A+), Blood Donor (yes), Phone (20807778326), Address (20807778326), Name, Address, Medications, Conditions, Surgeries, and Family History. Each field has a corresponding input box or dropdown menu. At the bottom right, there is a "Save Profile" button.

Figure 8: Profile Information setup page

Edit Profile

On the profile page, users have the option to edit their information. To initiate the editing process, locate the "Edit Profile" button situated beneath the profile details. Clicking this button will keep you on the same page, allowing you to modify your personal data at any time.



The screenshot displays a web application interface for editing a user profile. On the left, a dark blue sidebar contains the 'OngCare' logo and a user profile summary for 'piyol Das' (username: piyol, user ID: 5). The main content area is titled 'Personal Details' and contains a form with the following fields: First Name (piyol), Last Name (Das), Age (23), Email Address (piyol@gmail.com), Gender (male), Height (150), Weight (75.00), Blood Type (A+), Blood Donor (yes), Phone (01577750205), Address (None), Allergies (None), Medications (None), Conditions (None), Surgeries (no surgeries), and Family History (None). A red arrow points to the 'Edit Profile' button located at the bottom right of the form. The footer includes the text '© 2022 O Group 4' on the left and 'Design & Develop by Group 4' on the right.

Figure 9: Edit Profile option.

Log out

To conclude their session, users can log out by clicking the "Log Out" button. On the Home page, a prominently placed "Log Out" button is available. Additionally, users can log out from any page within the app by accessing the logout option in the header menu section. Upon

selecting the logout option, users will be redirected to the home page.

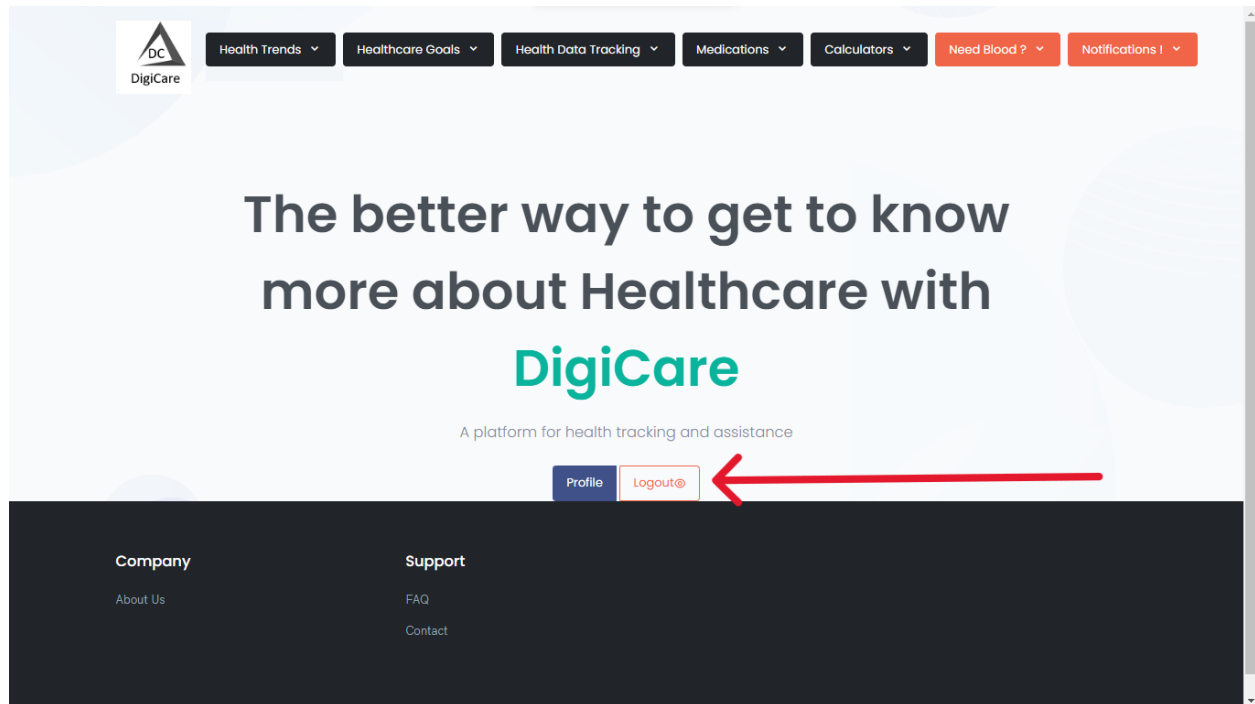


Figure 10: Log out from Home page

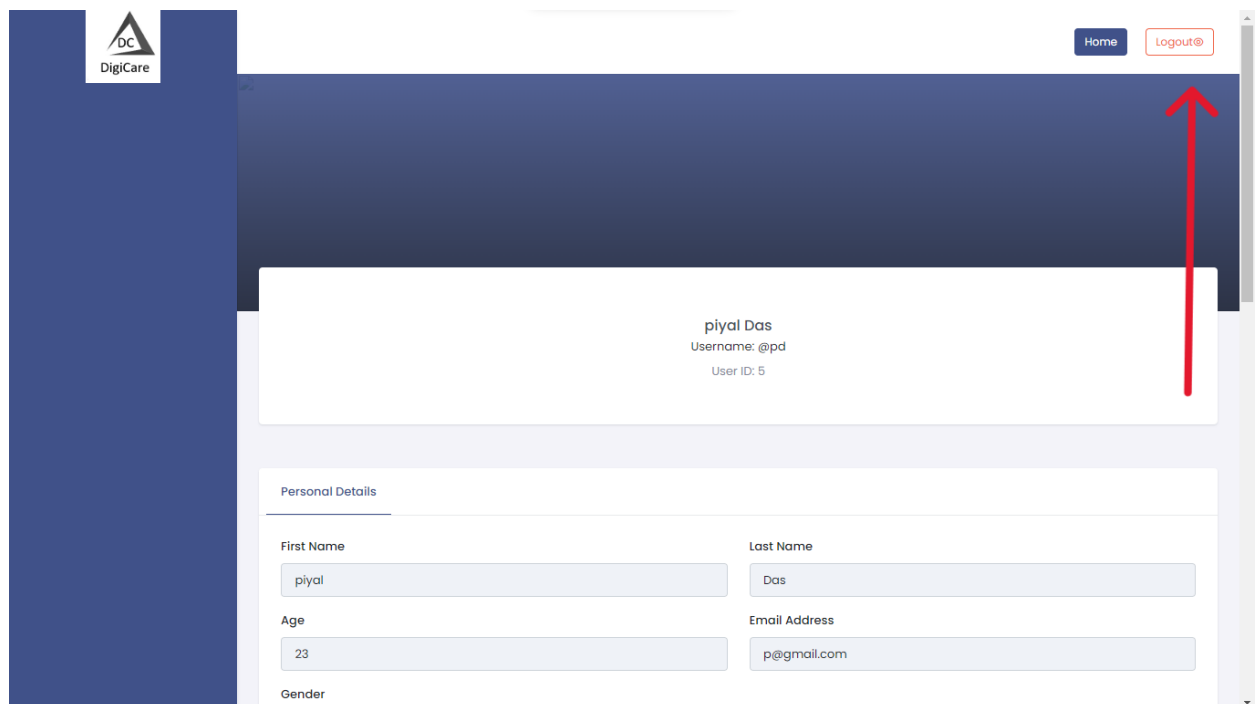


Figure 11: Log out from profile page.

Set Goals:

Within the top header menu bar of DigiCare, a key feature is highlighted—Healthcare Goals.

Clicking the "Healthcare Goals" button reveals a dropdown menu with two options: "Set a Goal"

and "See All Goal Status." To establish a personal goal, users should choose "Set a Goal,"

leading them to the "Add Goal" page. Here, users can input their goal details, including the

"Goal Name," "Goal Parameter," and "Goal Status." In "Goal Name," users describe their

objective, while in "Goal Parameter," they specify the details of the goal. The "Goal Status"

section allows users to indicate whether the goal is completed, in progress, or unsuccessful. The

fun fact is their user can not set up a health-related goal but also their daily life goals also.

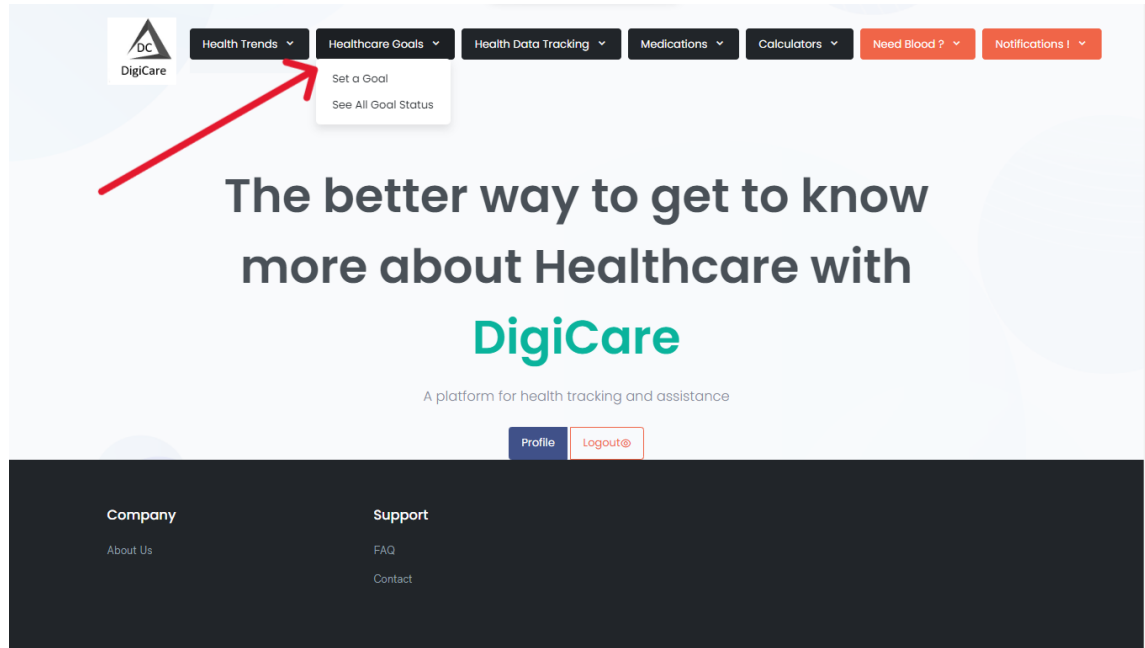


Figure 12: Health Goals in navigation bar

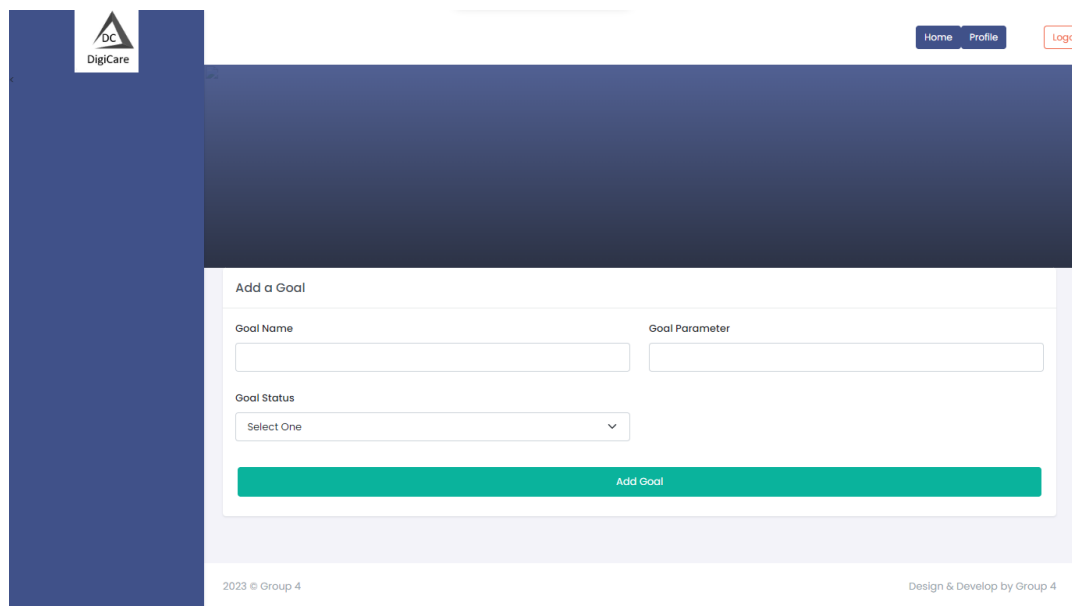


Figure 13: Add a Goal page

See All Goal Status:

Within DigiCare, users who have added goals can assess their progress by choosing "See All Goal Status" from the Healthcare Goals dropdown menu. This action directs them to the View Goals page, where they can review the status of their goals, including the number completed, in progress, or unsuccessful. Essentially, this page provides a historical overview of the user's goal achievements and outcomes. Moreover, the user can also select if the running goal is finished or failed to do that.

