

Software Engineering Department  
ORT Braude College

Capstone Project Phase B – 61999

Project No.: **23-1-D-8**

**Life-Style Builder**

June – 2023

Authors:

Shahar Almog – 206217606

Yehonatan Ben Shitrit – 315687442

GIT:

[https://github.com/yonibsh145/Healthy-life-style-supported-by-specialists](https://github.com/yonibsh145/Healthy-life-style-supported-by-doctors)

[**1.Introduction** 5](#_Toc138527193)

[**2.Project review and process description** 6](#_Toc138527194)

[2.1 Accomplishments 6](#_Toc138527195)

[2.2 Architecture 6](#_Toc138527196)

[2.2.1 Program Management 6](#_Toc138527197)

[2.2.2 Program utilization 7](#_Toc138527198)

[2.2.3 Reviews 7](#_Toc138527199)

[2.2.4 Communication 8](#_Toc138527200)

[2.3 Usage 8](#_Toc138527201)

[2.3.1 JavaScript 8](#_Toc138527202)

[2.3.2 Agile-Development 8](#_Toc138527203)

[2.3.3 ReactJS 9](#_Toc138527204)

[2.3.4 Render 9](#_Toc138527205)

[2.3.5 React Router 9](#_Toc138527206)

[2.3.6 React Bootstrap 10](#_Toc138527207)

[2.3.7 Hooks 10](#_Toc138527208)

[2.3.8 NodeJS 10](#_Toc138527209)

[2.3.9 DOM 10](#_Toc138527210)

[2.3.10 Express.JS 10](#_Toc138527211)

[2.3.11 Axios 10](#_Toc138527212)

[2.3.12 MongoDB 11](#_Toc138527213)

[2.3.13 Material-Tailwind 11](#_Toc138527214)

[2.3.14 Client-Server 11](#_Toc138527215)

[2.3.15 JSX 11](#_Toc138527216)

[2.4 Challenges and solutions 12](#_Toc138527217)

[2.5 Testing/Evaluation 13](#_Toc138527218)

[2.5.1 Unit Testing: 13](#_Toc138527219)

[2.5.2 Functionality Testing: 15](#_Toc138527220)

[2.6 Results and conclusions 17](#_Toc138527221)

[3.User Documentation 18](#_Toc138527222)

[3.1 User's Guide 18](#_Toc138527223)

[3.1.1 General Description 18](#_Toc138527224)

[3.1.2 Application Flow 19](#_Toc138527225)

[3.2 Maintained Guide 30](#_Toc138527226)

[3.2.1 Use case diagram 30](#_Toc138527227)

[3.2.2 Class diagram 31](#_Toc138527228)

[3.2.3 Activity diagram 32](#_Toc138527229)

[3.3 Software and hardware infrastructures 35](#_Toc138527230)

[3.4 Database 36](#_Toc138527231)

[References 39](#_Toc138527232)

**Abstract**

The healthy lifestyle programs are very popular now. People want to lose weight, prevent from diabetes, or simply feel good. Such programs collect information about users, connect them with specialist and have many other activities. But most of the existing tools are mainly directed for patient usage. There are no robust tools directed for specialists or other healthy lifestyle specialists that convenient for building and manage of such programs. During the app development project, we employed a range of technologies to build a software solution for doctors and clients, with a primary focus on enhancing life expectancy. Leveraging React, CSS, server-side programming with Express, and MongoDB, along with additional libraries such as Axios and Material-Tailwind for inline CSS, we achieved significant milestones in implementation. One of the key realizations throughout the project was the importance of React's capabilities. By utilizing its component-based structure, we created a responsive and user-friendly interface. React's modular approach allowed for the seamless integration of various UI elements, resulting in a dynamic and intuitive user experience. Additionally, the implementation of server-side programming using Express proved vital in facilitating effective communication between doctors and clients. Through well-defined routes and endpoints, we enabled clients to submit progress reports, provide feedback, and directly communicate with their respective doctors. This implementation not only ensured smooth data flow but also prioritized user privacy and security. Furthermore, the use of MongoDB as the database management system provided efficient data storage and retrieval. By leveraging its NoSQL structure, we could easily store and manage client information, feedback, and scores. MongoDB's scalability and flexibility proved advantageous, accommodating the growing demands of the application while maintaining optimal performance. In terms of additional libraries, Axios played a crucial role in handling client-server communication. Its simplicity and flexibility made it an ideal choice for making HTTP requests and handling responses. Moreover, the utilization of templates like Material-Tailwind for inline CSS streamlined the styling process. This allowed for the consistent application of attractive and responsive designs across various components, enhancing the overall visual appeal and user experience. Overall, the project implementation highlighted the power of React and its component-based architecture, the significance of server-side programming with Express, the efficiency of MongoDB as a NoSQL database, and the contributions of libraries such as Axios and Material-Tailwind. This software solution successfully empowered doctors and clients, fostering effective communication, data management, and user engagement while focusing on improving individuals' life expectancy.

# **Introduction**

"Life-Style-Builder" is a web application designed to provide the best solution for individuals with medical conditions who are struggling with an unhealthy lifestyle. It connects them with a diverse range of specialists, including trainers, specialists, and nutritionists.

The application offers a convenient platform for patients who aspire to lead healthier lives. They can connect with any specialist of their choice without the need to be a patient of a specific healthcare provider. Similarly, specialists can connect with patients regardless of their healthcare coverage.

Specialists have various capabilities within the app. They can create new programs tailored to specific medical conditions, add daily activities to these programs, access detailed analytics about patients using their programs, send messages to patients based on their progress, accept or reject patient requests to join their programs, and even share the programs they create on social media. The app also provides libraries for different medical conditions, containing relevant programs created by specialists.

Patients have the option to request personalized programs from specialists based on their individual profiles. They can also request to join existing programs offered by specialists. Additionally, patients can update their progress by recording the activities they have completed within their programs. They could provide feedback on the difficulty or ease of specific activities and even leave reviews for the programs they have participated in.

Our solution aims to simplify the process of achieving healthy lifestyle goals and promotes consistency in maintaining a healthy way of living. It enables specialists, trainers, and nutritionists to reach a wider patient base beyond specific healthcare networks, while empowering individuals to take control of their health and well-being.

# **Project review and process description**

## 2.1 Accomplishments

The lifestyle-builder application is project aimed at empowering individuals to adopt healthy and improved lifestyles. This comprehensive application serves as a platform where specialists from various fields can create personalized programs to enhance users' well-being and quality of life. With the ability to cater to individual needs and goals, the application provides users with a range of tools and resources to guide them on their journey towards a healthier lifestyle. The accomplishment of this project lies in successfully developing and implementing an intuitive and user-friendly application that seamlessly connects specialists and users. The lifestyle-builder application encompasses a well-structured framework with distinct roles for users, including unregistered users, registered users, and specialists. To ensure a seamless user experience, the project has been meticulously divided into several independent sections, each serving a specific purpose. This approach allows for efficient management and implementation of the various components of the application.

## 2.2 Architecture

2.2.1 Program Management  
Program Management plays a pivotal role within the lifestyle-builder application, encompassing a significant portion of the system's functionality. This essential component empowers specialists to create diverse and tailored programs for users, offering a range of options to suit individual needs and preferences. Within this process, specialists have the ability to define various attributes for each program, including its name, duration, description, and daily actions. When crafting a program, specialists carefully curate a series of actions that users are required to perform on a daily basis. While there is no strict limit on the number of actions per program, specialists are mindful not to overload or underestimate the tasks, as the primary aim is to provide users with optimal quality and achievable objectives. The flexibility of the program management system allows specialists to make edits to any program at any time. This ensures that programs can be modified and refined to align with evolving user requirements or to incorporate new insights and recommendations from the specialist. Furthermore, specialists are also granted the capability to remove any program from the system should it no longer align with their expertise or the overall objectives of the application. As a means of expanding the reach and impact of their programs, specialists are provided with the option to share their plans on social networks. This functionality enables them to disseminate their expertly crafted programs to a wider audience, promoting healthy and improved lifestyles among users beyond the application's immediate community.

2.2.2 Program utilization  
Program utilization is a central component of the application, serving as its primary focus. Registered users have the ability to explore and select programs of their choice through the application's homepage, establishing a seamless user experience. Each program is accompanied by comprehensive details, allowing users to evaluate its suitability before committing to its implementation. Additionally, users can access ratings and reviews provided by fellow users, providing valuable insights into the effectiveness and quality of programs available within the system. To avail themselves of a program, users must submit a request to a specialist for approval. As the specialist evaluates the user's specific needs and goals in relation to the program. This meticulous approval process adds an extra layer of assurance and expert oversight, fostering a tailored experience for each user. Once a program is approved for use, the user is presented with a breakdown of daily tasks that need to be performed. The user is required to confirm completion of these tasks, ensuring active engagement and accountability throughout the program duration. Detailed information and display of the daily tasks are prominently showcased on the user's main page upon logging into the application. Additionally, an activity page provides a comprehensive overview of the user's active programs, facilitating efficient tracking and management.   
Upon completion or during the program, users have the option to provide a review along with a corresponding score, allowing them to share their experiences and insights. This feature not only encourages user engagement but also enables the continuous improvement and refinement of programs based on user feedback.

2.2.3 Reviews  
Reviews play a pivotal role within the application, serving as a crucial aspect of user feedback and program evaluation. Users are provided with the opportunity to give reviews and ratings, enabling them to assess the effectiveness and quality of the programs they have utilized. This feedback mechanism empowers users to express their opinions and experiences, contributing to an informed decision-making process for other users considering program adoption. Furthermore, the reviews and ratings attributed to a specialist serve as a valuable metric for evaluating the specialist's overall performance and expertise. By aggregating the reviews and calculating an average rating, the application generates an assessment of the specialist's proficiency and effectiveness in designing and delivering programs. This rating system aids users in assessing the credibility and competency of specialists, enabling them to make informed choices when selecting programs. Conversely, the specialist also benefits from the reviews provided by users. They have access to comprehensive feedback on all the programs they have designed and prepared within the system. By reviewing user feedback, specialists gain valuable insights into the efficacy of their programs, allowing them to refine and improve their offerings. This feedback loop facilitates ongoing professional development and ensures that specialists can continuously enhance the quality and impact of their programs.

2.2.4 Communication  
Effective communication between users and specialists is a fundamental aspect of the system, ensuring a seamless and personalized experience. Within the application, users have the ability to initiate direct communication with specialists by sending messages through their respective profiles. This feature enables users to engage in meaningful conversations and seek clarification or guidance regarding their plans. The system incorporates a dedicated Inbox page where users can conveniently track and manage their message exchanges with specialists. This centralized location allows users to access a comprehensive history of their communication, ensuring easy reference and efficient follow-up on discussions with specialists.

## 2.3 Usage

In this section, we will provide a preview of the implementation process for our project, highlighting the key technologies and frameworks used. Our project involves building a web application using React, CSS, and MongoDB, along with other complementary tools. Let's explore how each of these components contributes to the overall development and functionality of our application.

### 2.3.1 JavaScript \*

JavaScript is a high-level, interpreted programming language primarily used for web development. It is a versatile language that runs on the client-side as well as the server-side. JavaScript enables dynamic and interactive functionality on web pages, allowing developers to create interactive forms, handle events, manipulate the DOM, and make asynchronous requests to servers.

