

Dedication

Thanks to Almighty **Allah**
who gave me the strength and patient to finish this work.

To my beloved parents **Samira and Ridha**
For raising me to believe that anything was possible. For always loving and supporting me,
without you none of my success would be possible.

For my sister Meriem and my brother Anis You are my window to the world of childhood, the joy and gaiety in life that I see in you, your eternal love, give me hope for a better future for all of us.

To all my friends:
You have always supported and encouraged me during these years of study. Thanks for being
always there for me.

To all my family members:
For the exceptional moments and unlimited care and support.

Thanks

Above all, I thank God, almighty, for all that he has given me in all that I have undertaken in my life so much courage and patience

The realization of this work was possible thanks to the participation of several people to whom we would like to express our gratitude.

I thank Mr. Kais Jrad , the director of Mega-DEV that eased my way through this project. Special thanks to Mr. Ben Ftima Fakher who helped me complete this work and who agreed to supervise my internship and for their advice which was very precious to me. Finally, my thanks also go to all the teachers of the Higher Institute of Computer Science and Mathematics of Monastir, who constantly combine their efforts to provide us with solid training.

My thanks go finally to the members of the jury, who will give my work value added through their recommendations and remarks so important, hoping they find in this report the qualities of clarity and motivation they expect, for which I will be very grateful.

Table of Contents

Chapter 1 : Context and Objectives.....	7
Introduction.....	7
1.1 Internship Context.....	7
1.2 Presentation of the host company.....	7
1.3 Motivation and problematic	8
1.4 Study of existing solution on the market	8
1.4.1 fitness Blender.....	8
1.4.2 BodyBuilding.....	11
1.4.3 6weeksixpack.....	13
1.5 Criticism of the existing solutions	15
1.6 Our solution.....	16
1.7 Methodologies	17
1.8.1Waterfall Approach	17
1.8.2Scrum.....	18
1.9conclusion.....	20
Chapter 2 : Specification and Conception	21
2.1Introduction.....	21
2.2Functional requirements	21
2.3Non-Functional requirements	22
2.5 Identification of Actors.....	23
2.4 Use-case diagrams.....	23
2.4.1Global use-case diagram:	24
2.4.2Authentication use-case diagram:.....	25
2.4.3Workout Management use-case diagram.....	26
2.4.4Chat Management use case Diagram.....	29
Conclusion	32
Chapter 3 :Design and Architecture	33
3.1 Introduction.....	33
3.2 Design Pattern	33
3.2.1 MVC MERN Stack.....	33
3.2.2 MVC Flow.....	35

3.3Physical Architecture.....	36
3.4 Deployment Diagram	37
3.5 Logical Architecture.....	38
3.6 Sequence diagrams	40
3.5.1 Authentication Sequence Diagram.....	41
3.5.1 User Management Sequence Diagram.....	41
3.5.1 New Chat Sequence Diagram	43
3.6 General Class Diagram.....	43
3.7 Gantt Diagram	44
3.8 Conclusion	45
Chapter 4 : Realization	46
4.1Introduction.....	46
4.2 Working environment and tools	46
4.2.1 Material Environment	46
4.2.2 Software Environment	47
4.3 Implementation.....	53
4.3.1 Home Interface.....	53
4.3.2 Registration interface	54
4.3.3 Login Interface.....	55
4.3.4 Admin Dashboard interface	56
4.3.5 Create workout interface	57
4.3.6 Manage Workout Interface	59
4.3.7 Manage Workout Interface	59
4.3.8 Chat Interface	60
4.3.8 Create Group Chat Interface	60
4.3.8 Leaderboard Interface	61
4.3.9 Profile Interface.....	62
Conclusion	62
General conclusion and perspectives.....	63
Webography	64