Goals' status

Goal ID	Goal Name	Parameter	Status	Reached?	Failed?
19	Walk	15000 steps	Reached		
20	Walk	1hr	Running	Reached	Failed
21	Sleep	5hr	Reached		

2023 © Group 4

Design & Develop by Group 4

Figure 14: View Goal Page

Health Trends / Statical Overview

One of the most captivating features of DigiCare lies in its ability to motivate users and instill a sense of uniqueness and competition. Within the navigation bar, there is a dedicated button for "Health Trends." This section features two pie charts—one illustrating the user's own goal status and the other depicting the aggregated goal status of all users. The charts showcase the percentage of goals achieved, failed, and currently in progress for both the individual user and the broader user community. It provides statistical representation, fostering motivation and a

competitive mindset for upcoming goals. Additional charts and features may be introduced in this section in the future.

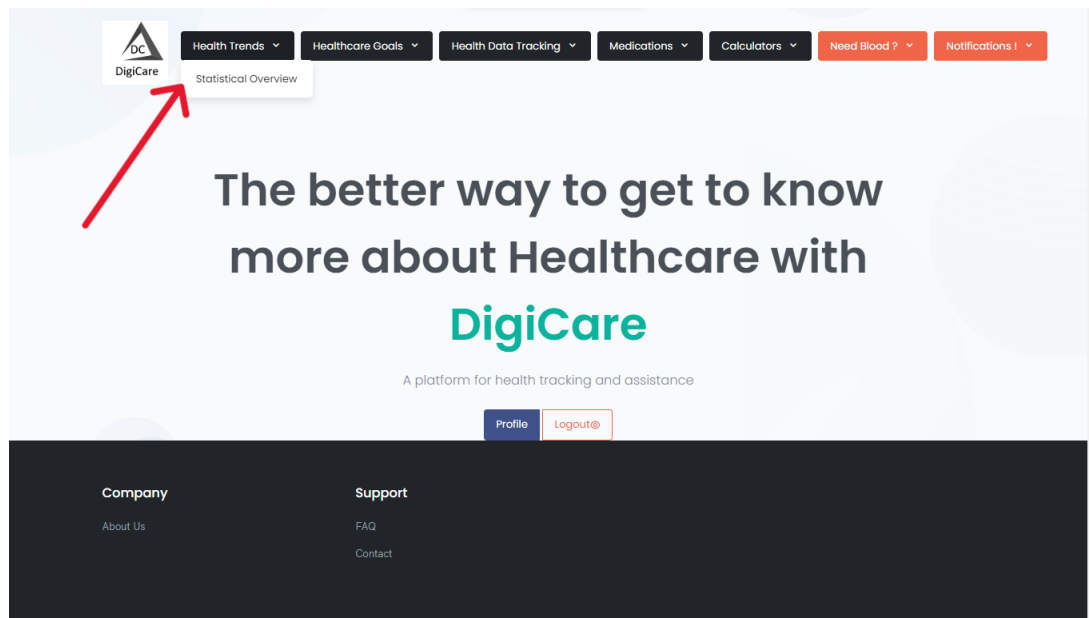


Figure 15: Static Overview Button on Health Trends

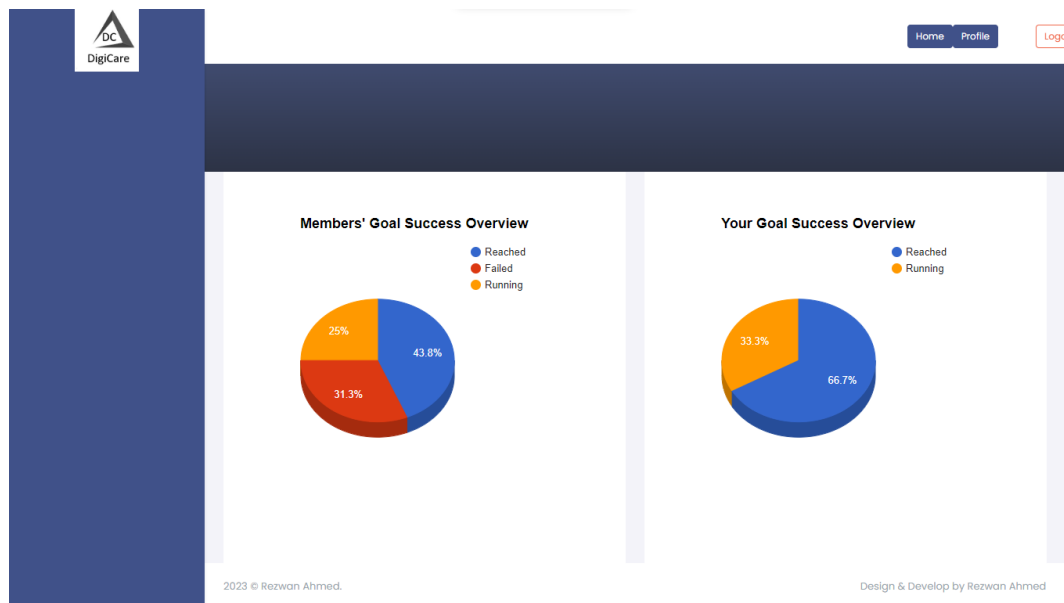


Figure 16: Goal Statistic page

Health Data Tracking / Add Health Data

DigiCare incorporates a health data tracking feature, allowing users to manually input their current or past health statuses and monitor them effortlessly. Additionally, users can record workout or exercise sessions, and this data is stored in the database for future reference. To add health data or exercise information, users should choose "Add Health Data" from the "Health Data Tracking" dropdown menu. This action directs them to the "Add Health Data" page, where they can input relevant details. Future plans include the potential integration of health devices for automatic data input.

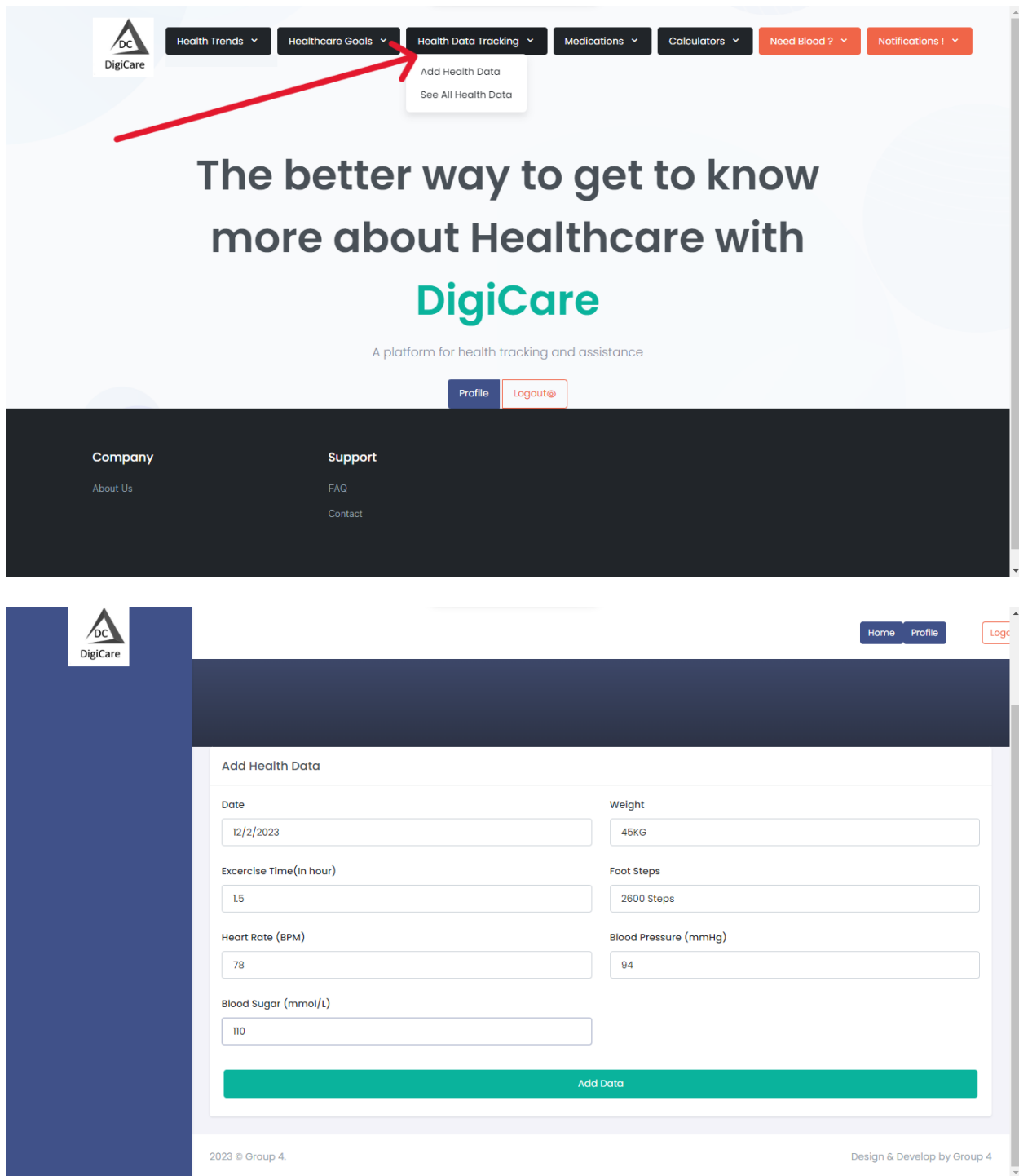
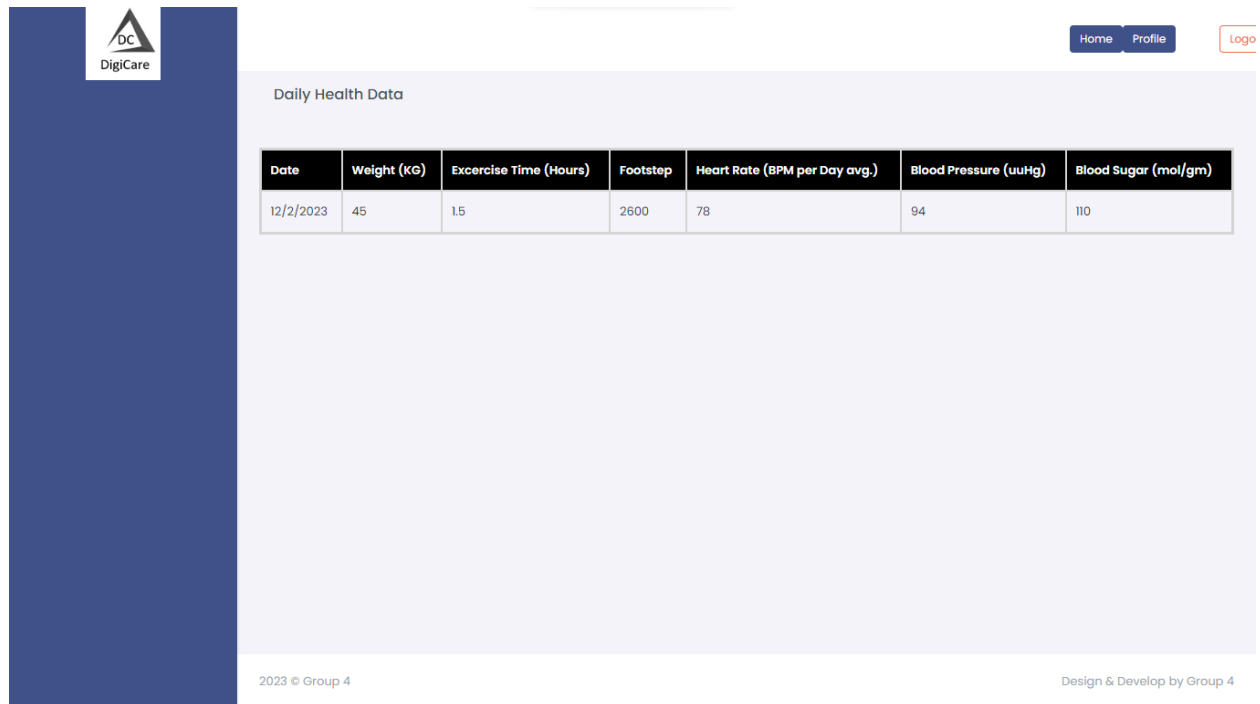


Figure 17: Add Health Data page

See All health data

Once a user has inputted health data, they can review it by selecting "See All Health Data" from the "Health Data Tracking" dropdown menu. This option provides a comprehensive history of the user's recorded health data inputs.