With JavaScript, developers can build a wide range of web applications, from simple scripts to complex single-page applications. It supports object-oriented, functional, and procedural programming paradigms. JavaScript's syntax is similar to other C-style languages, making it relatively easy to learn for those familiar with programming concepts.

2.3.2 Agile-Development \*  
For the development stage, we had to choose the most suitable software development methodology for our project, we have researched and analyzed a couple of project development methodologies such as Agile and Waterfall. our conclusion is that Agile is the most appropriate development methodology for our project.

Agile methodology is an iterative approach to project management and software development, which breaks down projects into small pieces. These project pieces are completed in work sessions that are often called sprints. Sprints generally run anywhere from a few days to a few weeks. These sessions run from the initial design phase to testing and quality assurance (QA).

The main benefit of Agile development methodology lies on the flexible approach to product development, the development team are responsive to changes, even at the last minute, and can adapt to it without much disruption.

Hence, the use of the Agile methodology will allow us to release an initial version of the system.

2.3.3 ReactJS \*  
ReactJS is an open-source JavaScript library used for building user interfaces (UIs) in web applications. It was developed by Facebook and released in 2013. React allows developers to create reusable UI components and efficiently update and render them based on changes in the application's data or state.

The main idea behind React is to build UIs as a composition of small, isolated components. Each component represents a piece of the user interface and encapsulates its own logic and state. These components can be combined to create more complex UI structures.

React utilizes a virtual DOM to efficiently update the user interface. The virtual DOM is an abstraction of the actual browser DOM, which represents the structure of an HTML document. When the state of a React component changes, react calculates the difference between the previous and current state and updates only the necessary parts of the virtual DOM. This approach improves performance by minimizing the number of actual DOM manipulations.

React also follows a unidirectional data flow, where data flows from parent components to child components through properties, or props, and any changes to the data are handled through callback functions. This helps maintain a predictable and easily understandable application state.

React is often used in combination with other libraries or frameworks to build complete web applications. It is commonly used with a routing library for managing different application views, and with a state management library (e.g., Redux) for managing complex application state.

2.3.4 Render \*Render is the process of generating and displaying the final output of a user interface based on the data and logic of the application, updating only the necessary parts for better performance.

2.3.5 React Router **\***React Router is allowing developers to manage different views and navigation in a declarative manner. With React Router, you can create routes, handle URL parameters, and dynamically render components based on the current URL.

2.3.6 React Bootstrap \*React Bootstrap is a library that combines the power of React and Bootstrap, providing pre-built UI components that are ready to use in React applications. It simplifies the process of building responsive and visually appealing interfaces by offering a wide range of customizable and reusable components.

2.3.7 Hooks **\***Hooks are a feature that allow developers to use state and other React features in functional components. They provide a way to manage component state, handle side effects, and reuse logic. Hooks enable writing cleaner and more modular code by eliminating the need for class components in many cases.

2.3.8 NodeJS \*  
Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to build server-side and networking applications. It provides an event-driven architecture and non-blocking I/O model, which makes it efficient and scalable for handling concurrent requests. Node.js utilizes the V8 JavaScript engine and offers a rich set of modules and libraries, enabling developers to create fast and scalable web applications and APIs using JavaScript on both the server and client sides.

2.3.9 DOM \*  
The Document Object Model is a programming interface for web documents. It represents the structure of an HTML or XML document as a tree-like structure, where each element can be accessed, manipulated, and modified dynamically using JavaScript, enabling interactive web page manipulation.

2.3.10 Express.JS \*  
Express.js is a minimalist web application framework for Node.js. It provides a set of features and tools that simplify the process of building web applications and APIs. With its intuitive routing system, middleware support, and modular design, Express.js enables developers to create scalable and robust server-side applications efficiently.

2.3.11 Axios \*  
Axios is a JavaScript library used for making HTTP requests from web browsers or Node.js. It provides a simple and intuitive API for performing asynchronous HTTP communication, handling response data, error handling, and more. Axios supports various request methods and offers features like request cancellation and interceptors.

2.3.12 MongoDB \*  
MongoDB is an open-source NoSQL database system. It is designed for storing, retrieving, and managing large volumes of structured and unstructured data. MongoDB uses a flexible document-oriented data model, where data is stored in JSON-like documents, allowing for dynamic and schema-less data structures. It provides high scalability and performance, with support for horizontal scaling across multiple servers. MongoDB also offers powerful querying capabilities, indexing, and aggregation framework for data analysis. Its flexible data model, scalability, and ease of use make it a popular choice for a wide range of applications, including web applications, content management systems, and real-time analytics.

2.3.13 Material-Tailwind \*  
Material-tailwind is a UI component library that combines the design principles of Material Design with the utility classes of Tailwind CSS. It provides a set of pre-styled and customizable UI components, such as cards, buttons, typography, and more, enabling developers to create visually appealing and responsive user interfaces quickly.

2.3.14 Client-Server \*  
Client-server denotes a relationship between cooperating programs in an application, composed of clients initiating requests for services and servers providing that function or service.

Clients, also known as service requesters, are pieces of computer hardware or server software that request resources and services made available by a server.

A server is a device or computer program that provides functionality for other devices or programs. Any computerized process that can be used or called upon by a client to share resources and distribute work is a server.

2.3.15 JSX \*  
JSX (JavaScript XML) is a syntax extension used in React to write HTML-like code within JavaScript. It allows developers to define the structure and appearance of React components. JSX code is transpiled into regular JavaScript before being executed in the browser, enabling the creation of dynamic and interactive user interfaces.

2.4 Challenges and solutions  
During the development of the project, we have faced several challenges,  
Firstly, A challenge arose when implementing the feature to share programs on social networks within the system. Each page within the application required specific information from the database and was rendered accordingly, with a unique name assigned to each page. However, to share a program on a social network, it was necessary to identify the specific program linked to the program's URL. To address this challenge, we utilized the React-Share library. Through this library, we created a distinct link for each program by incorporating its corresponding ID. This allowed us to establish a connection between the program and its data in the database. By clicking on the share link, the system intelligently retrieved the program's data based on its unique ID, ensuring accurate information retrieval for sharing purposes.

Secondly, during the development of our application, we encountered a challenge with the database design related to the management of patient programs and their statuses. Initially, each patient had a list of programs, and each program had its own status. However, as we progressed, we realized that it was becoming complex to handle program statuses within the program itself, to address this issue and facilitate the required functionality of displaying active programs on the patient's home page, we decided to modify the database structure. The new design introduces a more complex structure where each patient has an array containing program-status pairs, with this change, it becomes easier to retrieve a program and check its status, whether it is active or not. The modified structure allows for more flexibility in managing program statuses for each patient, providing a clearer representation of the relationship between programs and their statuses, by separating the program and status into distinct fields within the patient's array, we can easily access and update the status of each program as needed. This enhanced design simplifies the process of retrieving and displaying active programs for the patient's home page, improving the overall user experience.

Thirdly, during the development process, we encountered difficulties in implementing the functionality to display a patient's daily activities for the current day within each program. Initially, we attempted to save each activity with a specific date, but this approach proved to be complex and challenging to ensure that the patient sees their activities for the current date, subsequently, we explored an alternative solution. We introduced the concept of "daily activities" within each program, where each activity is associated with a specific day. Additionally, we modified the program structure to include two main parts: the day and a list of activities. The day field represents the duration of the program, while the list of activities corresponds to the activities scheduled for each specific day within the program, to implement this solution effectively, we created a pre-middleware function that runs before saving a program in the database. This function generates the daily activities list based on the duration of the program and populates it with the corresponding activities for each day. By doing so, we ensure that when a patient logs into the system, they will see their daily activities for the current day based on their active programs, this revised approach simplifies the process of retrieving and displaying the patient's daily activities, as the activities are now organized within the program structure itself. Patients can easily access their daily activities specific to the current day, providing a more intuitive and user-friendly experience within the system.

## 2.5 Testing/Evaluation

To be able to evaluate the performance of our system, we performed comprehensive tests that include system unit tests and functional tests through which we will verify the integrity of the system, and that it indeed performs and meets the established requirements.  
Based on the tests that we required at the "Phase A" with a bit of changes:

### 2.5.1 Unit Testing:

|  |  |  |  |
| --- | --- | --- | --- |
| **Registration Testing** | | | |
| **No.** | **The Test** | **Expected result** | **Result** |
| 1 | Enter the empty required field | Error alert: "Please fill in all the required fields" | **Passed** |
| 2 | Wrong password pattern | Error alert: "Please retype a password that matches the password pattern" | **Passed** |
| 3 | Password verification does not match | Error alert: "The passwords do not matches please retry again" | **Passed** |
| 4 | Wrong email pattern | Error alert: "Please retype the email that matches the email pattern" | **Passed** |
| 5 | Used email | Error alert: "The email you entered is already in use" | **Passed** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Login Testing** | | | |
| **No.** | **The Test** | **Expected result** | **Result** |
| 1 | Wrong email pattern | Error alert: "Please retype the email that matches the email pattern" | **Passed** |
| 2 | Empty required field | Error alert: "Please fill all the fields to login" | **Passed** |
| 3 | The email and password don't match | Error alert: "The email and the password don't match please try again" | **Passed** |
| 4 | Incorrect matched email and password | Error alert: "Invalid email or password" | **Passed** |

|  |  |  |  |
| --- | --- | --- | --- |
| **New Program Testing** | | | |
| **No.** | **The Test** | **Expected result** | **Result** |
| 1 | Empty program | Error Alert: "A program has to have at least one objective" | **Passed** |
| 2 | Empty program name | Error Alert: "The program must have a name" | **Passed** |
| 3 | Unsuitable day setup | Error alert: "Training Length must be larger than Training Day" | **Passed** |
| 4 | Program times as integers | Error alert: "Training Length and Days must be a number" | **Passed** |
| 5 | Existing action name | Error alert: "A training with the same name already exists." | **Passed** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Review Testing** | | | |
| **No.** | **The Test** | **Expected result** | **Result** |
| 1 | Empty rating | Error alert: "Review must have a rate" | **Passed** |
| 2 | Empty review | Error alert: "The review must have at least one word" | **Passed** |
| 3 | Multiply reviews | Error alert: "Program already reviewed" | **Passed** |