List of figures

Figure 1.1 Fitness Blender Home Page	8
Figure 1.2Fitness Blender video page	8
Figure 1.3:Bodybuilding home page	10
Figure 1.4:Bodybuilding Body Fit page	10
Figure 1.5: weeksixpack Home Page	12
Figure 1.6:sixweeksixpac Home Page	12
Figure 1.7: Waterfall methodology diagram	16
Figure 1.8: Scrum Process Diagram	17
Figure 2.1:General use case diagram	22
Figure 2.2:Authentication use case diagram	23
Figure 2.3:Workout Management Use-Case Diagram	25
Figure 2.4:Chats Management Use Case Diagram	28
Figure 3.1:The Controller Figure 3.2:The View	33
Figure 3.3:The model	34
Figure 3.4:MVC flow	34
Figure 3.5:Physical Architecture	35
Figure 3.6:Deployment Diagram	36
Figure 3.7:Logical Architecture Diagram	37
Figure 3.8:BackEnd diagram	38
Figure 3.9:Redux diagram	39
Figure 3.10:Authentication Sequence Diagram	40
Figure 3.11:User Management Sequence Diagram	41
Figure 3.12:Create New Conversation Sequence Diagram	42
Figure 3.13:: General Class Diagram	43
Figure 3.14:Gantt Diagram	44
Figure 4.1:Computer characteristics	46
Figure 4.2:ReactJS logo	47
Figure 4.3:Redux Logo	47
Figure 4.4:Bootstrap Logo	47
Figure 4.5:Reactstrap Logo	48
Figure 4.6:Node.js Logo	48
Figure 4.7:Express.js logo	49
Figure 4.8:MongoDB logo	49
Figure 4.9:Socket.io Logo	49
Figure 4.10:Visual Studio Code Logo	50
Figure 4.11:StarUML Logo	50
Figure 4.12:draw.io Logo	51
Figure 4.13:Git Logo	51
Figure 4.14:Postman Logo	51
Figure 4.15:Microsoft Word Logo	52

Figure 4.16:Home page	53
Figure 4.17:Registration Interface	54
Figure 4.18:Login interface	55
Figure 4.19:Admin Dashboard	56
Figure 4.20:Create workout 1	57
Figure 4.21:Create Workout 2	57
Figure 4.22:The Manage Workout Interface	58
Figure 4.23:Manage Users Interface	58
Figure 4.24:Chat Interface	59
Figure 4.25:Create Group Chat Interface	60
Figure 4.26:Leaderboard Interface	60
Figure 4.27:Profile Interface	61

Tables

- 1.1 Comparison of Existing Apps and current project
- 2.1 Authentication use case scenario
- 1.1 Restore password use case scenario.
- 1.1 Table 2.4: Consult workout use case scenario.
- 1.1 Create workout use case Scenario.
- 1.1 Modify workout use case scenario.
- 1.1 Delete Workout use case scenario.
- 1.1 Chats Management Use Case Diagram.
- 1.1 Consult Chats Use Case Scenario
- 2.9 Start New Conversation use case scenario
- 1.1 Create Group Chat use case Scenario
- 1.1 Add users to group chat use case Scenario.

General Introduction

Physical activity or exercise can improve our health and reduce the risk of developing a number of diseases, such as diabetes, cancer and cardiovascular disease. It can have immediate and long-term health benefits. Most importantly, regular activity can improve our quality of life.

So, as more people realize this important fact, we tend to notice an increase in the number of people attending gyms, or seeking fitness related content online.

When someone chooses online fitness solution, it means that they get to save money, time and achieve the desired goal all in the comfort of your home.

It is in this context that our end-of-studies project will consist of making an application to manage fitness activities. The purpose of this report is to present the approach followed to achieve the creation of a Workout Management Application.

We present in the following the general organization of our report which is organized in four chapter:

1. The first chapter, **Context and Objectives**, is dedicated to the presentation of the host organization Mega-DEV and the general overview of the project, the subject problematics, the description of the proposed solution and the adopted methodology of work.
2. The second chapter, **Specification and Conception**, describes the specification of needs through describing the non-functional and functional requirements in an informal way, an in a semi-forma way through use case diagrams.
3. The third chapter, **Design and Architecture**, focuses on the general design and architecture of the proposed system, as well as providing the class diagram and the sequence diagrams.
4. Finally, the fourth chapter entitled **Realization** presents the tools we used for the development of our project and this report, then we finish by showing screenshots of our application

The report ends with a general conclusion in which we summarize the synthesis of the work done and prospects

Chapter 1 : Context and Objectives

Introduction

This chapter will be divided as follows: First, I will present the context of the project and its problematic, as well as give an overview of the host company, Second, I will present the solution that I developed after in-dept study of existing solution on the market, then I will specify the work methodology that I will follow through developing the application and a conclusion

1.1 Internship Context

The work in this report was carried out as part of my end-of studies project to obtain an applied license in computer science at the Higher Institute of Computer science and Mathematics of Monastir. This internship was carried out within Mega dev agency on the subject of: Design and development of a web application that helps users achieve their fitness goals.