The screenshot displays a web application interface for viewing health data. On the left is a dark blue sidebar with the 'DigiCare' logo. The top right contains navigation links for 'Home', 'Profile', and 'Logo'. The main content area is titled 'Daily Health Data' and features a table with the following data:

Date	Weight (KG)	Exercise Time (Hours)	Footstep	Heart Rate (BPM per Day avg.)	Blood Pressure (uuHg)	Blood Sugar (mol/gm)
12/2/2023	45	1.5	2600	78	94	110

At the bottom of the page, there is a copyright notice '2023 © Group 4' on the left and 'Design & Develop by Group 4' on the right.

Figure 18: View Health Data Page

Medications / Add Medication

Within the DigiCare System, users can input their daily medication details to facilitate timely reminders for medication intake. To achieve this, users should choose "Add Medication" from the "Medications" dropdown menu. Here, users can provide all necessary information and set up reminders for their medications.

The screenshot displays the DigiCare web application interface. At the top, a navigation bar features several dropdown menus: 'Health Trends', 'Healthcare Goals', 'Health Data Tracking', 'Medications', 'Calculators', 'Need Blood?', and 'Notifications'. A red arrow highlights the 'Medications' dropdown, which is open, showing 'Add Medication' and 'See All Medications' options. Below the navigation bar is a hero section with the text 'The better way to get to know more about Healthcare with DigiCare' and a subtitle 'A platform for health tracking and assistance'. The footer contains 'Company' (About Us) and 'Support' (FAQ, Contact) links. The 'Add a Medication' form is shown with fields for 'Medication Name' (Napa), 'Choose the Medication Type' (Tablet), 'Strength (Mg)' (500), and 'When will you take this?' (Night). A green 'Add a Medication' button is at the bottom of the form.

Figure 19: Add Medication Page

See all medications.

After a user has entered their medication details, they can access a comprehensive history of their recorded medications by selecting "See All Medications" from the "Medications" dropdown menu. This option allows users to review a detailed log of their medication inputs over time. There user can select whether their medication course is completed or not also.

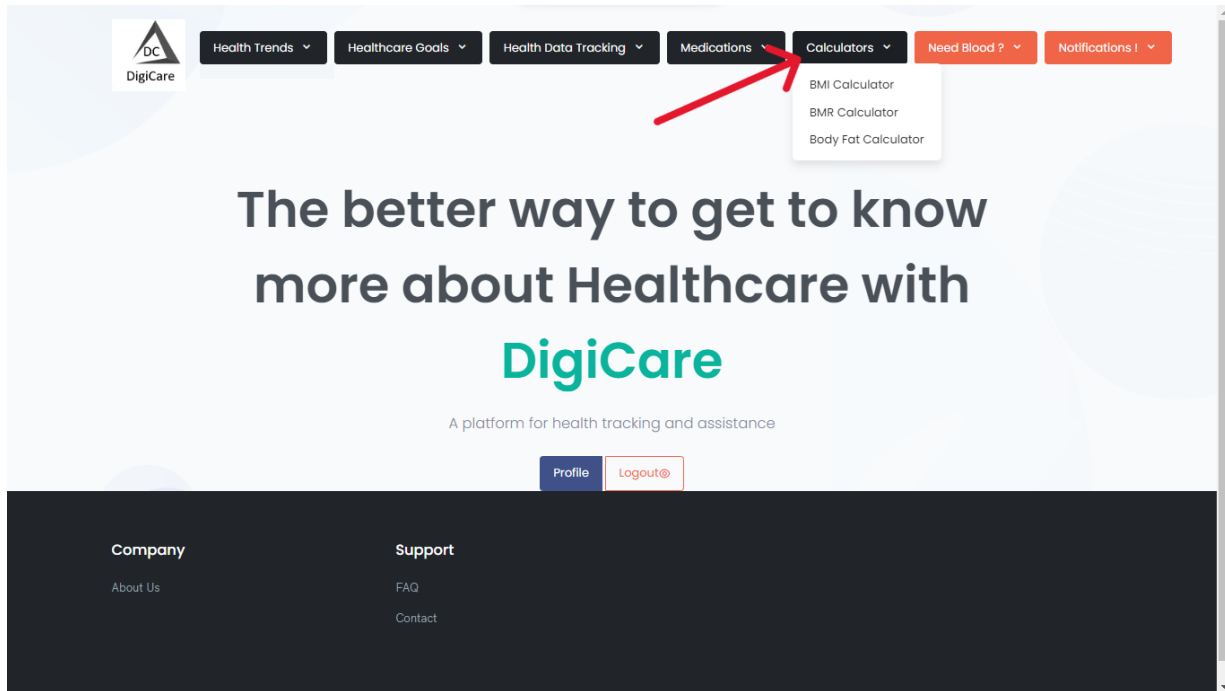
The screenshot displays the 'View Medication' page. On the left is a dark blue sidebar with the DigiCare logo. The top right features navigation links: 'Home', 'Profile', and a 'Logout' button. The main content area is titled 'Medications' and contains a table with the following data:

Medication ID	Medication Name	Medication Type	Medication Strength (mg)	Medication Time	Medication Status	Action
4	Napa	Tablet	500	Night	Course Running	<button>Course Completed</button>

At the bottom of the page, the footer includes '2023 © Rezwan Ahmed.' on the left and 'Design & Develop by Rezwan Ahmed' on the right.

Figure 20: View Medication page

Calculators / BMI Calculators



Within the "Calculator" dropdown menu of DigiCare, users can utilize the BMI calculator. This tool not only computes the Body Mass Index (BMI) based on user inputs but also presents a comprehensive chart indicating health conditions associated with different BMI ranges,

including classifications such as good, bad, etc. The BMI calculator assists users in assessing their health status and provides valuable information on their body composition.

The screenshot shows the 'The Body Mass Index (BMI) Calculator' form on the DigiCare website. The form is located on the left side of the page, with a dark blue sidebar on the far left containing the DigiCare logo. The top navigation bar includes links for 'Home', 'Profile', and 'Logout'. The form itself is titled 'The Body Mass Index (BMI) Calculator' and includes input fields for 'Weight (kg)', 'Height (m)', 'Age', and 'Gender'. The 'Gender' dropdown menu is currently set to 'Male'. A blue 'Calculate BMI' button is positioned at the bottom of the form. To the right of the form, there is a large, empty light gray area. At the bottom of the page, the footer text reads '2023 © Group 4' on the left and 'Design & Develop by Group 4' on the right.

Figure 21: BMI Calculator

The screenshot shows the results of the BMI calculation on the DigiCare website. The page layout is consistent with the previous screenshot, featuring the DigiCare logo in the sidebar and the same top navigation bar. The main content area now displays the results under the heading 'Your BMI:'. The BMI value is shown as 'BMI: 25.95 kg/m²' in red text. Below this, the input values are listed: 'Weight: 75.0 kg', 'Height: 1.7 m', 'Age: 25', and 'Gender: male'. A blue 'Calculate Again' button is located below the input values. To the right of the results, there is a large, empty light gray area. At the bottom of the page, the footer text remains '2023 © Group 4' on the left and 'Design & Develop by Group 4' on the right.

Figure 22: After Calculations of BMI showing the results

Calculator / BMR Calculator

The screenshot shows the 'Find your BMR!' section of the DigiCare application. It includes a brief definition of BMR and a form with input fields for Weight (kg), Height (m), Age, and Gender (a dropdown menu currently showing 'Male'). A 'Calculate BMR' button is at the bottom of the form. The page has a dark blue sidebar with a 'MENU' button and a top navigation bar with 'Home', 'Profile', and 'Logout' links. Footer text includes '2023 © Group 4' and 'Design & Develop by Group 4'.

Find your BMR !
The basal metabolic rate (BMR) is the amount of energy needed while resting in a temperate environment when the digestive system is inactive.

The Basal Metabolic Rate (BMR) Calculator

Weight (kg):

Height (m):

Age:

Gender:

[Calculate BMR](#)

2023 © Group 4

Design & Develop by Group 4

In the "Calculator"

dropdown menu of DigiCare, users can access the Basal Metabolic Rate (BMR) calculator. This tool calculates the estimated number of calories a person's body requires at rest. It's important to note that the data generated by the BMR calculator is not stored in the database; it serves as an on-the-fly calculation to provide users with real-time insights into their basal metabolic rate.

The screenshot shows the result of the BMR calculation. It displays 'Your BMR:' followed by '34058.85 calories/day'. Below this, it lists the input values: Weight: 89.0 kg, Height: 69.0 m, Age: 59, and Gender: male. A 'Calculate Again' button is at the bottom of the result box. The page layout is consistent with the previous screenshot, featuring the same sidebar and navigation bar.

Your BMR:

BMR:
34058.85 calories/day

Weight:
89.0 kg

Height:
69.0 m

Age:
59

Gender:
male

[Calculate Again](#)

2023 © Group 4

Design & Develop by Group 4

Figure 23: BMR Calculation

Calculator / Body Fat Calculator

DigiCare offers a Body Fat Calculator within its "Calculator" dropdown menu. This feature enables users to estimate their body fat percentage based on input data. The calculator utilizes various factors to provide a comprehensive analysis of body composition. Notably, it's important to mention that the data generated by the Body Fat Calculator is not stored in the database, ensuring privacy. This tool is designed to offer users immediate insights into their body fat

composition, aiding them in their health and fitness assessments.

The screenshot shows a web application for body fat calculation. The header includes a 'DigiCare' logo on the left and navigation links for 'Home', 'Profile', and 'Logout@' on the right. A blue sidebar is on the left. The main content area has a light blue header with the text 'Want to measure body fat?' and a sub-header 'Body Fat Calculator'. Below this is a form with the following fields: 'Select Gender:' (a dropdown menu with 'Male' selected), 'Weight (kg):' (a text input with '75'), 'Waist Circumference (cm):' (a text input with '43'), 'Wrist Circumference (cm):' (a text input with '45'), 'Hip Circumference (cm):' (a text input with '76'), and 'Forearm Circumference (cm):' (a text input with '84'). A blue 'Calculate Body Fat' button is at the bottom of the form. The footer shows '2023 © Group 4' on the left and 'Design & Develop by Group 4' on the right.

Want to measure body fat ?

The Body Fat Calculator can be used to estimate your total body fat based on specific measurements.