### 2.5.2 Functionality Testing:

|  |  |  |  |
| --- | --- | --- | --- |
| **UI Testing** | | | |
| **No.** | **The Test** | **Expected result** | **Result** |
| 1. | Buttons check | Every button in the application is working properly | **Passed** |
| 2 | Navigation check | With every redirect, the application is working as intended | **Passed** |
| 3 | Database check | Showing the correct content in the application as the database | **Passed** |
| 4 | Page loading | Make sure that every page time loading is optimistically | **Passed** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Program Testing** | | | |
| **No.** | **The Test** | **Expected result** | **Result** |
| 1. | Adding objectives | Adding any objective to the program is shown on the screen | **Passed** |
| 2 | Delete objectives | Removing any objective from the program is shown on the screen | **Passed** |
| 3 | Saving program | The program should be shown in program library of the specialist | **Passed** |
| 4 | Edit objectives | editing any objective from the program is shown on the screen | **Passed** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Share Testing** | | | |
| **No.** | **The Test** | **Expected result** | **Result** |
| 1. | Platforms validate | The sharing is working in every platform connected to the application | **Passed** |
| 2 | Link validate | The reference link to the program is valid through the shared link | **Passed** |
| 3 | Share connection | The connection between the social media the user profile is working properly | **Passed** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Chat Testing** | | | |
| **No.** | **The Test** | **Expected result** | **Result** |
| 1. | Sending message | The message sent from one user to another user needed to be presented on both chat screen | **Passed** |
| 2 | Open new chat | The correct users that chatting is shown | **Passed** |
| 3 | Close chat | The chat is closed, and the screen switched to the last page | **Passed** |
| 4 | Delete history | Removing all the chat history between the users | **Passed** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Rendering Testing** | | | |
| **No.** | **The Test** | **Expected result** | **Result** |
| 1 | User loading | Every page is refereeing to the specific user type | **Passed** |
| 2 | Data load | The data for each page is shown properly | **Passed** |

# User Documentation

## 3.1 User's Guide

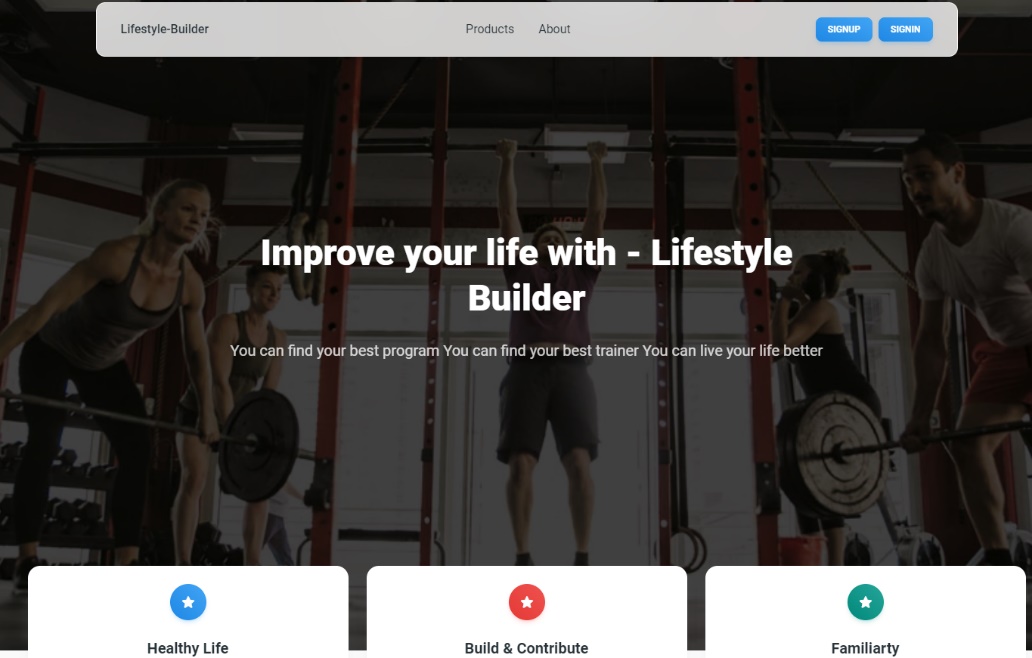
### 3.1.1 General Description

Lifestyle Builder, an application designed to help specialists and specialists create and share programs for managing a healthy lifestyle. In this guide, we will provide an overview of the features and functionalities offered by Lifestyle Builder, empowering users to improve their well-being based on their specific roles and access levels. Lifestyle Builder caters to three distinct user types: unregistered users, regular users, and specialists.

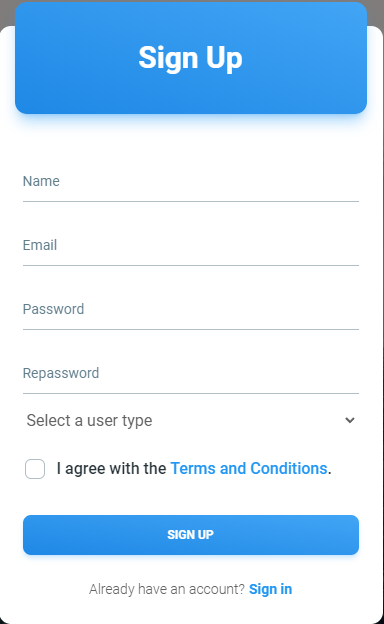
a.) Unregistered users have the opportunity to explore the application and gain insights into its features and offerings. While they cannot actively engage with the programs or access certain functionalities, they can view the available programs, read descriptions, and understand the benefits of participating in Lifestyle Builder. Unregistered users are encouraged to create an account to unlock the full potential of the platform.  
b.) Regular users, upon creating an account, gain access to a wide range of features within Lifestyle Builder. They can browse and select programs that align with their health goals, preferences, and lifestyle. Regular users have the ability to actively participate in the programs, track their progress, log their activities, and benefit from the guidance and expertise provided by specialists. Additionally, they can watch and review programs, sharing their experiences and feedback with the community.  
c.) Specialists, including specialists and experts in various fields, play a vital role in the Lifestyle Builder ecosystem. They have the ability to create and share programs tailored to specific health concerns, such as nutrition, fitness, stress management, and more. Specialists can leverage the platform's intuitive interface to design comprehensive programs, set goals, and provide valuable insights. They can also engage with regular users through the chat feature, offering personalized guidance and support, further enhancing the user experience.

### 3.1.2 Application Flow

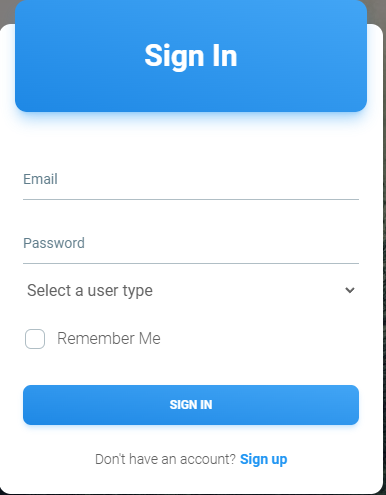
**Home Page:**  
landing page for unregistered/unlogged users. Here the user can review the app's capabilities Get to know the system, examine the programs, profiles and reviews of the programs that are at his disposal. In order to perform an active action, the user will have to register/log in to the system.



**Register Page:**Registration page, the user must fill in all the fields in accordance with the system requirements. In addition, he must choose which type of user he wants to be - a normal user or a specialist.

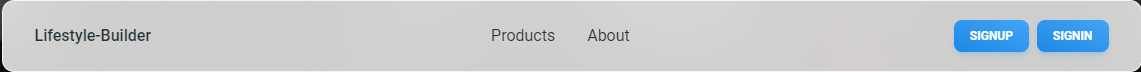


**Login Page:**On the system's login page, users are required to enter the email address and password associated with their system registration. In the event that a user does not possess a system account, they may proceed to the registration stage. During the login process, users are also prompted to specify their user type based on their registration details.

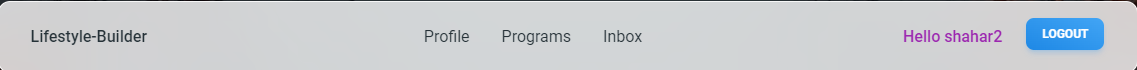


**Navbar:**  
The system offers three distinct navigation bar configurations, each tailored to specific user groups and their corresponding functionality. These configurations cater to the diverse needs of different user categories, ensuring an optimal user experience.

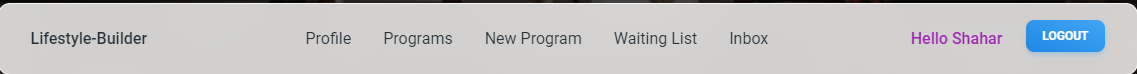
The first configuration is designed for non-registered users, allowing them to navigate through the system's features and content seamlessly. It provides a user-friendly interface, facilitating easy access to relevant information and functionalities while prioritizing a simple and intuitive user journey.



The second configuration is specifically tailored for registered users, enabling them to leverage additional features and personalized options. This configuration grants registered users' access to their individual profiles, preferences, and exclusive functionalities, enhancing their overall interaction with the system.



The third configuration is exclusively crafted for specialists, catering to their advanced requirements and specialized tasks within the system. It offers a comprehensive set of tools and features that cater to the specialist user's domain-specific needs, facilitating efficient workflow management and enabling them to leverage their expertise effectively.



**User Homepage:**The user's home page, where you can access a comprehensive overview of all the programs available within the application. Each program offers a range of interactive actions that you can initiate according to your preferences and requirements.

By clicking on the "Watch" option, you will be seamlessly directed to view the content of the selected program. This feature enables you to explore the program's multimedia materials, such as videos, presentations, or other relevant resources.

If you wish to utilize a program and benefit from its functionalities, you can click on the "Use" option. This action will trigger a request to the designated specialist, notifying them of your intention to engage with the program. The specialist will then review your request and provide the necessary permissions for you to access and utilize the program's features.

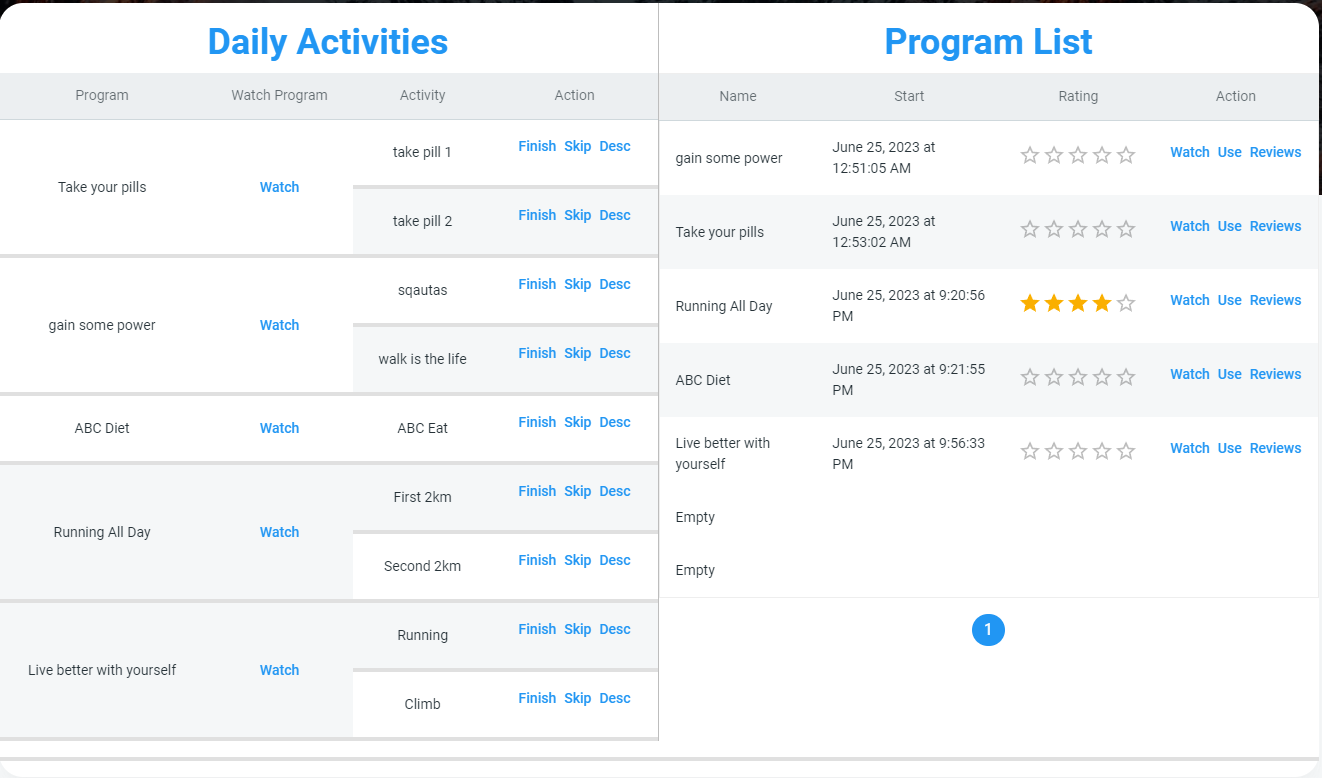
Selecting the "Review" option will grant you access to a comprehensive collection of user-generated reviews for the particular program of interest. This functionality enables you to gain insights and perspectives from other users who have engaged with the program, aiding you in making informed decisions and assessing its suitability for your needs.