1.2 Presentation of the host company

Mega-DEV is a web design and development company that provides end-to-end development service for web and mobile development.

It was created in 2010 in Houmet-souk, djerba. Since then, it has developed and created several projects for multiple local organization and foreign ones and in

different sectors such as health care, Real Estate, travel, E-commerce, Education ... etc.

1.3 Motivation and problematic

Every single person wants to be the best version of himself, everyone just wants to become fit, to lose weight and to build on lean muscle, and we all know how working out benefits your health and reduces stress as well as being an energy source for your body. But not everybody can do it.

Not everyone has a gym nearby or has the time to go to the gym in the first place, and even they did have the time, sticking to going to the gym 3 or 4 time a week can be very challenging as your motivation level decreases over time.

Which is why a need for a solution that helps people work out from home arises.

Such a solution could help millions of people around the globe achieve their dream of becoming fit, it could:

- guide people in their fitness journey without making mistake
- Connect them with other people to stay motivated along the way

In this report, I will present my solution as well as go over some of existing web application that tend to offer a solution to this problem.

1.4 Study of existing solution on the market

In this section, I will showcase the different solutions available on the market. The main purpose of this section is to find the strong features and drawbacks of these solution so that the development of my project is oriented into making it the best available option for its target users.

Here's a demonstration of the available web applications that I tested:

1.4.1 fitness Blender

Fitness Blender is a fitness platform that offers full-length workout videos, workout programs as well as diet and meal plans. It also provides articles and information related to fitness.

This is being done through their platform which is optimized for user experience

Strong Features

Fitness Blender has some attractive feature such as:

Community page: This is where the trainer post articles about fitness which the user of the website can interact by writing comments and ask questions.

Full-length workout videos: Voice instructions that help with correctly executing the moves and low injury risk due to guidance.

- **Content:** Fitness Blender has a great selection of videos for any fitness level;

Calendar: this feature allows the clients to set up a remainder of which full-length workout videos to follow and on which date and time.

Drawbacks:

Despite its interesting features, it still has some weak points

-It doesn't connect users with each other to help motivate and encourage them to stick to their programs until the end. This is crucial since people going through diet often experience undesirable effects of withdraw of certain food such as sugar, having someone to encourage during these difficult times is paramount for success

-It doesn't show the results of people that successfully finished their programs with noticeable results

-They don't offer specific programs to specific user while taking consideration the user's condition such as age, gender and available time

-Workouts and Food plans are bought separately which could prove expensive

In conclusion, fitness blender is easy to use and a good source of multiple types of workouts, but it doesn't have the features to give selected workouts to selected user, as well as connecting participants together