Body Fat Calculator

Select Gender:

Male

Weight (kg):

75

Waist Circumference (cm):

43

Wrist Circumference (cm):

45

Hip Circumference (cm):

76

Forearm Circumference (cm):

84

Calculate Body Fat

2023 © Group 4

Design & Develop by Group 4

Figure 24: Body Fat Calculation

The screenshot shows the result page of the body fat calculator. The header and sidebar are the same as in Figure 24. The main content area has a light blue header with the text 'Body Fat Result'. Below this is a box with the following information: 'Gender: male', 'Body Fat Percentage: 27.86%', and 'Your Fitness Evaluation: Obese'. A black 'Calculate Again' button is at the bottom of this box. Below this is another box with the title 'Body Fat Percentage vs Fitness Level Chart' and a table with the following data:

Body Fat Percentage	Fitness Level
< 8%	Essential Fat
8% - 14%	Athletes
14% - 18%	Fitness
18% - 25%	Average
> 25%	Obese

Body Fat Result

Gender: male

Body Fat Percentage: 27.86%

Your Fitness Evaluation: Obese

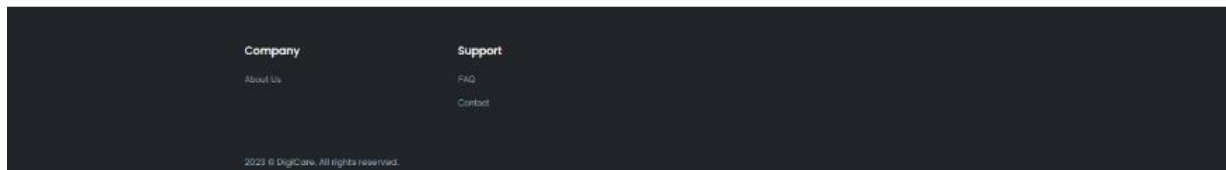
Calculate Again

Body Fat Percentage vs Fitness Level Chart

Body Fat Percentage	Fitness Level
< 8%	Essential Fat
8% - 14%	Athletes
14% - 18%	Fitness
18% - 25%	Average
> 25%	Obese

Figure 25Body Fat Calculation Result

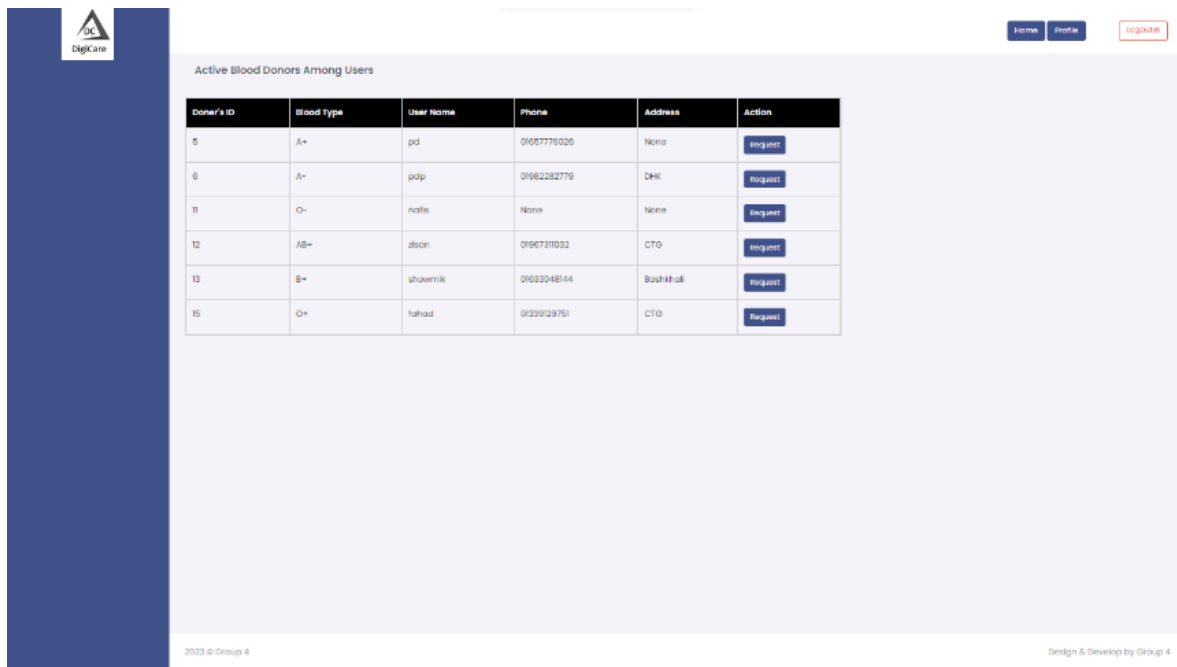
Need Blood / Blood Donation / Blood Bank



DigiCare prioritizes the well-being of its users through the Blood Bank feature, fostering a sense of community and compassion. Blood donation is a noble act that contributes to the welfare of society, saving lives in critical situations. In the Blood Bank section, users in need of blood can access a dedicated button guiding them to potential donors within the DigiCare community. This unique system ensures privacy by facilitating communication between users without disclosing personal details. Users seeking blood can send direct messages to potential donors, and if the donor accepts the request, they can proceed to coordinate and support each other. DigiCare's Blood Bank serves as a platform not only for urgent blood needs but also as a testament to the


spirit of altruism, where users can come together to make a meaningful impact on each other's lives. So, to make it simplify for a successful log in, these are the steps to follow:

1. Navigate to the Blood Bank section through the dedicated button in DigiCare.
2. In this section, users in need of blood can review profiles or user number of potential donors from the DigiCare community.
3. Users can send direct messages to potential donors expressing their need for blood. This communication system ensures privacy, as no personal details are shared at this stage.
4. If a potential donor accepts the user's request, they can proceed to coordinate the blood donation without revealing additional personal information.
5. Once the donor accepts the request, users can coordinate the logistics of the blood donation, fostering a sense of community and support within the DigiCare platform.



Donor's ID	Blood Type	User Name	Phone	Address	Action
5	A+	pdj	01667776026	None	Request
6	A-	pdjp	01662282779	DHK	Request
11	O-	natls	None	None	Request
12	AB+	alsan	0166731032	CTO	Request
13	B+	shovmik	01603048144	Bashkhal	Request
15	O+	fahad	01339129751	CTO	Request

Figure 26: Active Blood Donar's page



[Home](#) [Profile](#) [Logout@](#)

Send Donor a Request

Enter Donor's ID

12

Additional Message

Need AB+ Blood tomorrow morning at CMC clinic

Send

2023 © Group 4

Design & Develop by Group 4

Figure 27: Requesting Donar for blood.

Notifications

DigiCare's notification system ensures seamless communication between users in need of blood and potential donors. Here's a detailed breakdown:

1. When a user sends a blood request, the recipient's notification section displays the request as pending, providing real-time updates on the status of their request.
2. Simultaneously, potential donors receive a notification detailing the blood request, including essential information about the user in need. This allows donors to make informed decisions about whether they want to contribute.
3. In the donor's notification, they have the option to accept or reject the blood donation proposal. If they choose to accept, the notification guides them on how to connect with the requester while maintaining privacy. If they decline, the system automatically communicates this decision to the user in need.
4. The recipient is promptly notified of the donor's decision, providing transparency and enabling quick and informed action based on the donor's choice.
5. DigiCare is committed to enhancing its notification system to include additional features and functionalities, ensuring users stay informed about various aspects of their health and well-being. Future updates may introduce notifications for health data tracking, goal achievements, and other pertinent information, contributing to a comprehensive and user-

friendly experience.

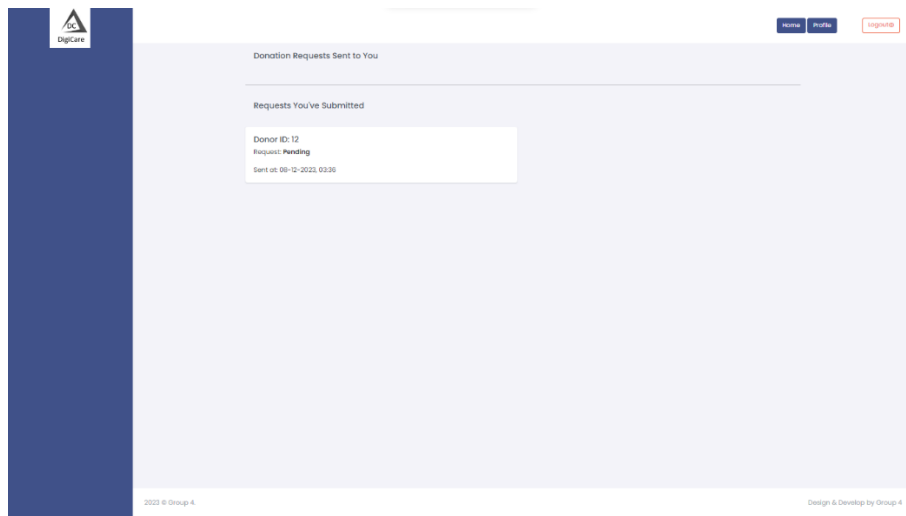


Figure 28:Receiver side notification pane- Receiver after sending request to the Blood Donar

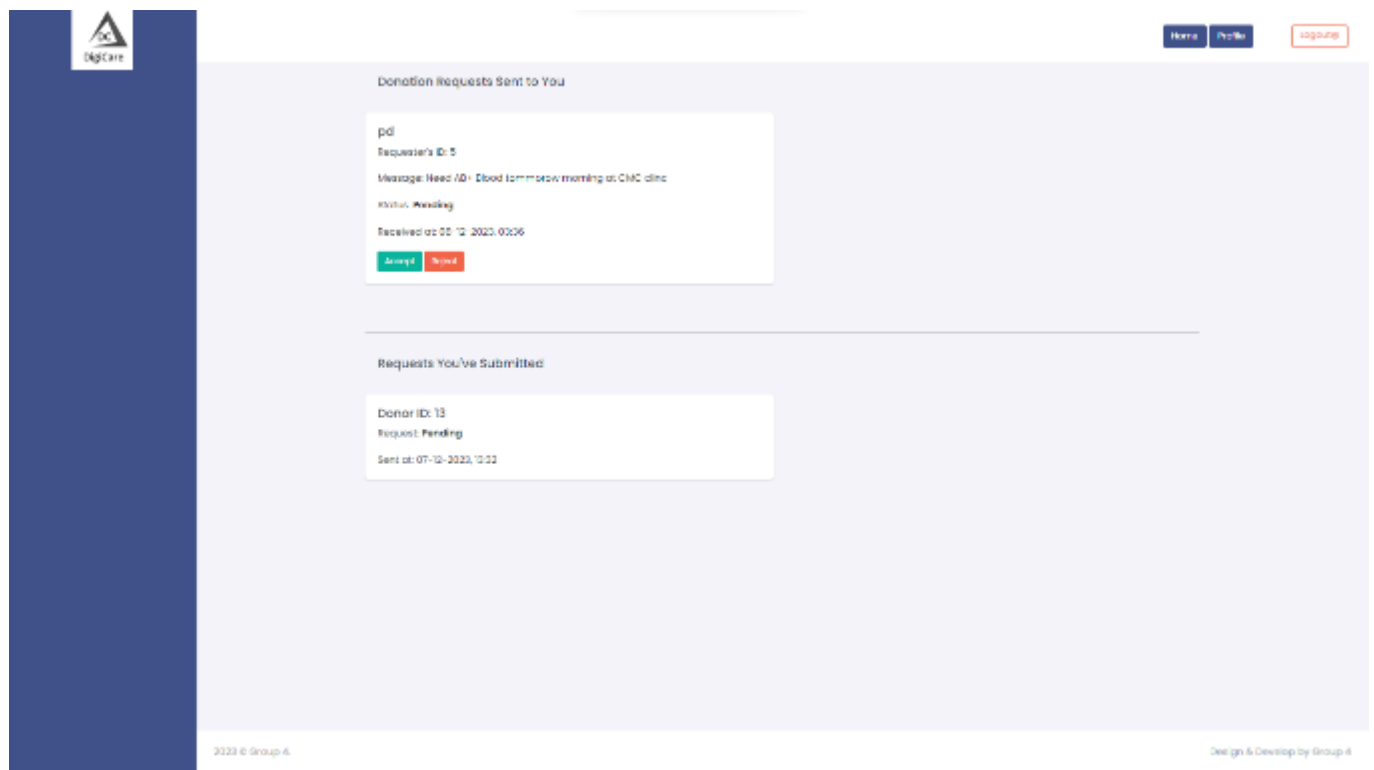


Figure 29: Donar side notification pane- after receiving the request from the needy