In addition to program exploration, the user's home page also displays active programs that you have engaged with. Within each active program, you will find a list of pending actions that are yet to be completed on the current day. Once you perform an action, you can simply click on the "Done" button to notify the system that the respective activity has been successfully executed for the day, allowing for accurate tracking of your progress, and ensuring a streamlined user experience.

Furthermore, the user's home page will display a personalized list of daily activities based on their active programs. This feature aims to provide a convenient overview of the tasks and actions that need to be completed by the user each day.

Upon accessing the home page, the user will be presented with a section dedicated to their active programs. Within each active program, there will be a list of daily activities that are scheduled for the current day. These activities represent the tasks and goals outlined in the program designed to support the user's healthy lifestyle.

To indicate the completion of an activity, the user can simply click on the "finish" button associated with each task. This action will notify the system that the respective activity has been successfully executed for the day. By marking activities as "finish," the user can accurately track their progress, maintain consistency in their healthy lifestyle journey, and ensure a streamlined user experience. Upon clicking on the "Desc" button, a description of the activity will show up.



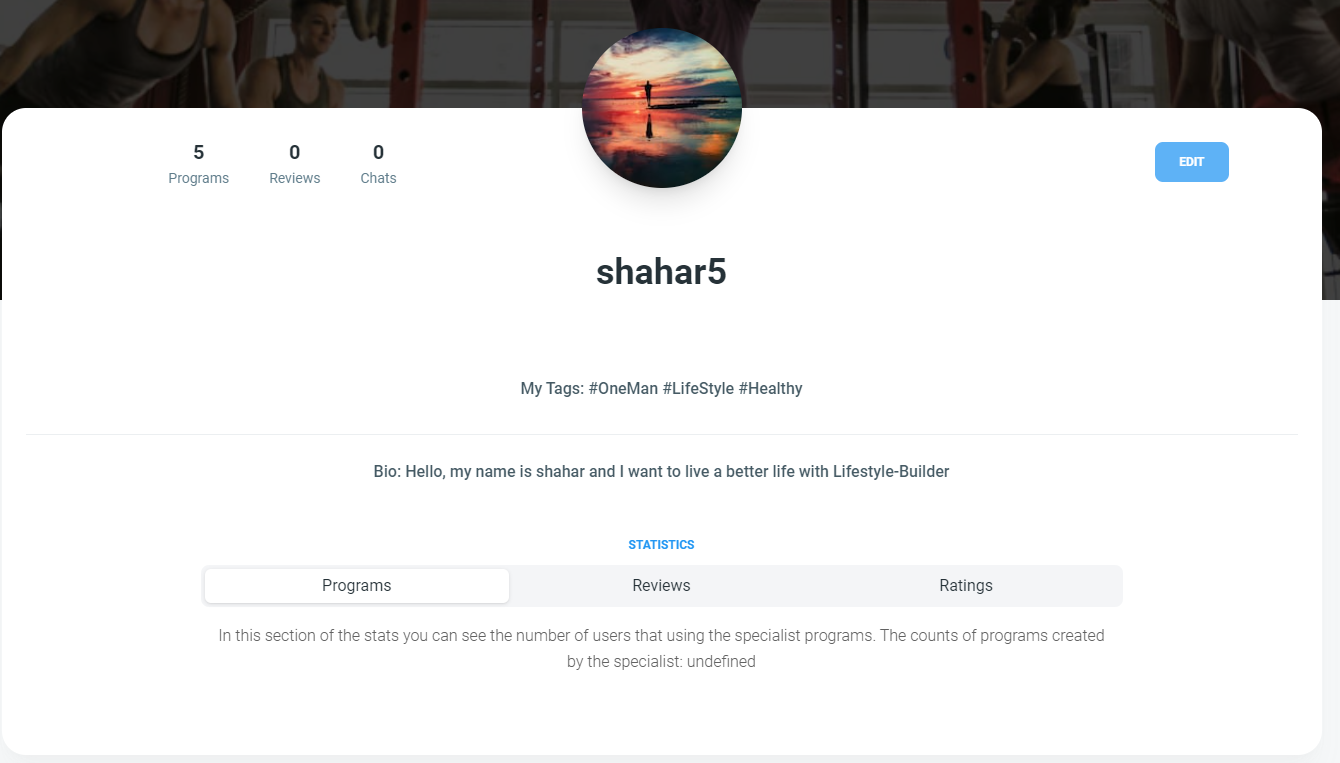
**User Profile:**In the user's personal profile, a comprehensive overview of their information and details is readily accessible. This profile section serves as a central hub for users to manage and customize their personal data in line with their evolving needs.

The profile encompasses various aspects, including basic information such as name, username, email address, and contact details. Additionally, users can provide specific details related to their medical history, height, weight, and other pertinent information that can contribute to a more tailored and effective health management experience.

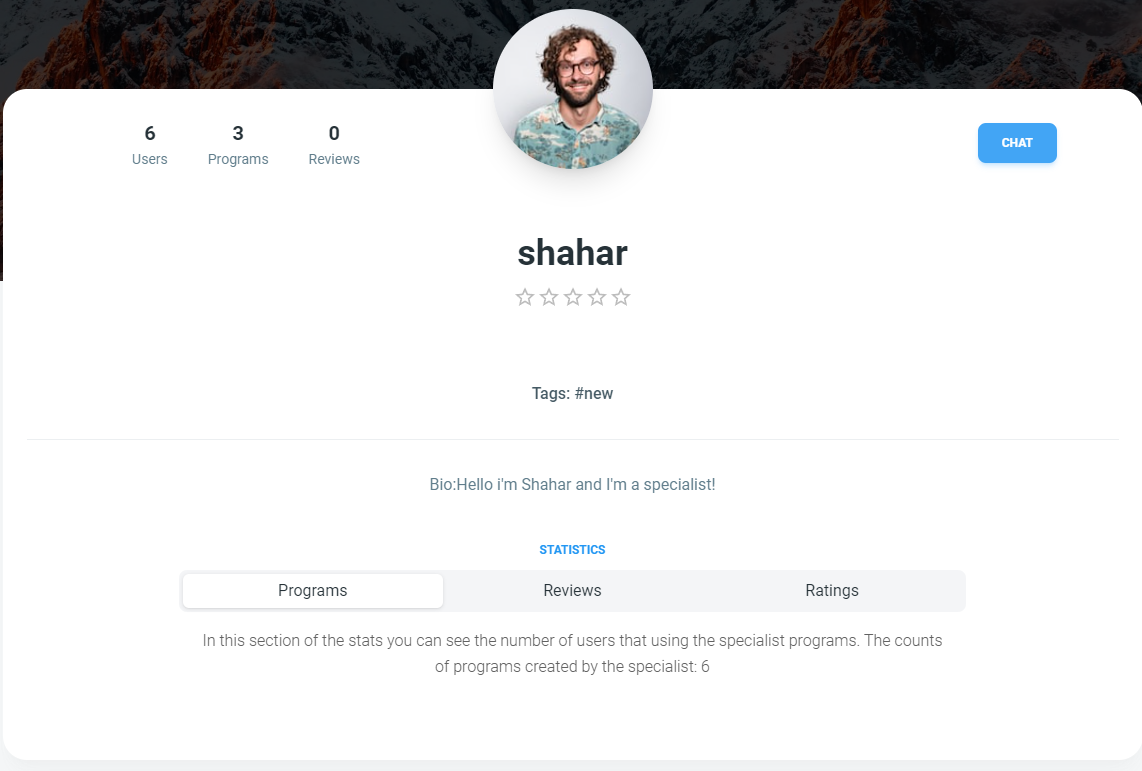
The profile page also showcases valuable statistics that offer insights into the user's engagement within the application. Users can view the number of programs they have utilized, providing a quantitative measure of their participation and commitment to improving their lifestyle. Furthermore, users can observe the number of reviews they have contributed, highlighting their active involvement in sharing feedback and experiences with programs and specialists.

To ensure the accuracy and relevance of their profile information, users have the privilege of utilizing the "edit" button. By selecting this option, users can effortlessly modify their personal data, update medical history, adjust height, and weight details, and make any necessary revisions to their profile information. This feature empowers users to maintain accurate and up-to-date records that align with their evolving health journey.

The user's personal profile serves as a comprehensive repository of their information, enabling them to track their engagement, tailor their health management approach, and exercise control over their profile data to ensure it accurately reflects their unique circumstances and requirements.



**User Watch Profile:**On the profile viewing page, the user can look at the profile of a user/specialist. Also, on this page you can see details about the user. By clicking on the "Chat" button it will be possible to send a message to the user/specialist.



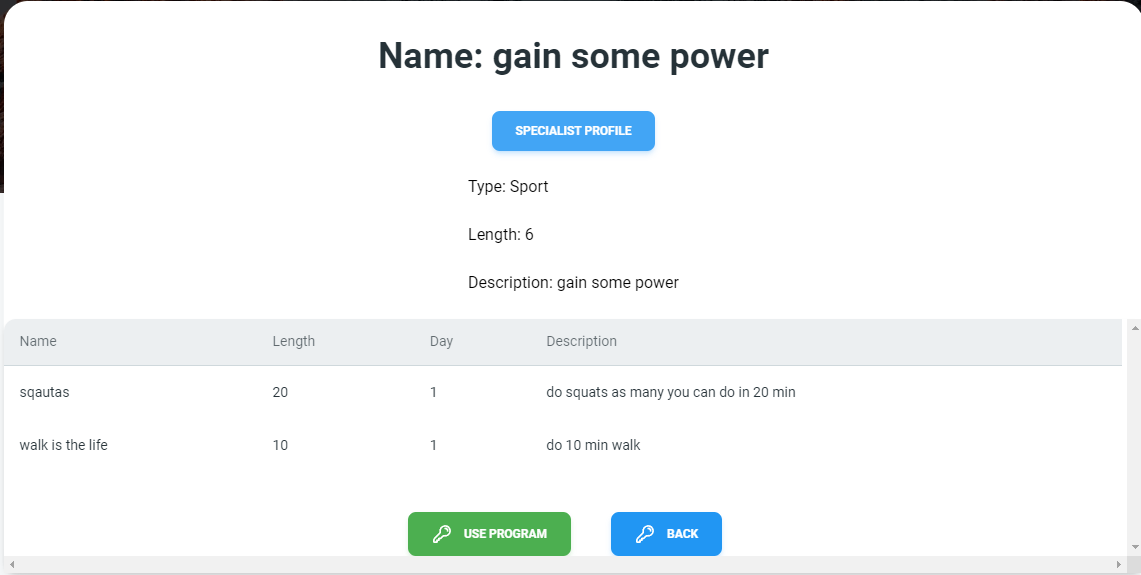
**User program functions:**On the program viewing page, users are provided with a comprehensive display of the program's content and materials. By accessing this page, users can explore and engage with the program's multimedia resources, which may include videos, presentations, documents, and other relevant materials. This allows users to gain valuable insights and knowledge related to the program's focus area.

To foster a deeper connection with the specialist who created the program, users can click on the "Specialist Profile" button. This action seamlessly redirects users to the profile page of the specialist, where they can learn more about the specialist's background, expertise, and other relevant information. This feature enhances the user's understanding of the program's creator and facilitates a sense of trust and confidence in the program's content.

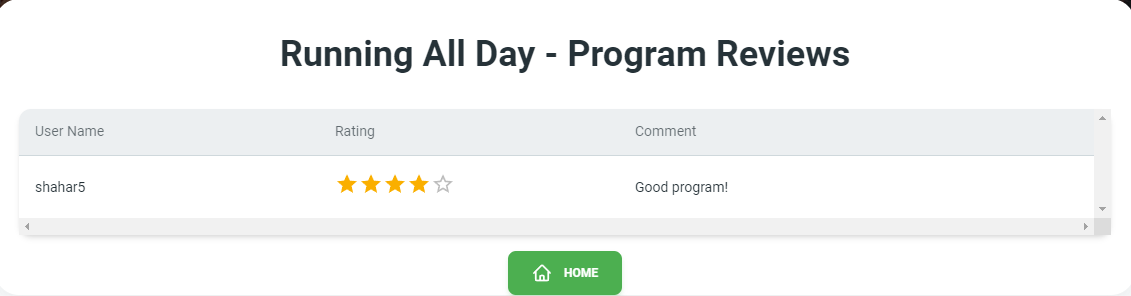
To navigate back to the home page and explore other programs and features, users can simply click on the "Back" button. This intuitive functionality ensures a seamless user experience, allowing users to effortlessly switch between different program views and access the wider range of available programs within the application.

When users are interested in utilizing a specific program and wish to benefit from its functionalities, they can click on the "Use Program" button. By doing so, users initiate a request to the specialist, expressing their intention to engage with the desired program. However, before the user can start using the program, the specialist must review and approve the request. This approval process ensures that users receive the necessary permissions and guidance from the specialist, creating a personalized and supervised program experience tailored to the user's needs and goals.

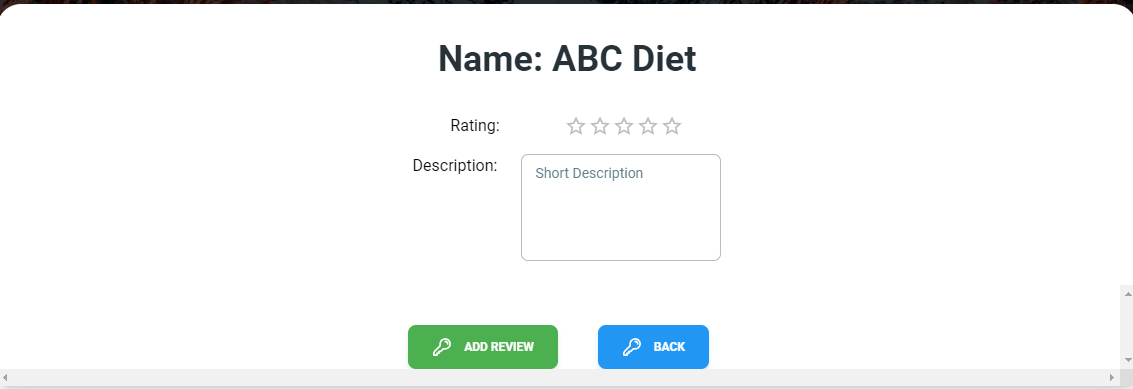
By incorporating these user-friendly features and functionalities into the program viewing page, the application aims to provide a seamless and engaging user experience, facilitating program exploration, specialist interaction, and efficient program request and approval processes.



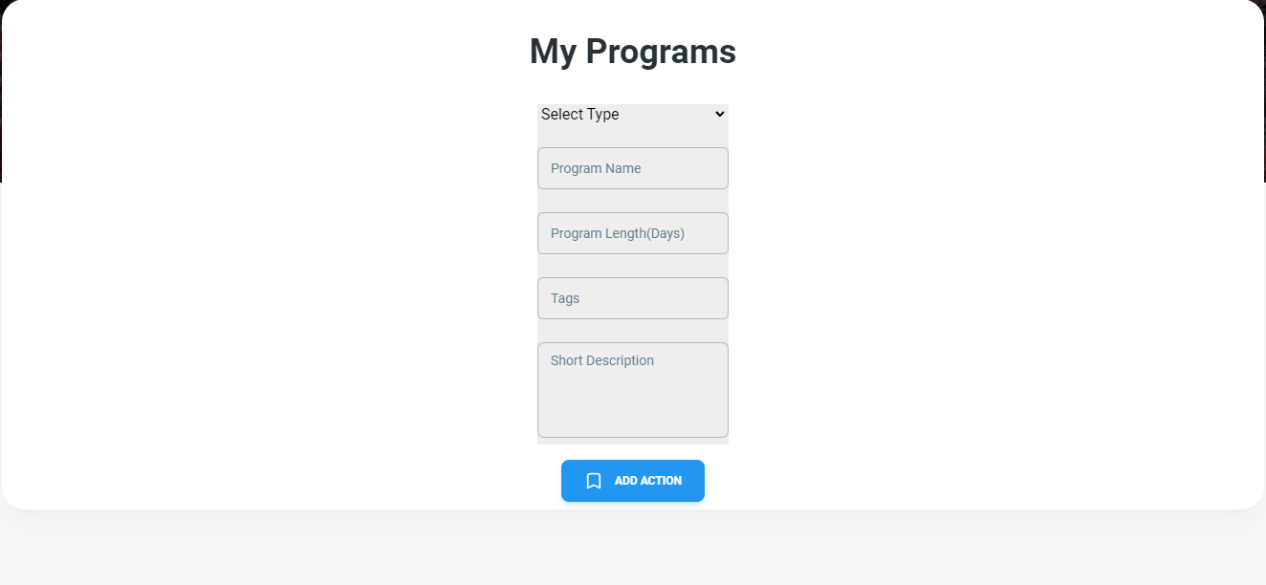
On the review viewing page, the user will see a list of reviews from other users. He will see the content of the review and grade and thus be able to get an impression of the program's software and its maker. From this page the user can return to the home page.



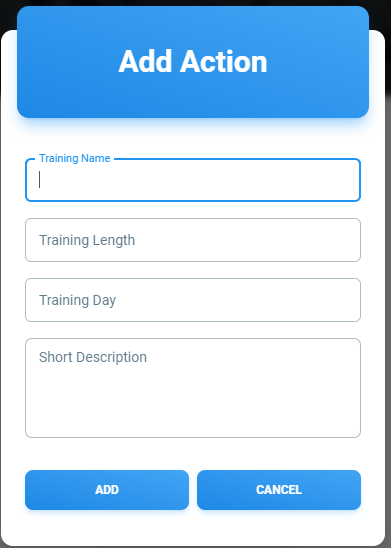
The review writing page is a page where the user can write a review for the program he is using/finished. The user must enter the rating of the feature and a comment. Clicking the "Back" button will cancel writing the review Clicking the "Add Review" button will add a review to the program



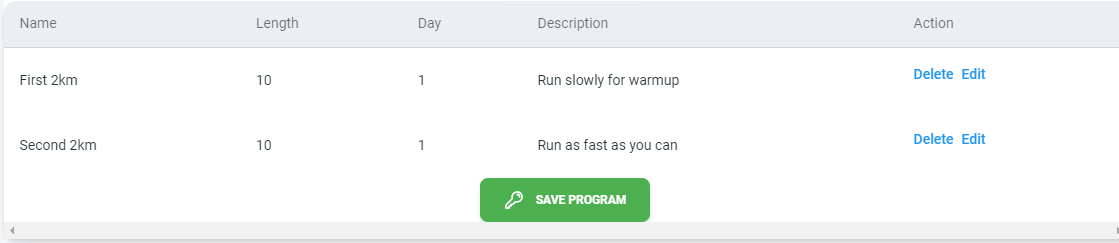
**Specialist Create program:**This is a page for writing a new program for a specialist. The specialist must fill in all the required fields, the length of the program must be a number.



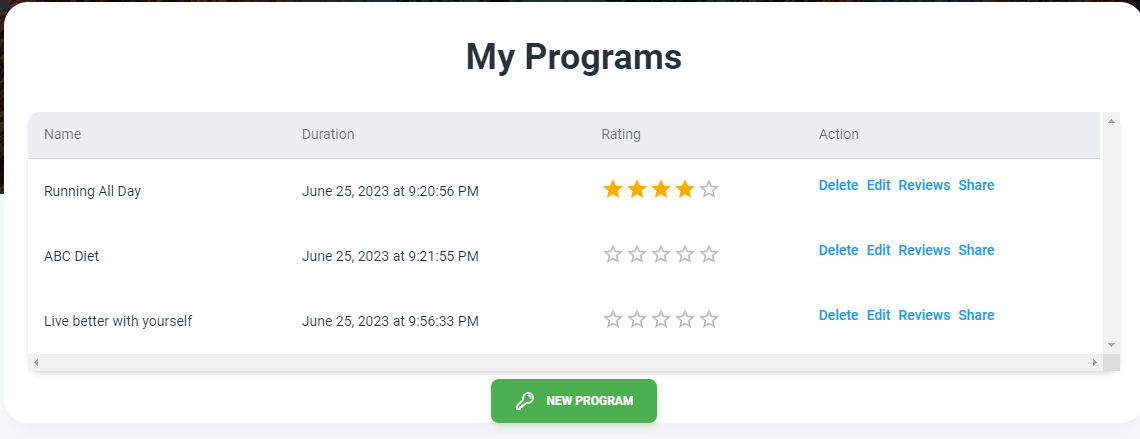
When clicking the "Add Action" button, content will be added to the program that needs to be executed. The specialist must fill in all the fields. The longitude and day fields must be a numeric value After finishing by pressing the "Add" button, the action was added to the program. The "Cancel" button will cancel the addition.