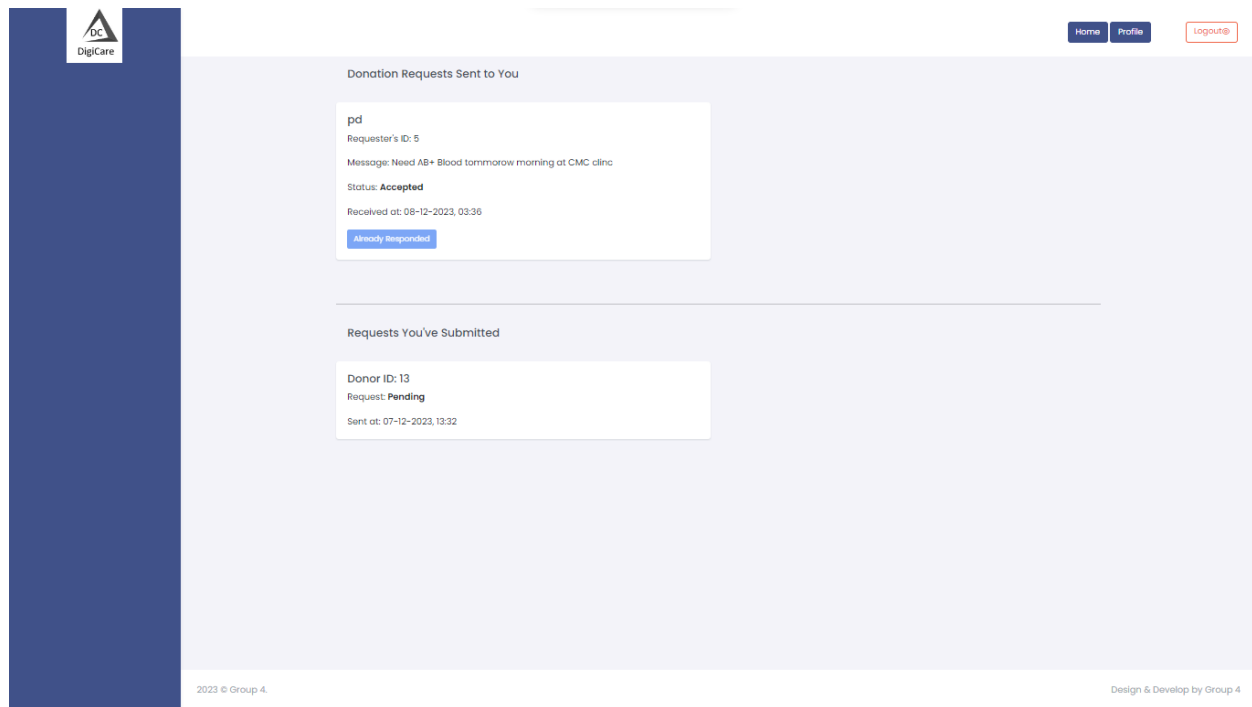
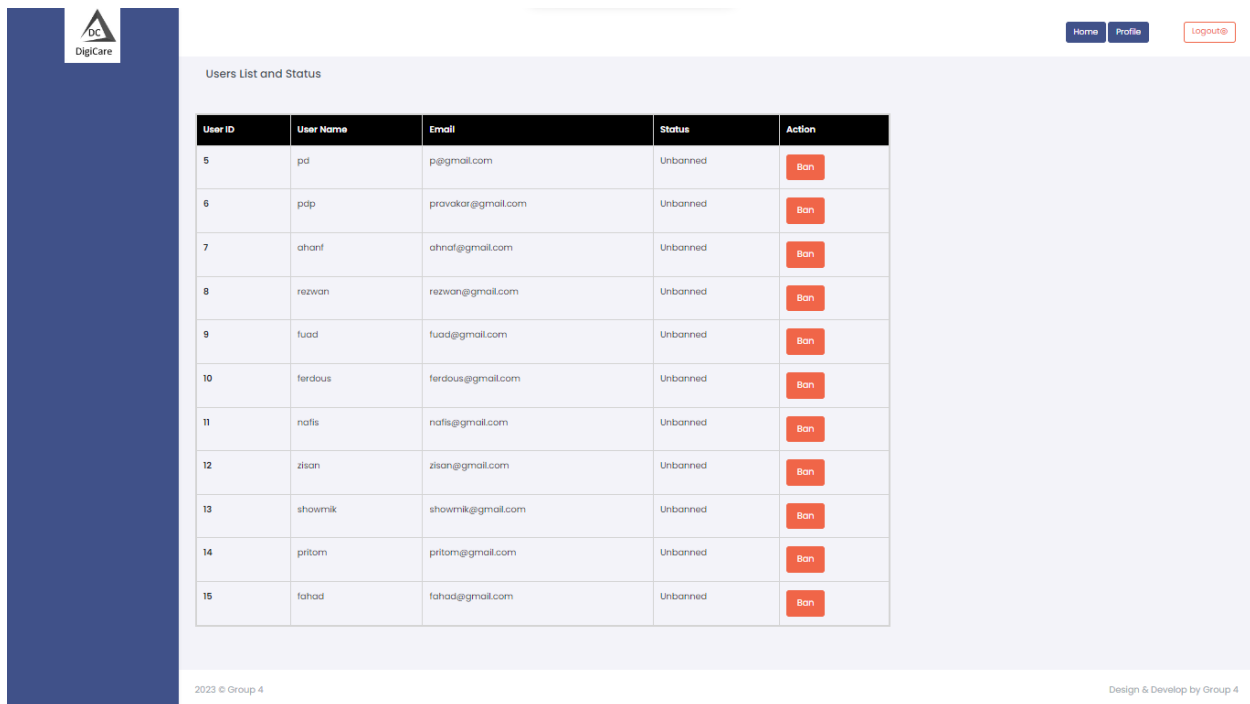


Figure 30: Donar Accepted the request.

Ban / Unban from Admin Pannel

The administration panel of this project is meticulously crafted using phpMyAdmin, providing administrators with a comprehensive toolset for system oversight. The admin dashboard, a focal point of control, offers a unique and powerful feature—the capability to ban and unban users.

This feature serves as a crucial security measure, allowing administrators to regulate user access to the platform. Banning a user restricts their ability to log in, offering an effective means to manage and address any issues that may arise. The manual creation of the admin panel through phpMyAdmin ensures a tailored and secure environment, empowering administrators to maintain the integrity and security of the health management system. The ban/unban feature enhances the overall governance of user access, contributing to the robustness and reliability of the platform.



User ID	User Name	Email	Status	Action
5	pd	p@gmail.com	Unbanned	Ban
6	pdp	pravakar@gmail.com	Unbanned	Ban
7	ahanf	ahnaf@gmail.com	Unbanned	Ban
8	rezwan	rezwan@gmail.com	Unbanned	Ban
9	fuad	fuad@gmail.com	Unbanned	Ban
10	ferdous	ferdous@gmail.com	Unbanned	Ban
11	nafis	nafis@gmail.com	Unbanned	Ban
12	zisan	zisan@gmail.com	Unbanned	Ban
13	showmik	showmik@gmail.com	Unbanned	Ban
14	pritom	pritom@gmail.com	Unbanned	Ban
15	fahad	fahad@gmail.com	Unbanned	Ban

Figure 31: User list and status from the admin panel

Backend Development

The backend of our project is developed using Flask, a lightweight web framework for Python. It handles data, manages user authentication, and serves as the core logic for our application. It is utilized for routing, handling HTTP requests, and managing our application's logic. We use SQLAlchemy to interact with a MySQL database, allowing us to define models and perform database operations seamlessly.

```
digicare.py > ...
1  from flask import Flask, render_template, request, redirect, url_for, session, flash, jsonify
2  from flask_migrate import Migrate
3  from flask_sqlalchemy import SQLAlchemy
4  from werkzeug.security import generate_password_hash, check_password_hash
5  from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user
6  from sqlalchemy import Column, Integer, String, Numeric
7  import pymysql
8  from pytz import timezone
9  from datetime import datetime
10
11
12  app = Flask(__name__)
13  app.config['SECRET_KEY'] = 'CSE471'
14  app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://digicareAdmin:abcd1234@localhost:3306/digicare'
15  app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
16  db = SQLAlchemy(app)
17  migrate = Migrate(app, db)
18  |
19
20
```

Flask-Login is employed for user authentication. Users can register, log in, and log out of the system. Additionally, the system includes a setup process to gather essential user information after registration. Also in case the password for logging in is forgotten by the user there is a forget password and reset password option. The application includes various routes and views for all the features mentioned above. There is also a goal model which is

used to track user goals, a med model which stores information about medications users are taking and a HealthData model captures health-related information such as weight.

Register

```
136 @app.route('/')
137 def index():
138     return render_template('index.html')
139
140 @app.route("/register", methods=['GET', 'POST'])
141 def register():
142     if request.method == 'POST':
143         email = request.form.get('useremail')
144         username = request.form.get('username')
145         password = request.form.get('userpassword')
146         seq_q = request.form.get('seq_q')
147         seq_a = request.form.get('seq_a')
148         #aType = request.form.get('aType')
149         #print(aType)
150         user = User.query.filter_by(email=email).first()
151         if user:
152             flash('Email already exists.', category='error')
153         else:
154             new_user = User(email=email, username=username, password=generate_password_hash(password, method='sha256'), seq_q=seq_q, seq_a=seq_a, aT
155             db.session.add(new_user)
156             db.session.commit()
157             return redirect(url_for('index'))
158
159     return render_template("signup.html", user=current_user)
160
```

Login, Logout

```
120
121 login_manager = LoginManager()
122 login_manager.login_view = 'login'
123 login_manager.init_app(app)
124
125 @login_manager.user_loader
126 def load_user(id):
127     return User.query.get(int(id))
128
```

```

162
163 @app.route("/login", methods=['GET', 'POST'])
164 def login():
165     if request.method == 'POST':
166         email = request.form.get('useremail')
167         password = request.form.get('userpassword')
168         user = User.query.filter_by(email=email).first()
169         if user:
170             if check_password_hash(user.password, password):
171                 if user.flag == "Banned":
172                     return redirect(url_for('AccountBanned'))
173                 flash('Logged in successfully.', category='success')
174                 login_user(user, remember=True)
175                 if user.Fname==None or user.Lname==None or user.age==None or user.gender==None or user.height==None or user.weight==None or user.bloodType==None:
176                     return redirect(url_for('setup'))
177             return redirect(url_for('index'))
178         else:
179             flash('Incorrect password, try again.', category='error')
180     else:
181         flash('Email does not exist.', category='error')
182     return render_template("login.html", user=current_user)
183
184
185

```

```

213
214 @app.route("/logout")
215 @login_required
216 def logout():
217     logout_user()
218     return redirect(url_for('login'))
219

```

Setup Profile

```

187
188 @app.route("/setup", methods=['GET', 'POST'])
189 @login_required
190 def setup():
191     user=current_user
192     if request.method == 'POST':
193         user.Fname = request.form.get('Fname')
194         user.Lname = request.form.get('Lname')
195         user.age = request.form.get('age')
196         user.gender = request.form.get('gender')
197         user.height = request.form.get('height')
198         user.weight = request.form.get('weight')
199         user.bloodType = request.form.get('bloodType')
200         user.bloodDonate = request.form.get('bloodDonate')
201         user.phone = request.form.get('phone')
202         user.address = request.form.get('address')
203         user.allergies = request.form.get('allergies')
204         user.medications = request.form.get('medications')
205         user.conditions = request.form.get('conditions')
206         user.surgeries = request.form.get('surgeries')
207         user.familyHistory = request.form.get('familyHistory')
208         db.session.commit()
209         return redirect(url_for('index'))
210     return render_template("setup.html", user=current_user)
211