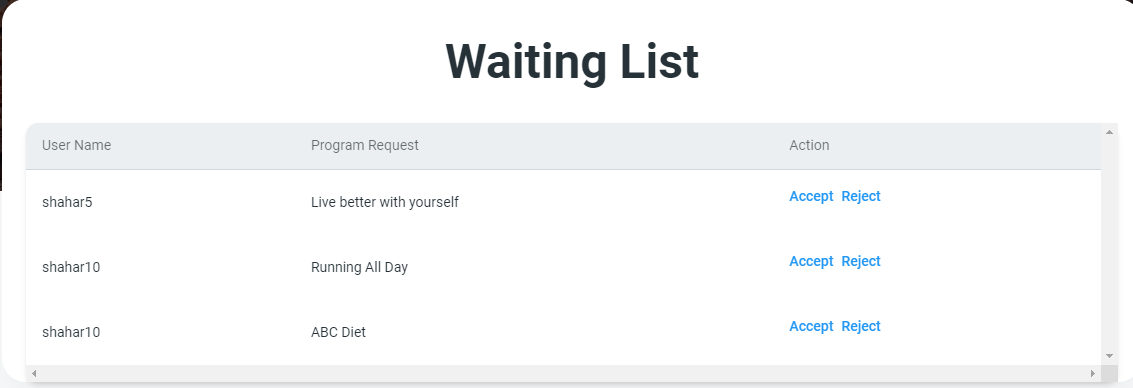
After adding an action, the specialist will see the response on the screen. A table with the details of the content he added and for which a deletion and editing operation can be performed. After he has finished adding all the contents, he wants by clicking again on the "Add Action" button. By pressing the "Save Program" button, the program will be saved in the system, and will be added to the specialist's list of programs



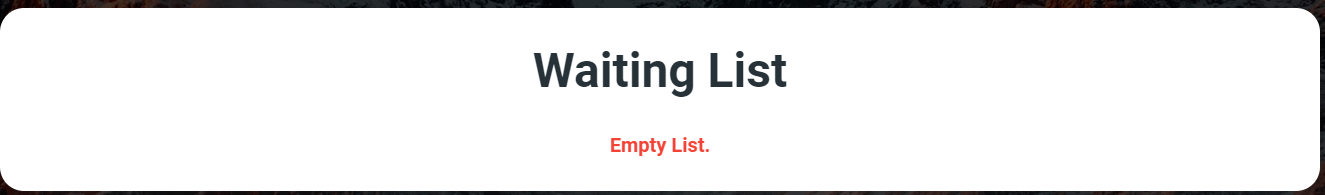
**Specialist Program:**On the program list page, the specialist will see all the programs in his possession. In the table he will see the details of the programs and for each program he can perform several actions.  
 Delete - will delete the program from the system.  
Edit - will go to the edit page of the existing program   
Reviews - will show the reviews of the specific program   
Sharing - the specialist will be able to share his program on social networks.



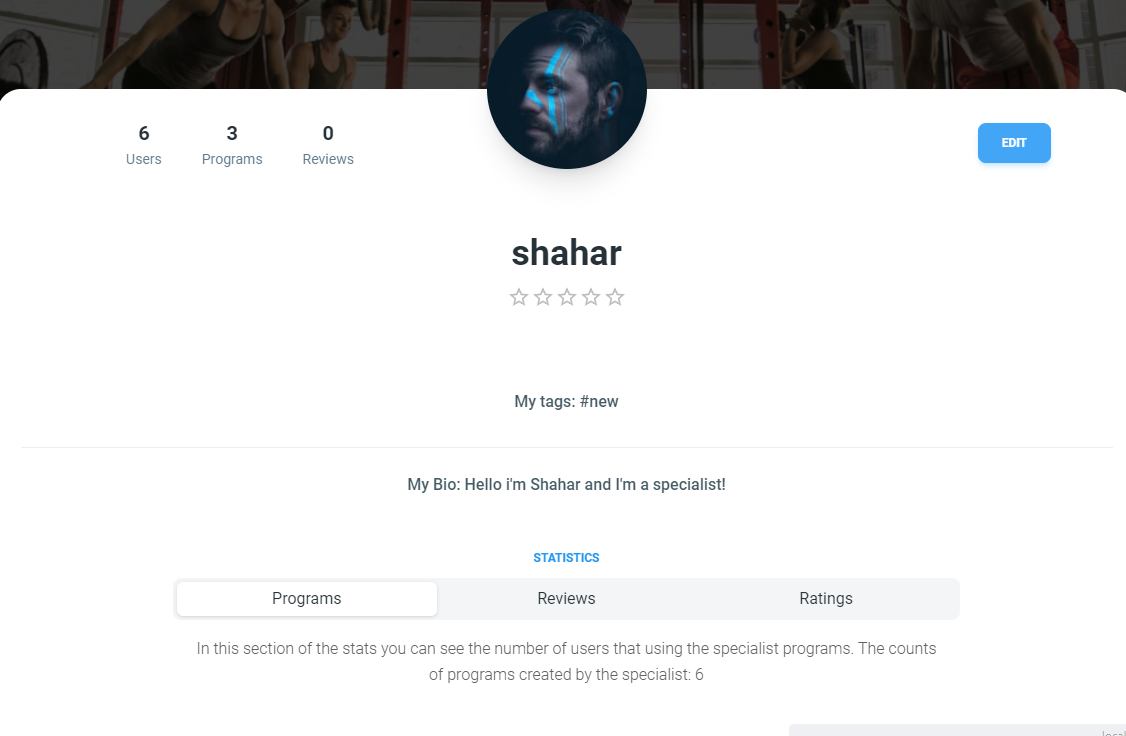
**Specialist waiting list:**On the viewing page of the waiting list, the specialist will see all the users who are waiting for program approval. The table will show the applicant whose program and which program he has. The specialist can approve or reject the user's request.



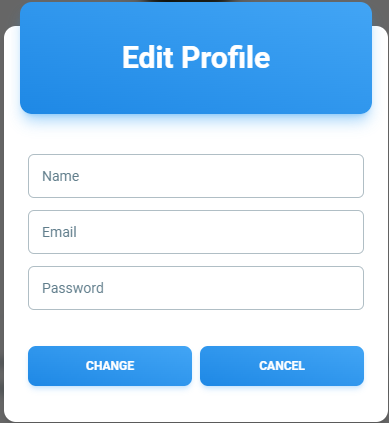
In the situation where the specialist has no active requests, he will see the page empty.



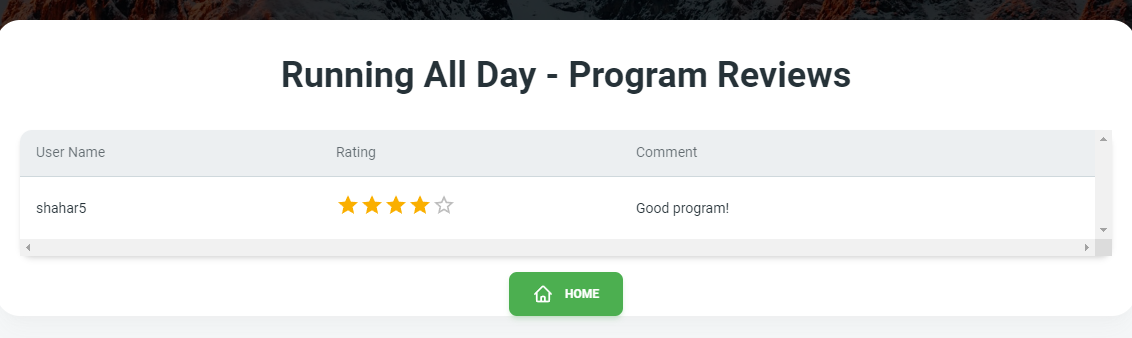
**Specialist Profile:**On the specialist's profile page it will be possible to see the details of the specialist and his statistics. The number of programs he owns, how many users use his programs, the number of reviews made to his programs and his rating based on these reviews. By clicking on the "Edit" button, the specialist can edit his personal details.



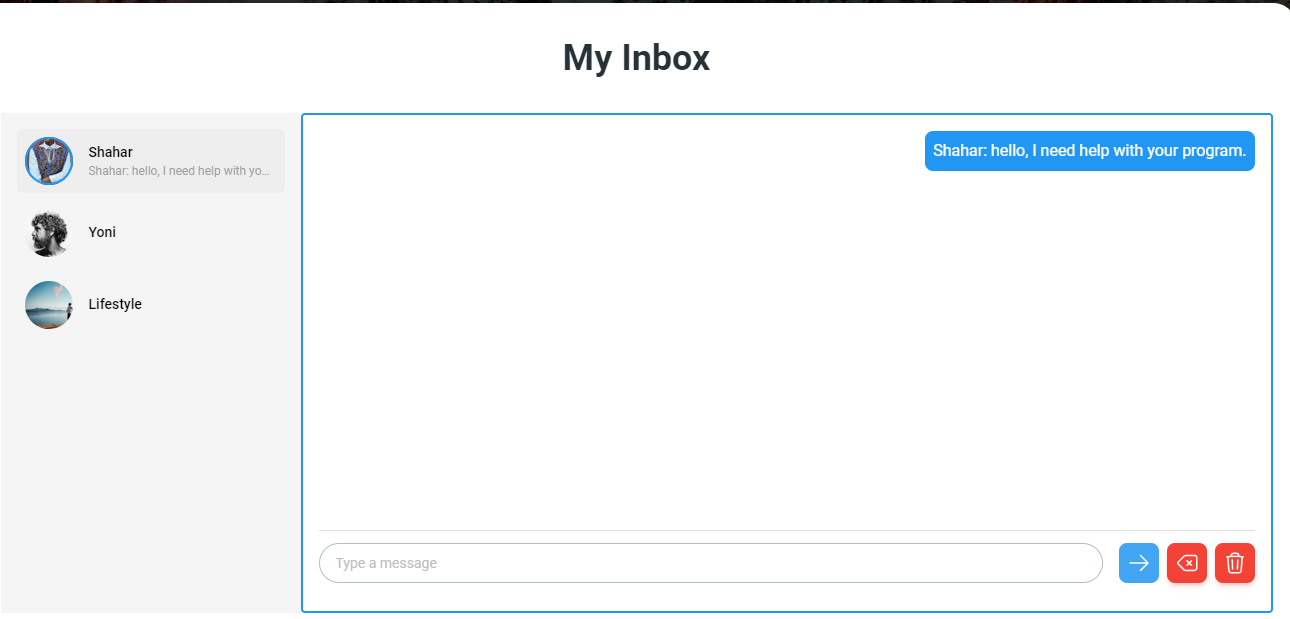
On the editing page, the specialist can choose which detail he wants to change. By clicking the "Change" button will change its details.



**Specialist Reviews:**On the reviews viewing page, the specialist can view all the reviews he received for a specific program, the name of which he will see above, and then in the table the rating and comments he received on the program and which user gave them.