```

Forget Password, Reset Password

```
220
221
222 @app.route("/forgotpassword", methods=['GET', 'POST'])
223 def forgotpassword():
224     if request.method == 'POST':
225         email = request.form.get('useremail')
226         seq_q = request.form.get('seq_q')
227         seq_a = request.form.get('seq_a')
228         user = User.query.filter_by(email=email).first()
229         if user:
230             if user.seq_q == seq_q and user.seq_a == seq_a:
231                 return render_template("resetpassword.html", email=user.email)
232             else:
233                 return render_template("index.html", user=current_user)
234         else:
235             return render_template("index.html", user=current_user)
236     return render_template("forgotpassword.html", user=current_user)
237
238
```

```
239
240 @app.route("/resetpassword", methods=['GET', 'POST'])
241 def resetpassword():
242     if request.method == 'POST':
243         email = request.form.get('useremail')
244         password = request.form.get('userpassword')
245         user = User.query.filter_by(email=email).first()
246         print(email)
247         print(password)
248         if user:
249             user.password = generate_password_hash(password, method='sha256')
250             db.session.commit()
251             return redirect(url_for('login'))
252         else:
253             return render_template("index.html", user=current_user)
254     return render_template("resetpassword.html", user=current_user)
255
256
```


Goal Model

```
267 @app.route("/AddGoal", methods=['GET', 'POST'])
268 @login_required
269 def AddGoal():
270     if request.method == 'POST':
271         name = request.form.get('gname')
272         description = request.form.get('gdesc')
273         status = request.form.get('gs')
274         user_id = current_user.id
275         new_goal = Goal(name=name, description=description, status=status, user=user_id)
276         db.session.add(new_goal)
277         db.session.commit()
278         return redirect(url_for('ViewGoals'))
279     return render_template("AddGoal.html", user=current_user)
280
281
282
283 @app.route("/ViewGoals", methods=['GET', 'POST'])
284 @login_required
285 def ViewGoals():
286     goals = Goal.query.filter_by(user=current_user.id).all()
287     return render_template("ViewGoals.html", user=current_user, goals=goals)
```

```
290 @app.route("/GoalReached/<int:id>", methods=['GET', 'POST'])
291 @login_required
292 def GoalReached(id):
293     goal = Goal.query.get_or_404(id)
294
295
296     goal.status = "Reached"
297     db.session.commit()
298     return redirect(url_for('ViewGoals'))
299
300
301
302 @app.route("/GoalFailed/<int:id>", methods=['GET', 'POST'])
303 @login_required
304 def GoalFailed(id):
305     goal = Goal.query.get_or_404(id)
306     goal.status = "Failed"
307     db.session.commit()
308     return redirect(url_for('ViewGoals'))
```

Med Model

```
344 @app.route("/AddMed", methods=['GET', 'POST'])
345 @login_required
346 def AddMed():
347     if request.method == 'POST':
348         name= request.form.get('mname')
349         type = request.form.get('mtype')
350         strength = request.form.get('mstrength')
351         time = request.form.get('mtime')
352         user_id = current_user.id
353         new_med = Med(name=name, type=type, strength=strength, time=time, user=user_id)
354         db.session.add(new_med)
355         db.session.commit()
356         return redirect(url_for('ViewMeds'))
357     return render_template("AddMed.html", user=current_user)
358
359
360
361 @app.route("/ViewMeds", methods=['GET', 'POST'])
362 @login_required
363 def ViewMeds():
364     meds = Med.query.filter_by(user=current_user.id).all()
365     return render_template("ViewMeds.html", user=current_user, meds=meds)
366
367
368 @app.route("/MedDone/<int:id>", methods=['GET', 'POST'])
369 @login_required
370 def MedDone(id):
371     med = Med.query.get_or_404(id)
372     med.status = "Course Completed"
373     db.session.commit()
374     return redirect(url_for('ViewMeds'))
```

HealthData Model

```
314 @app.route("/AddHealthData", methods=['GET', 'POST'])
315 @login_required
316 def AddHealthData():
317     if request.method == 'POST':
318         date = request.form.get('date')
319         weight = request.form.get('weight')
320         etime = request.form.get('etime')
321         fstep = request.form.get('fstep')
322         heartRate = request.form.get('heartRate')
323         bloodPressure = request.form.get('bloodPressure')
324         bloodSugar = request.form.get('bloodSugar')
325         user_id = current_user.id
326         new_healthdata = HealthData(date=date, weight=weight, etime=etime, fstep=fstep, heartRate=heartRate, bloodPressure=bloodPressure, bloodSugar=bloodSugar)
327         db.session.add(new_healthdata)
328         db.session.commit()
329         return redirect(url_for('ViewHealthData'))
330     return render_template("AddHealthData.html", user=current_user)
331
332
333
334
335 @app.route("/ViewHealthData", methods=['GET', 'POST'])
336 @login_required
337 def ViewHealthData():
338     healthdata = HealthData.query.filter_by(user=current_user.id).all()
339
340     return render_template("ViewHealthData.html", user=current_user, healthdata=healthdata)
```

There are also database models of User

```
25 class User(UserMixin, db.Model):
26     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
27     username = db.Column(db.String(100), nullable=False)
28     email = db.Column(db.String(100), nullable=False)
29     password = db.Column(db.String(100), nullable=False)
30     seq_q = db.Column(db.String(100))
31     seq_a = db.Column(db.String(100))
32     Fname = db.Column(db.String(100))
33     Lname = db.Column(db.String(100))
34     age = db.Column(db.Integer)
35     gender = db.Column(db.String(100))
36     height = db.Column(db.Numeric(5,2))
37     weight = db.Column(db.Numeric(5,2))
38     bloodType = db.Column(db.String(100))
39     bloodDonate = db.Column(db.String(100))
40     phone = db.Column(db.String(100))
41     address = db.Column(db.String(100))
42     allergies = db.Column(db.String(100))
43     medications = db.Column(db.String(100))
44     conditions = db.Column(db.String(100))
45     surgeries = db.Column(db.String(100))
46     familyHistory = db.Column(db.String(100))
47     aType = db.Column(db.String(100))
48     flag = db.Column(db.String(100))
49
50     def __repr__(self) -> str:
51         return f'{self.id} - {self.email} - {self.username} - {self.password}'
```


Goal

```
54
55 class Goal(db.Model):
56
57     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
58     name = db.Column(db.String(100), nullable = False)
59     description = db.Column(db.String(100), nullable = False)
60     status = db.Column(db.String(100), nullable = False)
61     user = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
62
63
64     def __repr__(self) -> str:
65         return f'{self.id} - {self.name} - {self.description}'
66
```

Med

```
68
69
70 class Med(db.Model):
71     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
72     name = db.Column(db.String(100), nullable = False)
73     type = db.Column(db.String(100), nullable = False)
74     strength = db.Column(db.String(100), nullable = False)
75     time = db.Column(db.String(100), nullable = False)
76     user = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
77     status = db.Column(db.String(100), default="Course Running")
78
79
80     def __repr__(self) -> str:
81         return f'{self.id} - {self.name} - {self.type} - {self.strength} - {self.time}'
82
83
```

HealthData

```
83
84
85 class HealthData(db.Model):
86
87     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
88     date = db.Column(db.String(100), nullable = False)
89     weight = db.Column(db.Integer, nullable = False)
90     etime = db.Column(db.String(100), nullable = False)
91     fstep = db.Column(db.Integer, nullable = False)
92     heartRate = db.Column(db.Integer, nullable = False)
93     bloodPressure = db.Column(db.String(100), nullable = False)
94     bloodSugar = db.Column(db.Integer, nullable = False)
95     user = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
96
97
98     def __repr__(self) -> str:
99         return f'{self.id} - {self.date} - {self.etime} - {self.heartRate} - {self.bloodPressure} - {self.bloodSugar}'
100
101
```

DonationRequest

```
102
103
104 class DonationRequest(db.Model):
105     id = db.Column(db.Integer, primary_key=True, autoincrement=True)
106     user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
107     user_name= db.Column(db.String(100), nullable=False)
108     donor_id = db.Column(db.Integer, nullable=False)
109     message = db.Column(db.String(100), nullable=False)
110     is_seen = db.Column(db.String(100), default=None)
111     accepted = db.Column(db.String(100), default=None)
112     timestamp = db.Column(db.DateTime, default=datetime.utcnow)
113
114
115
116     def __repr__(self) -> str:
117         return f'{self.id} - {self.donor_id} - {self.donor_id} - {self.message} - {self.is_seen} - {self.accepted}'
118
119
```

In the profile setup model if the users are potential blood donors (bloodDonate=`yes`) their info is displayed in the list of potential blood donors.

Blood Donor Routes

```
379 @app.route("/BloodDonorList", methods=['GET', 'POST'])
380 @login_required
381 def BloodDonorList():
382     users = User.query.filter_by(aType="User", flag='Unbanned', bloodDonate='yes' ).all()
383     return render_template("BloodDonorList.html", user=current_user, users=users)
384
385
386 @app.route('/request_blood', methods=['GET', 'POST'])
387 def request_blood():
388     if request.method == 'POST':
389         requester_id = current_user.id
390         donor_id = request.form.get('donor_id')
391         user_name = current_user.username
392         message = request.form.get('message')
393
394         # timezone
395         bd_tz = timezone('Asia/Dhaka')
396         current_time = datetime.now().astimezone(bd_tz)
397
398
399         new_request = DonationRequest(user_id=requester_id, donor_id=donor_id, user_name=user_name, message=message, accepted="Pending", timestamp=current_time)
400         db.session.add(new_request)
401         db.session.commit()
402         return redirect(url_for('index'))
403     return render_template("request_blood.html", user=current_user)
```

Notifications

The application implements a notification system for blood donation requests. Users can send and receive donation requests, with options to accept or reject them.

```

410 @app.route("/BloodDonationNotifications", methods=['GET', 'POST'])
411 @login_required
412 def BloodDonationNotifications():
413     current_user_id = current_user.id
414
415     # donation requests made by the current user (requester)
416     requester_requests = DonationRequest.query.filter_by(user_id=current_user_id).all()
417
418     # donation requests received by the current user (donor)
419     donor_requests = DonationRequest.query.filter_by(donor_id=current_user_id).all()
420
421     # additional information of donors based on donor_id (Not Using this anymore)
422     donor_ids = [request.donor_id for request in requester_requests]
423     donors_info = User.query.filter(User.id.in_(donor_ids)).all()
424
425     return render_template("BloodDonationNotifications.html",
426                           user=current_user,
427                           requester_requests=requester_requests,
428                           donor_requests=donor_requests,
429                           donors_info=donors_info)

```

```

433 @app.route("/NotificationsAccept/<int:id>", methods=['GET', 'POST'])
434 @login_required
435 def NotificationsAccept(id):
436     request = DonationRequest.query.get_or_404(id)
437     if request.accepted == "Pending":
438         request.accepted = "Accepted"
439         db.session.commit()
440     return redirect(url_for('BloodDonationNotifications'))
441
442 @app.route("/NotificationsReject/<int:id>", methods=['GET', 'POST'])
443 @login_required
444 def NotificationsReject(id):
445     request = DonationRequest.query.get_or_404(id)
446     if request.accepted == "Pending":
447         request.accepted = "Rejected"
448         db.session.commit()
449     return redirect(url_for('BloodDonationNotifications'))

```

Statistics

A statistics route provides an overview of goal statistics for all users and personalized statistics for the logged-in user.

```
455 # Route to get goal statistics for all users
456 @app.route('/statistics', methods=['GET', 'POST'])
457 @login_required
458 def get_statistics():
459     #with app.app_context():
460     total_users = Goal.query.distinct(Goal.id).count()
461
462     running_count = Goal.query.filter_by(status='Running').count()
463     reached_count = Goal.query.filter_by(status='Reached').count()
464     failed_count = Goal.query.filter_by(status='Failed').count()
465     running_percentage = round((running_count / total_users) * 100, 2) if total_users > 0 else 0
466     reached_percentage = round((reached_count / total_users) * 100, 2) if total_users > 0 else 0
467     failed_percentage = round((failed_count / total_users) * 100, 2) if total_users > 0 else 0
468     current_user_id = current_user.id
469     c_running_count = Goal.query.filter_by(user=current_user_id, status='Running').count()
470     c_reached_count = Goal.query.filter_by(user=current_user_id, status='Reached').count()
471     c_failed_count = Goal.query.filter_by(user=current_user_id, status='Failed').count()
472     c_total_count = c_running_count + c_reached_count + c_failed_count
473     c_running_percentage = round((c_running_count / c_total_count) * 100, 2) if c_total_count > 0 else 0
474     c_reached_percentage = round((c_reached_count / c_total_count) * 100, 2) if c_total_count > 0 else 0
475     c_failed_percentage = round((c_failed_count / c_total_count) * 100, 2) if c_total_count > 0 else 0
476
477     return render_template('statistics.html',
478         running_percentage= running_percentage,
479         reached_percentage= reached_percentage,
480         failed_percentage= failed_percentage,
481         c_running_percentage= c_running_percentage,
482         c_reached_percentage= c_reached_percentage,
483         c_failed_percentage= c_failed_percentage)
```

Calculators

The application features calculators for BMI, BMR, and body fat index, providing users with health-related insights.