**Inbox:**On the inbox page, the user/specialist can send messages to active users in the application and communicate with them. By pressing the send button or by pressing the Enter key a message will be sent. The user can switch between user conversations by clicking on the desired user on the left. Clicking the "Trash" button will delete the history of the diversion



## 3.2 Maintained Guide

### 3.2.1 Use case diagram

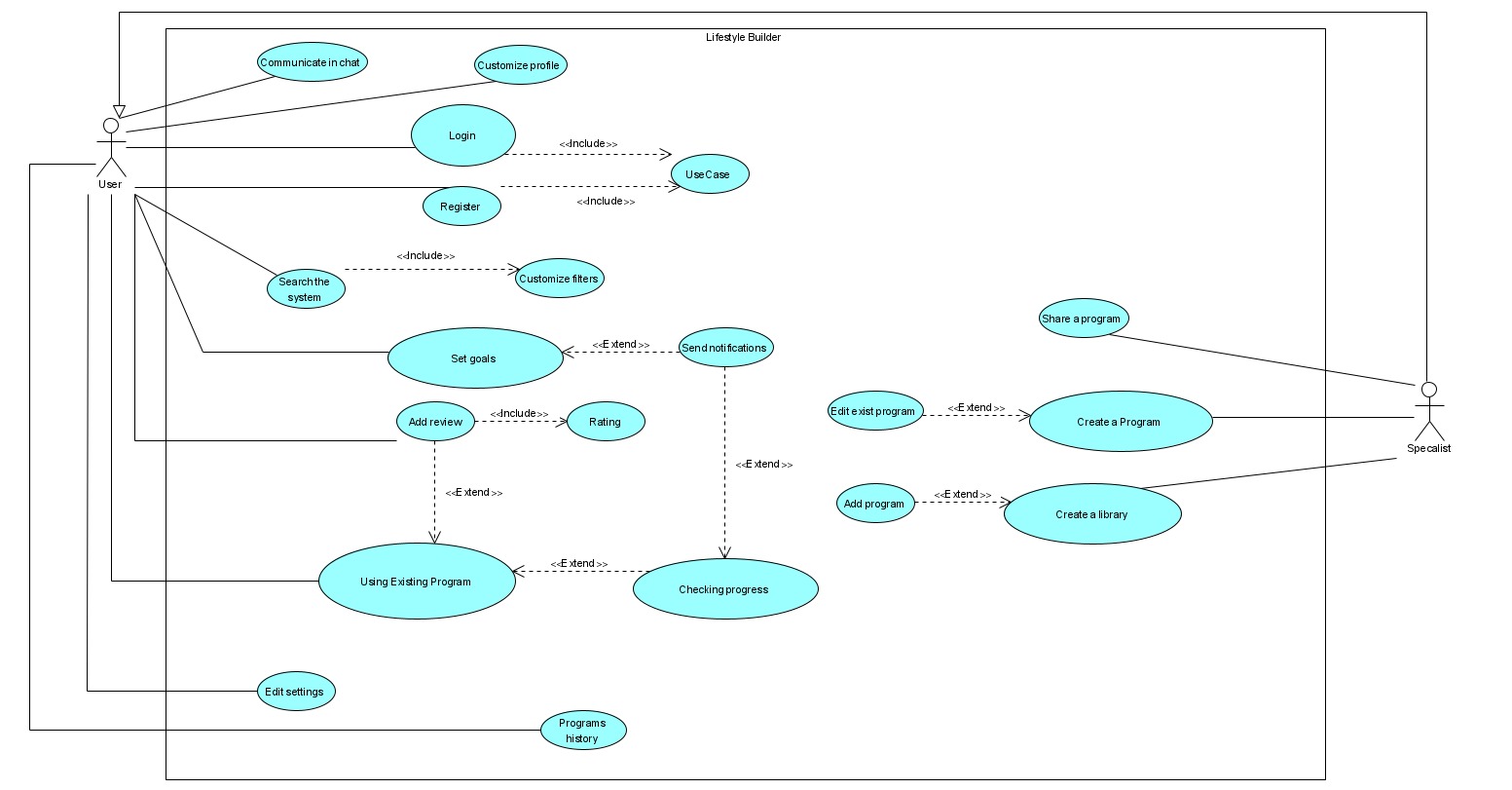


Fig1: Use Case diagram

### 3.2.2 Class diagram

Fig2: Class diagram

### 3.2.3 Activity diagram

**Create Program**:

1. The Specialist login into the application
2. The system validates the login and presents the homepage to the specialist.
3. The specialist decides to create a new program.
4. The specialist adding content to the program.
5. The program is being saved in the database.
6. The specialist needs to choose if he wants to save the program in an existing library
7. The answer is "Yes" jumped to step 10.
8. The answer is "No " the specialist is creating a new library.
9. The new library is saved in the database.
10. The program is saved in the chosen library.
11. The system will ask if the specialist wants to share the program.
12. The answer is "No" jump to step 14.
13. The answer is "Yes" the specialist is sharing the program through any social media.
14. End of process.

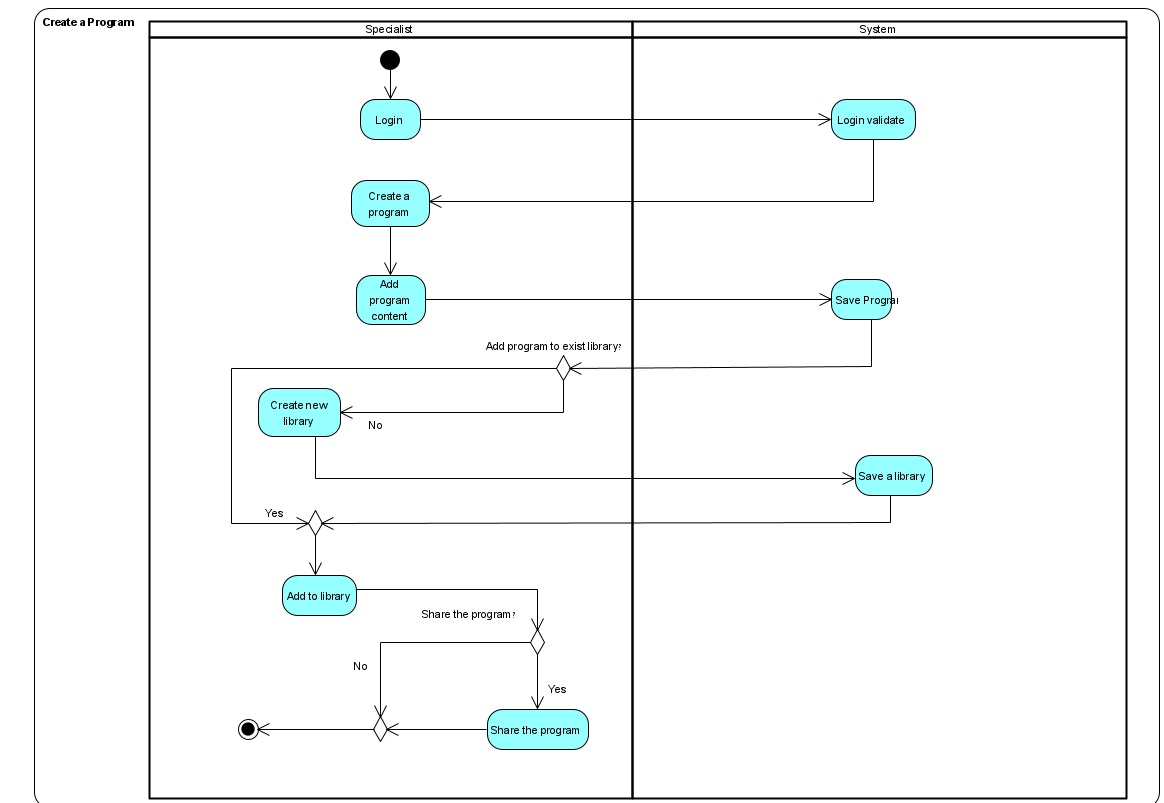


Fig3: Activity diagram 1

**Daily Goals**

1. The user login into the application
2. The system validates the login and presents the homepage to the user.
3. The user checks if he has any active daily program.
4. The answer is "No" jump to step 14.
5. The answer is "Yes" the user selects an active program.
6. The user checks his daily goals.
7. The system uploads his goals from the database.
8. The user attempts a relevant medical assignment.
9. The user documenting the assignment.
10. The data is saved in the database.
11. The system asks the user if he finish the program goals.
12. The answer is "Yes" jump to step 3.
13. The answer is "no" the user mark the finished goals and jumps to step 8.
14. End of process.

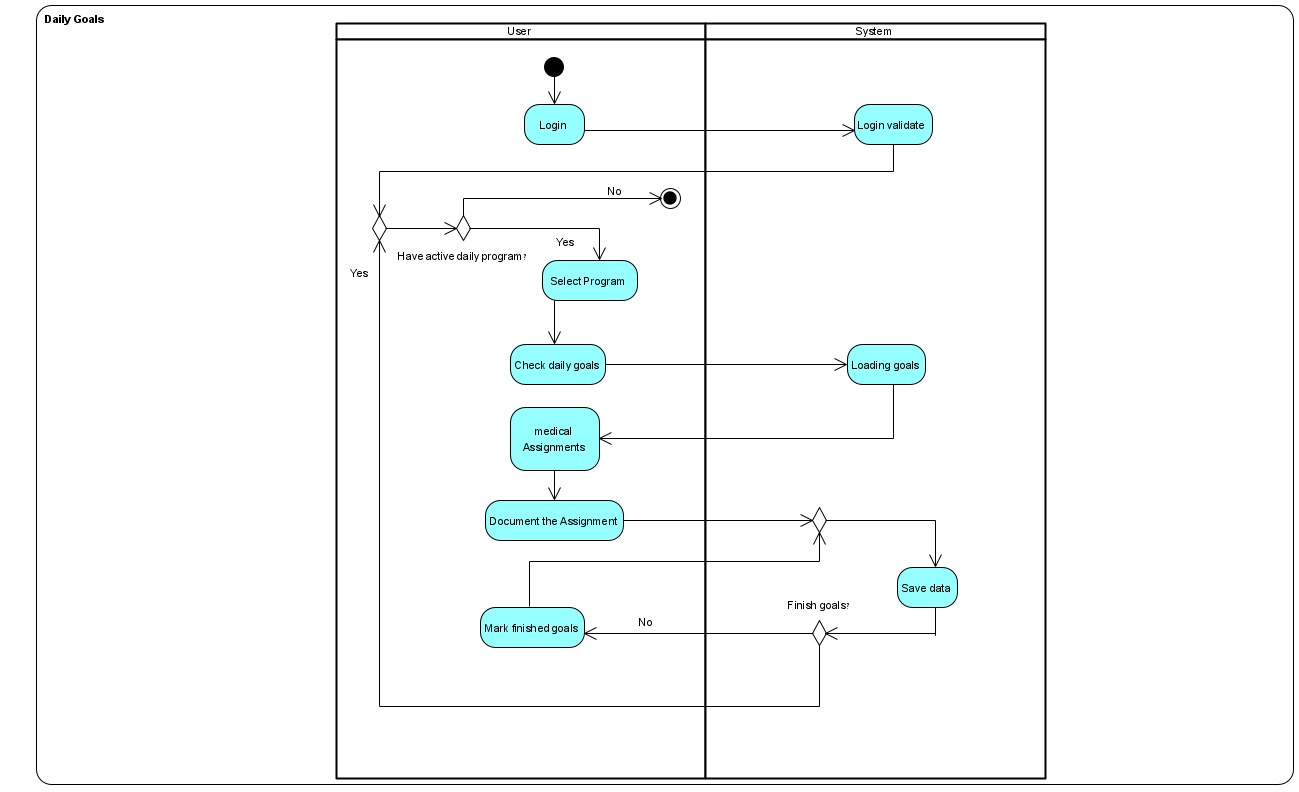


Fig3: Activity diagram 2

**Review a program**

1. The user login into the application.
2. The system validates the login and presents the homepage to the user.
3. The user selects any program.
4. The system asks what he wants to do.
5. The answer is "Nothing" jump to step 17.
6. The answer is "Select other programs" jump to stem 3.
7. The answer is "Watch reviews" jump to step 15.
8. The answer is "Review a program" the user must do two things to continue.
9. First the user must rate the program.
10. Second the user has to review the program.
11. The new total rate of the program is calculated, and the review is saved.
12. The system asks if the user wants to choose another program.
13. The answer is "Yes" jump to step 3
14. The answer is "No" jump to step 15.
15. The user watches the existing program's reviews.
16. After the user finishes with the reviews jump to step 4
17. End of process.