BMI

```

526 @app.route("/BmiIndex", methods=['GET', 'POST'])
527 @login_required
528 def BmiIndex():
529     return render_template('BmiIndex.html')
530
531
532 @app.route('/calculate', methods=['POST'])
533 def calculate():
534     if request.method == 'POST':
535         weight = float(request.form['weight'])
536         height = float(request.form['height'])
537         age = int(request.form['age'])
538         gender = request.form['gender']
539
540         bmi = calculate_bmi(weight, height)
541         return render_template('BmiResult.html', weight=weight, height=height, bmi=bmi, age=age, gender=gender)
542
543 def calculate_bmi(weight, height):
544     bmi = weight / (height * height)
545     return round(bmi, 2)

```

BMR

```

550 @app.route("/BmrIndex", methods=['GET', 'POST'])
551 @login_required
552 def BmrIndex():
553     return render_template('BmrIndex.html')
554
555 @app.route('/calculate_bmr', methods=['POST'])
556 def calculate_bmr():
557     if request.method == 'POST':
558         weight = float(request.form['weight'])
559         height = float(request.form['height'])
560         age = int(request.form['age'])
561         gender = request.form['gender']
562
563         bmr = calculate_bmr_value(weight, height, age, gender)
564         return render_template('BmrResult.html', weight=weight, height=height, age=age, gender=gender, bmr=bmr)
565
566 def calculate_bmr_value(weight, height, age, gender):
567     if gender == 'male':
568         bmr = 88.362 + (13.397 * weight) + (4.799 * height * 100) - (5.677 * age)
569     else: # female
570         bmr = 447.593 + (9.247 * weight) + (3.098 * height * 100) - (4.330 * age)
571
572     return round(bmr,2)

```

Body Fat Index

```

582 @app.route('/calculate_bodyfat', methods=['POST'])
583 def calculate_bodyfat():
584     if request.method == 'POST':
585         gender = request.form['gender']
586         weight = float(request.form['weight'])
587         waist = float(request.form['waist'])
588         wrist = float(request.form['wrist'])
589         hip = float(request.form['hip'])
590         forearm = float(request.form['forearm'])
591
592         body_fat = calculate_body_fat_percentage(gender, weight, waist, wrist, hip, forearm)
593         fitness_evaluation = evaluate_fitness(body_fat, gender)
594
595         return render_template('bodyfat_result.html', gender=gender, body_fat=body_fat, fitness_evaluation=fitness_evaluation)
596
597 def calculate_body_fat_percentage(gender, weight, waist, wrist, hip, forearm):
598     if gender == 'male':
599         body_fat = (weight * 0.732) + (waist / 3.785) - (wrist * 0.393) - (hip * 0.394) + (forearm * 0.434) - 28.57
600     else: # female
601         body_fat = (weight * 0.732) + (waist / 3.786) - (wrist * 0.393) - (hip * 0.394) + (forearm * 0.434) - 28.57
602     return round(body_fat, 2)

```

Fitness Evaluation

```

604 def evaluate_fitness(body_fat, gender):
605     if gender == 'male':
606         if body_fat < 8:
607             return "Essential Fat"
608         elif 8 <= body_fat < 14:
609             return "Athletes"
610         elif 14 <= body_fat < 18:
611             return "Fitness"
612         elif 18 <= body_fat < 25:
613             return "Average"
614         else:
615             return "Obese"
616     else: # female
617         if body_fat < 21:
618             return "Essential Fat"
619         elif 21 <= body_fat < 25:
620             return "Athletes"
621         elif 25 <= body_fat < 31:
622             return "Fitness"
623         elif 31 <= body_fat < 38:
624             return "Average"
625         else:
626             return "Obese"

```

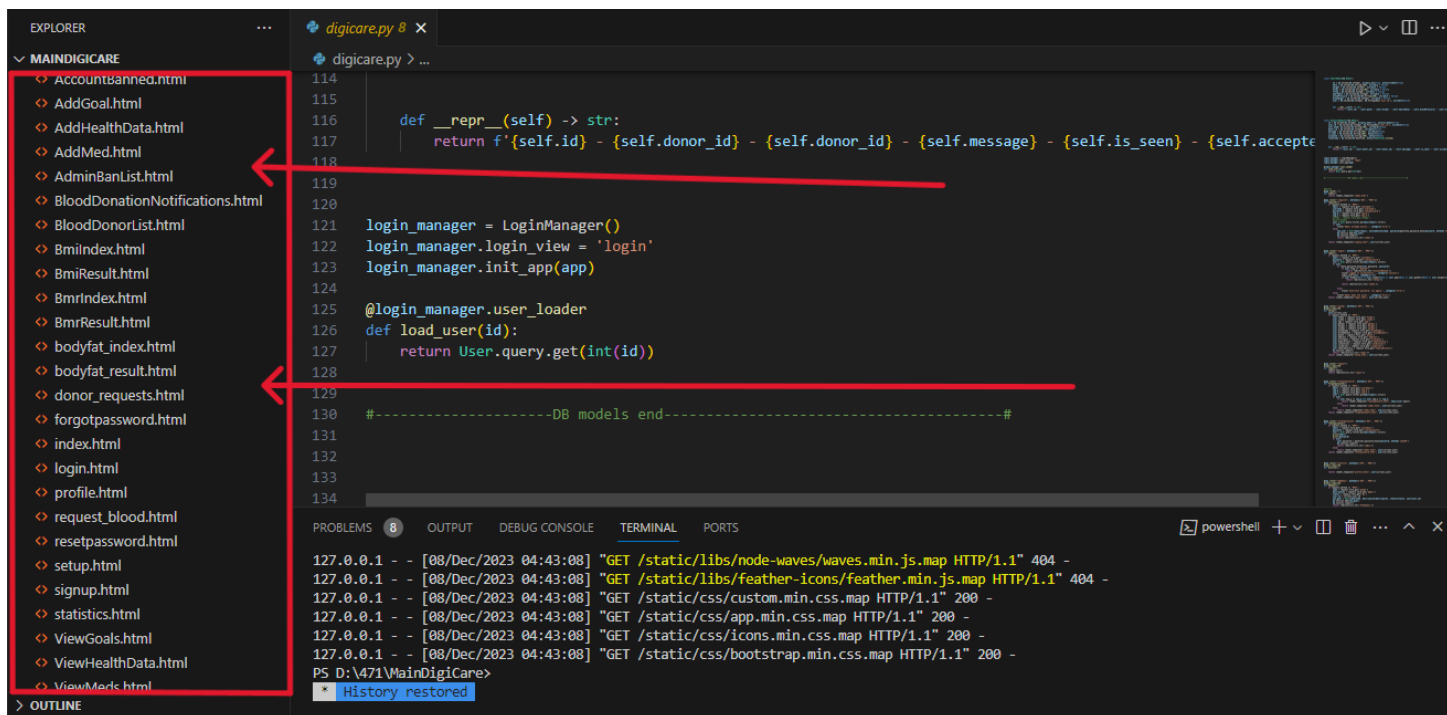
Admin Panel

An admin panel is available for administrative purposes. Admins can ban and unban users as needed.

```
489 @app.route("/AdminBanList", methods=['GET', 'POST'])
490 @login_required
491 def AdminBanList():
492     users = User.query.filter_by(aType="User").all()
493     return render_template("AdminBanList.html", user=current_user, users=users)
494
495
496 @app.route("/AdminBan/<int:id>", methods=['GET', 'POST'])
497 @login_required
498 def AdminBan(id):
499     user = User.query.get_or_404(id)
500
501
502     user.flag = "Banned"
503     db.session.commit()
504     return redirect(url_for('AdminBanList'))
505
506 @app.route("/AdminUnban/<int:id>", methods=['GET', 'POST'])
507 @login_required
508 def AdminUnban(id):
509     user = User.query.get_or_404(id)
510
511
512     user.flag = "Unbanned"
513     db.session.commit()
514     return redirect(url_for('AdminBanList'))
515
516 @app.route("/AccountBanned", methods=['GET', 'POST'])
517 def AccountBanned():
518     return render_template("AccountBanned.html", user=current_user)
```


Frontend Development

The front end of the DigiCare project is meticulously designed to provide a seamless and user-friendly experience. All pages share a common foundation built with HTML, CSS, and JavaScript, incorporating elements from Bootstrap for enhanced design aesthetics. The front-end files are systematically organized within the templates folder. Through efficient routing, each page establishes a connection with the database, ensuring the cohesive functioning of the entire project. This structured approach to front-end development not only ensures consistency in design but also facilitates a dynamic and responsive user interface. The integration of HTML, CSS, JavaScript, and Bootstrap elements creates an aesthetically pleasing and intuitively navigable interface, contributing to the overall success and user satisfaction of the DigiCare platform.



Index.html

The provided `index.html` file is the frontend code for the main page of DigiCare. It is designed using a combination of HTML, CSS, and JavaScript, with additional styling elements borrowed from Bootstrap for a visually appealing and responsive layout. The document includes metadata for defining the page title, description, and author. It references various external libraries and resources for icons, styling, and functionality. The navigation bar is thoughtfully organized with dropdown menus, directing users to different sections of the platform related to health tracking and assistance. The hero section showcases a call-to-action with login and signup buttons. The footer contains links for company information, support, and copyright details. The JavaScript

files at the end handle dynamic page elements and functionality. Overall, the code reflects a well-structured and aesthetically pleasing homepage for the DigiCare platform.

```
<section class="section pb-0 hero-section" id="hero">
  <div class="bg-overlay bg-overlay-pattern"></div>
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-lg-10 col-sm-8">
        <div class="text-center mt-lg-5 pt-5">
          <h1 class="display-4 fw-semibold mb-3 lh-base">The better way to get to know more about Healthcare with <span
            class="text-success">DigiCare </span></h1>
          <p class="lead text-muted lh-base">A platform for health tracking and assistance</p>

          <div class="d-flex gap-2 justify-content-center mt-4">
            {% if current_user.is_authenticated %}
            <div class="btn-group me-0">
              <a href="/profile" class="btn btn-primary me-1">Profile</a>
              <a href="/logout" class="btn btn-outline-danger">Logout<i class="ri-eye-line align-middle ms-0"></i></a>
            </div>
            {% else %}
            <div class="me-0">
              <a href="/login" class="btn btn-danger me-0">Login<i class="ri-eye-line align-middle ms-0"></i></a>
              <a href="/register" class="btn btn-primary">Sign Up</a>
            </div>
            {% endif %}
          </div>
        </div>
      </div>
    </div>
  </div>
<!-- end row -->
</div>
</div>
```

This part is basically for the main viewpoint which the users going to view in the first page.