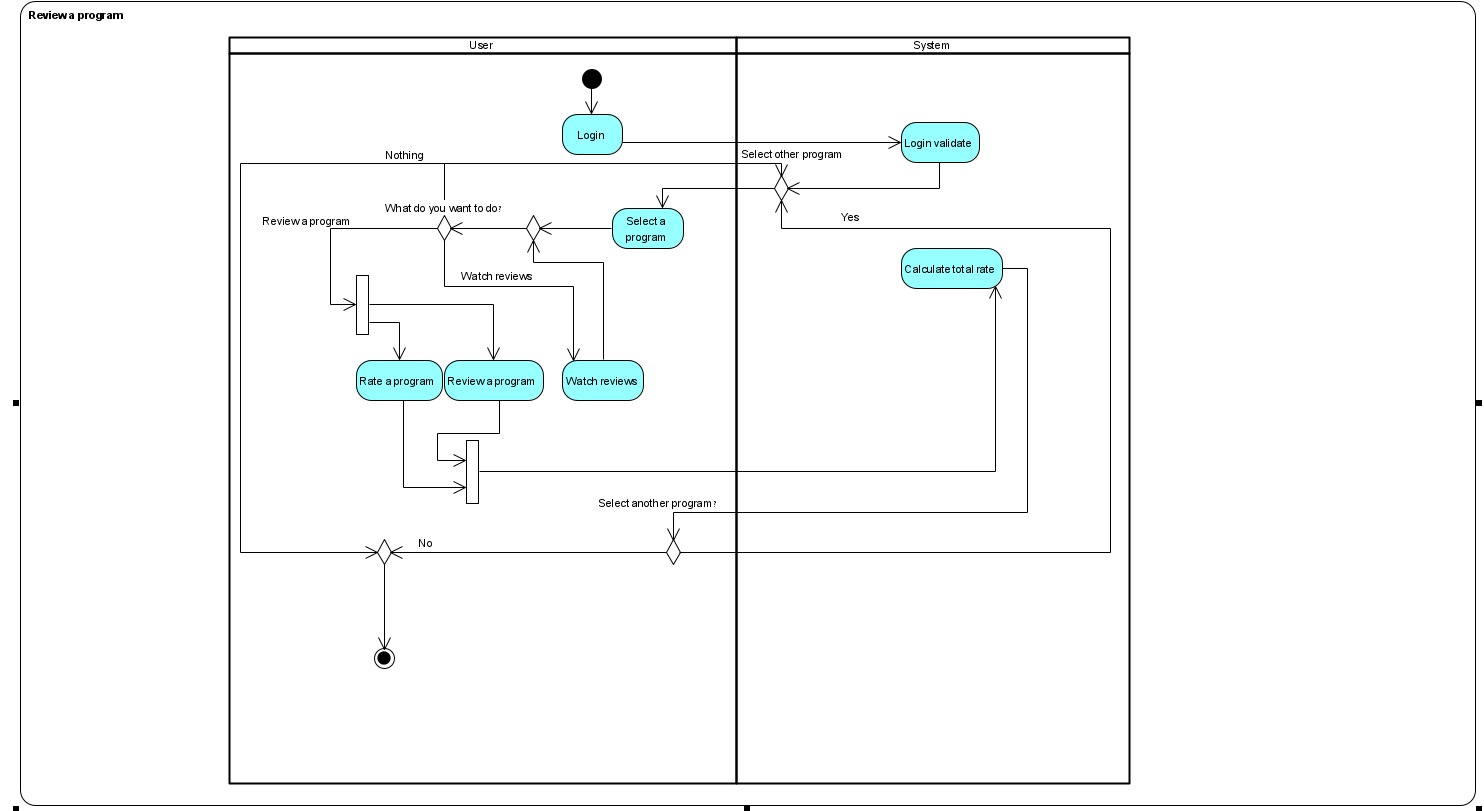


Fig3: Activity diagram 3

3.3 Software and hardware infrastructures  
To run the application, the Node.JS needed with the following dependencies:  
Server Dependencies:

    "bcrypt": "^5.1.0",

    "bcryptjs": "^2.4.3",

    "cors": "^2.8.5",

    "dotenv": "^16.0.3",

    "express": "^4.18.2",

    "express-async-handler": "^1.2.0",

    "firebase": "^9.20.0",

    "firebase-admin": "^11.7.0",

    "jsonwebtoken": "^9.0.0",

    "mongodb": "^5.3.0",

    "mongoose": "^7.1.0",

    "nodemon": "^2.0.22",

    "redux": "^4.2.1"

Execute server line: npm run dev

Client Dependencies:

  "dependencies": {

    "@emotion/react": "^11.11.0",

    "@emotion/styled": "^11.11.0",

    "@heroicons/react": "^2.0.12",

    "@material-tailwind/react": "^1.2.4",

    "@mui/material": "^5.13.3",

    "axios": "^1.4.0",

    "prop-types": "^15.8.1",

    "react": "^18.2.0",

    "react-dom": "^18.2.0",

    "react-icons": "^4.9.0",

    "react-router-dom": "^6.4.2",

    "react-share": "^4.4.1"

  },

  "devDependencies": {

    "@types/react": "^18.0.22",

    "@types/react-dom": "^18.0.7",

    "@vitejs/plugin-react": "^2.2.0",

    "autoprefixer": "^10.4.13",

    "postcss": "^8.4.18",

    "prettier": "^2.7.1",

    "prettier-plugin-tailwindcss": "^0.1.13",

    "tailwindcss": "^3.2.1",

    "vite": "^3.2.0"

  }

Execute client line: npm run dev

## 3.4 Database

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** **Name** | **Value** | **Type** | |
| users | userID | ObjectId | |
| name | String | |
| username | String | |
| email | String | |
| password | String | |
| medicalHistory | String | |
| hight | Number | |
| weight | Number | |
| role | String | |
| programs | program | ObjectId |
| programStatus | String |
| messages | [messages] | |
| numOfMessages | Number | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** **Name** | **Value** | **Type** | |
| specialists | userID | ObjectId | |
| name | String | |
| username | String | |
| email | String | |
| password | String | |
| patients | [users] | |
| programs | [program] | |
| rating | Number | |
| requests | user | User: ObjectId |
| program | Program: ObjectId |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** **Name** | **Value** | | **Type** | |
| programs | programId | | ObjectId | |
| duration | | Number | |
| startDate | | Date | |
| endDate | | Date | |
| limtedDuration | | Number | |
| specialist | | ObjectId | |
| activities | | [activity] | |
| kindOfProgram | | String | |
| reviews | | [reviews] | |
| rating | | Number | |
| dailyActivities | | day | Number |
| dailyActivity | [activity] |
| **Table** **Name** | | **Value** | **Type** | |
| Review | | reviewID | ObjectId | |
| userID | ObjectId | |
| rating | Number | |
| comment | String | |

|  |  |  |
| --- | --- | --- |
| **Table** **Name** | **Value** | **Type** |
| Activity | activityId | ObjectId |
| name | String |
| previousActivityId | ObjectId |
| completed | boolean |
| feedback | String |

3.5 Results and conclusions  
Our project have effectively accomplished its primary objective of establishing a platform that facilitates seamless connections between users and specialists, empowering the creation and provision of programs tailored to individual needs. Through our application, users can effortlessly locate and access the desired programs, enhancing their journey towards a healthier lifestyle.

Furthermore, we have successfully developed a user-friendly and versatile application capable of operating across various platforms. Employing responsive design principles, we ensured optimal user experiences across different screen sizes and resolutions, accommodating the diverse preferences and habits of our user base.

Throughout the project's duration, we encountered numerous challenges that necessitated the acquisition of new knowledge and skills. To effectively implement our vision, we undertook the task of familiarizing ourselves with technologies such as React, Node.js, CSS design patterns (Material-tailwind) and MongoDB. This demanding process demanded dedication and perseverance, ultimately empowering us to overcome obstacles and deliver a robust and functional application.

To address coding challenges that emerged during development, we leveraged valuable online resources like research unit of ChatGPT and the renowned developer community platform, Stack Overflow. This enabled us to seek guidance and assistance from experienced professionals worldwide, facilitating efficient troubleshooting and continuous progress in our project implementation.

Overall, the development of this system has provided us with invaluable experience and insights. Our utilization of Git for version control allowed us to efficiently track and manage changes made to the project, ensuring seamless collaboration among team members. As we delved into React, we gained a deep appreciation for its strengths and potential in building modern, interactive user interfaces.

In conclusion, our project has achieved its goals by creating a platform that seamlessly connects users and specialists, delivering user-friendly cross-platform functionality. We have embraced new technologies, overcome challenges, and honed our skills throughout the development process, resulting in a successful and impactful application.

In pursuit of enhancing the system's architecture, several potential future improvements arise. One such improvement involves the development of program libraries built upon existing programs within the system, thereby imbuing them with fresh and innovative content.

Furthermore, an additional avenue for enhancement lies in enabling live interactions through real-time calls between the system's expert and the user. This feature facilitates seamless communication and fosters a dynamic and engaging user experience.

References**:**

**2.3.1 - JavaScript :** <https://he.wikipedia.org/wiki/JavaScript>  
**2.3.2 - Client Server Definition:** <https://www.omnisci.com/technical-glossary/client-server>  
**2.3.3 - Agile Development:** <https://www.techtarget.com/searchcio/definition/Agile-project-management>  
**2.3.4 - ReactJS:** <https://reactjs.org>  
**2.3.5 - React Router:** <https://reactrouter.com/en/main>

**2.3.6 - Render :** <https://legacy.reactjs.org/docs/rendering-elements.html>  
**2.3.7 - Hooks:** <https://legacy.reactjs.org/docs/hooks-intro.html>  
**2.3.8 - React-Bootstrap:** <https://react-bootstrap.github.io/>  
**2.3.9 - NodeJS:** <https://nodejs.org/en/>  
**2.3.10 - MongoDB:** <https://www.mongodb.com/>  
**2.3.11 - Express.JS**: <https://expressjs.com/>  
**2.3.12 - DOM:** <https://he.wikipedia.org/wiki/Document_Object_Model>

**2.3.13 - Axios:** <https://axios-http.com/docs/intro>

**2.3.14 - JSX:** <https://legacy.reactjs.org/docs/introducing-jsx.html>

**2.3.15 - Material-Tailwind:** <https://www.material-tailwind.com/>