In every static page, the same thing is running repeatedly. But according to the features, there is some modifications such as:

```

templates > request_blood.html > html > body > footer.footer > div.container-fluid > div.row > div.col-sm-6
127     <div class="col-lg-12">
128         <div class="card">
129             <div class="card-header align-items-center d-flex">
130                 <h4 class="card-title mb-0 flex-grow-1">Send Donor a Request</h4>
131                 <div class="flex-shrink-0">
132                 </div>
133             </div>
134         </div><!-- end card header -->
135         <div class="card-body">
136             <div class="live-preview">
137                 <div class="row gy-4">
138                     <div class="col-xxl-3 col-md-6">
139                         <div>
140                             <label for="mname" class="form-label">Enter Donor's ID</label>
141                             <input type="text" class="form-control" id="donor_id" name="donor_id">
142                         </div>
143                     </div>
144                 </div>
145                 <div class="col-xxl-3 col-md-6">
146                     <div>
147                         <label for="basiInput" class="form-label">Additional Message</label>
148                         <input type="text" class="form-control" id="message" name="message">
149                     </div>
150                 </div>
151             </div>
152             <div class="btn btn-link fw-medium text-decoration-none text-dark">
153                 <button class="btn btn-success w-100" type="submit">Send</button>
154             </div>
155         </form>
156     </div>
157     <!--end row-->
158 </div>
159 <div class="d-none code-view">
160 </div>
161 </div>
162 </div>
163 </div>
164 </div>
165 </div>
166 </div>
167 </div>
168 </div>
169 </div>
170 </div>
171 </div>
172 </div>
173 </div>

```

Figure 32: Blood Donar page

```

templates > BloodDonorList.html > html > body > div.main-content > div.page-content > div.container-fluid > footer.footer > div.container-fluid
162
163
164 /* Alternate Row Colors */
165 .table-striped tbody tr:nth-of-type(odd) {
166     background-color: #fafafa; /* Alternate row color */
167 }
168
169 /* Table Header Styling */
170 .table thead th {
171     background-color: #007bff; /* Table header background color */
172     color: #ffffff; /* Table header text color */
173     border-color: #007bff; /* Table header border color */
174 }
175
176 /* Table Body Text Color */
177 .table tbody td {
178     color: #333333; /* Table body text color */
179 }
180 </style>
181 </head>
182 <body>
183     <div class="col-xxl-8">
184         <h5 class="mb-3 p-3">Active Blood Donors Among Users</h5>
185         <!-- Table with Styling -->
186         <div class="table-responsive">
187             <table class="table table-bordered table-hover">
188                 <thead class="table-light">
189                     <tr>
190                         <th scope="col">Doner's ID</th>
191                         <th scope="col">Blood Type</th>
192                         <th scope="col">User Name</th>
193                         <th scope="col">Phone</th>
194                         <th scope="col">Address</th>
195                         <th scope="col">Action</th>
196                     </tr>
197                 </thead>
198                 <tbody>
199                     {% for u in users %}
200                     <tr>
201                         <td>{{ u.id }}</td>
202                         <td>{{ u.bloodType }}</td>
203                         <td>{{ u.username }}</td>
204                         <td>{{ u.phone }}</td>
205                         <td>{{ u.address }}</td>
206                         <td><a href="/request_blood" class="btn btn-primary btn-sm">Request</a></td>
207                     </tr>
208                     {% endfor %}

```

Figure 33: Blood Donar List

```

> index.html < statistics.html 9+
templates > < statistics.html > html > body > div.main-content > div.page-content > div.container-fluid
112
113
114     <div class="position-relative mx-n4 mt-n4">
115         <div class="profile-wid-bg profile-setting-img">
116             
117             <div class="overlay-content">
118                 <div class="text-end p-3">
119
120                 </div>
121             </div>
122         </div>
123     </div>
124
125     <div class="row">
126         <div class="col-lg-6">
127
128             <!-- 1st pie chart -->
129
130             <html>
131                 <head>
132                     <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
133                     <script type="text/javascript">
134                         google.charts.load("current", {packages:["corechart"]});
135                         google.charts.setOnLoadCallback(drawChart);
136                         function drawChart() {
137                             var data1 = google.visualization.arrayToDataTable([
138                                 ['Task', 'Percentage'],
139                                 ['Reached',    {{ reached_percentage }}],
140                                 ['Failed',    {{ failed_percentage }}],
141                                 ['Running',   {{ running_percentage }}]
142                             ]);
143
144                             var options1 = {
145                                 title: "Members' Goal Success Overview",
146                                 titleTextStyle: {
147
148                                     fontSize: 18,
149                                     bold: true,
150                                     italic: false
151                                 },
152                                 is3D: true,
153                             };
154
155                             var chart1 = new google.visualization.PieChart(document.getElementById('first_chart'));
156                             chart1.draw(data1, options1);
157                         }
158                     </script>
159                 </head>

```

Figure 34: Statistic page

```

templates > AdminBanList.html > html > body > div.main-content > body > div.page-content > div.container-fluid > footer.footer > div.container-fluid > div.row
155 .table-striped tbody tr:nth-of-type(odd) {
156     background-color: #fafafa; /* Alternate row color */
157 }
158
159 /* Table Header Styling */
160 .table thead th {
161     background-color: #007bff; /* Table header background color */
162     color: #ffffff; /* Table header text color */
163     border-color: #007bff; /* Table header border color */
164 }
165
166 /* Table Body Text Color */
167 .table tbody td {
168     color: #333333; /* Table body text color */
169 }
170 </style>
171 </head>
172 <body>
173
174 <div class="page-content">
175     <div class="container-fluid">
176
177         <div class="position-relative mx-n4 mt-n4">
178
179         </div>
180         <div class="col-xxl-8">
181             <h5 class="mb-3 p-3">Users List and Status</h5>
182             <!-- Captions -->
183             <table class="table caption-top table-nowrap">
184                 <table class="table table-bordered table-hover">
185                     <thead class="table-light">
186                         <tr>
187                             <th scope="col">User ID</th>
188                             <th scope="col">User Name</th>
189                             <th scope="col">Email</th>
190                             <th scope="col">Status</th>
191                             <th scope="col">Action</th>
192                         </tr>
193                     </thead>
194                     <tbody>
195                         <tr>
196                             <th scope="row">{{u.id}}</th>
197                             <td>{{u.username}}</td>
198                             <td>{{u.email}}</td>
199                             <td>{{u.flag}}</td>
200                             <td>{{if u.flag == 'Unbanned' %}}
201                                 <a href="/AdminBan/{{u.id}}" class="btn btn-danger">Ban</a></td>
202                             <td>{{else %}}
203                                 <a href="/AdminUnban/{{u.id}}" class="btn btn-success">Unban</a></td>
204                             <td>{{endif %}}
205                         </tr>
206                     </tbody>
207                 </table>
208             </div>
209         </div>
210     </div>
211 </div>

```

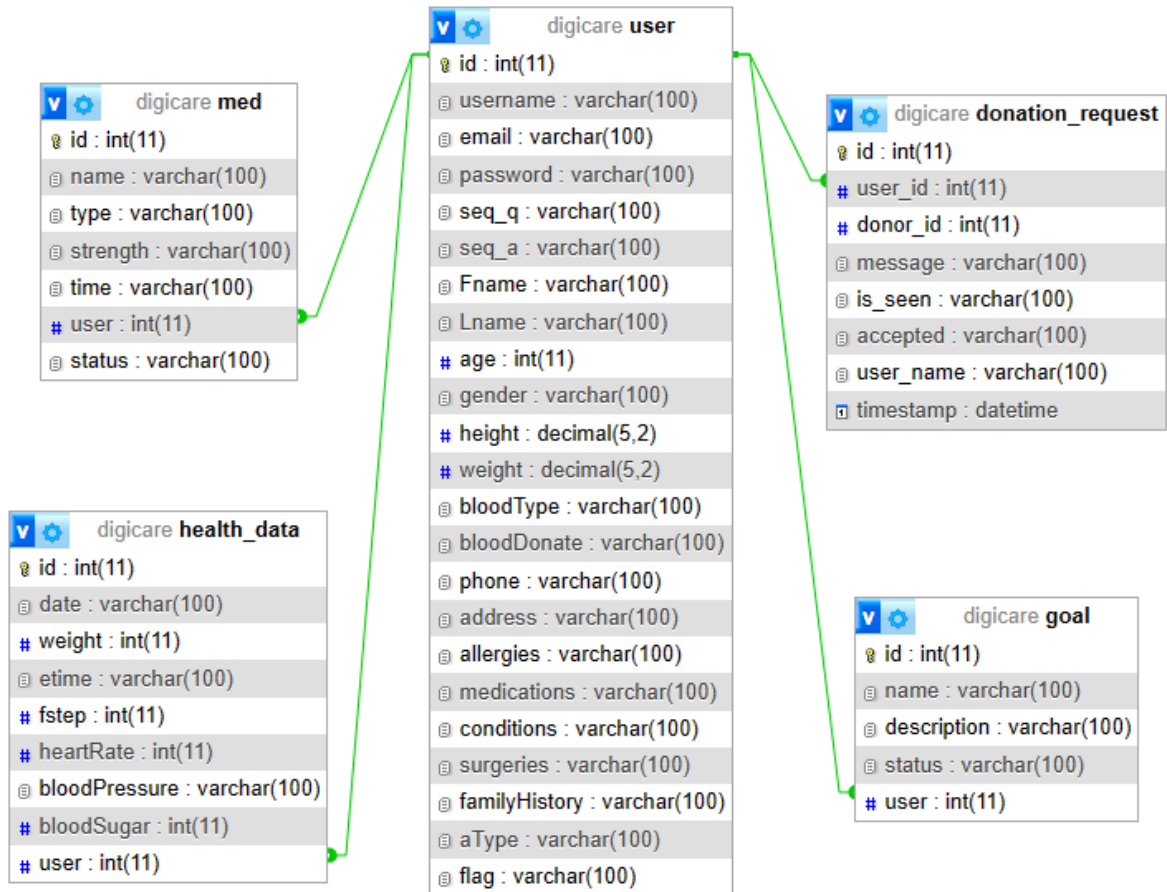
Figure 35: admin ban list

The frontend development of DigiCare represents a harmonious blend of HTML, CSS, and JavaScript, with additional styling contributions from Bootstrap for a polished and responsive design. The carefully crafted layout ensures a seamless and visually appealing user experience across various devices. Each page, using a shared base, is organized within the templates folder, enabling efficient routing that establishes a connection with the database, allowing the entire

project to function cohesively. The navigation bar elegantly guides users to different sections of the platform, fostering an intuitive journey through health tracking and assistance features. The strategic use of dropdown menus enhances user interaction. The hero section, complemented by call-to-action buttons, invites users to engage with the platform. The overall design, as evidenced by the attached screenshots, reflects a commitment to user-friendly aesthetics and navigability, encapsulating the essence of DigiCare's mission for comprehensive health management.

Database Schema

Digicare Database Flow Diagram *



Technology (Framework, Languages)

The project is built on a Flask backend, seamlessly integrated with HTML, CSS, and JavaScript for the frontend. This combination of technologies forms a robust platform that encompasses a wide range of health management features. The database functionality is supported by MySQL or phpMyAdmin, ensuring efficient data storage and retrieval. This technological foundation provides a solid framework for the development and implementation of diverse functionalities within the health management and assistance platform.

Conclusion

In summary, Digicare is a one-stop platform for managing health and getting assistance. It offers a range of user-friendly features like easy sign-up, secure log-in, and customizable profiles. With tools to track health data, set goals, and connect with potential blood donors, Digicare is all about making health management simple. Users can input and manage their health metrics, set goals, and track progress in real-time. The platform also includes handy calculators for BMI, BMR, and body fat. In the Blood Donation section, users can request blood donations and receive notifications, creating a supportive community. Digicare's admin features ensure a safe and ethical environment. Administrators can manage blood donation requests, accept, or reject them, and oversee user activities. The platform's goal is to revolutionize health monitoring with an intuitive interface and practical features. In a nutshell, Digicare is reshaping how individuals approach health management, emphasizing simplicity, connectivity, and innovation.