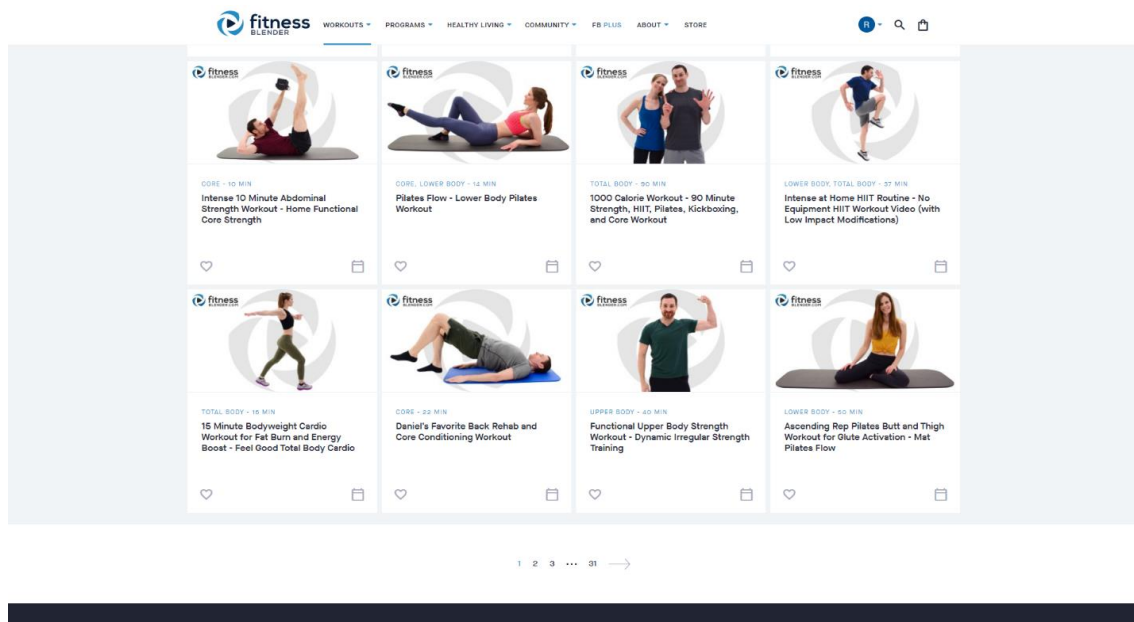


Figure 1.1 Fitness Blender Home Page

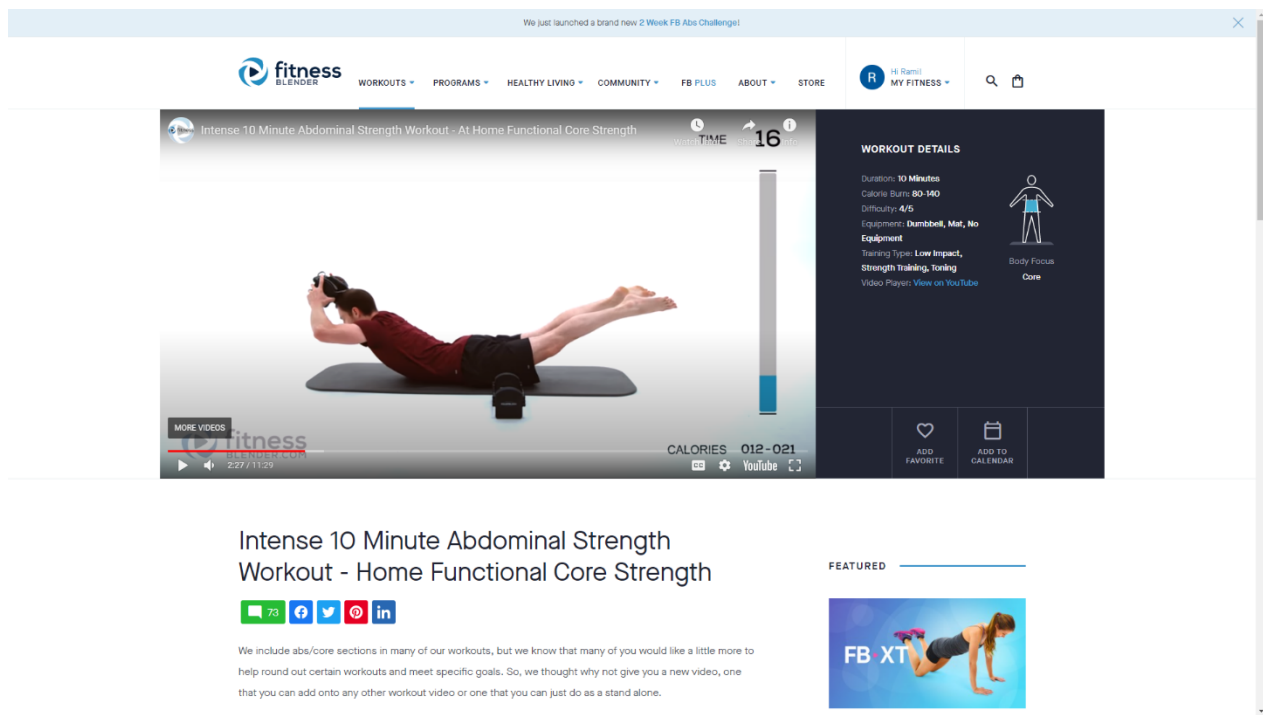


Figure 1.2 Fitness Blender video page

1.4.2 BodyBuilding

Originally, BodyBuilding.com is an online retailer of dietary and sports supplement company.

But recently, it has become a great resource for workout plans created by experts. It has a huge database of workout videos which are updated regularly.

Strong features

Among it's features, BodyBuilding.com offers:

Customizable fitness plans

Step-by-step workout guides

Community where users can interact with each other and share fitness-related information

New content regularly

Drawbacks

Even though BodyBuilding.com offers efficient and dynamic services. There are some weak points.

-It's mainly a shop, and its main interface is designed to promote their supplement products and equipment.

-Most of their diets and workouts require the user to buy supplements.

-Workout plans are offered to everybody users have to choose the workout plans themselves.

-Most of their workout require gym equipment.

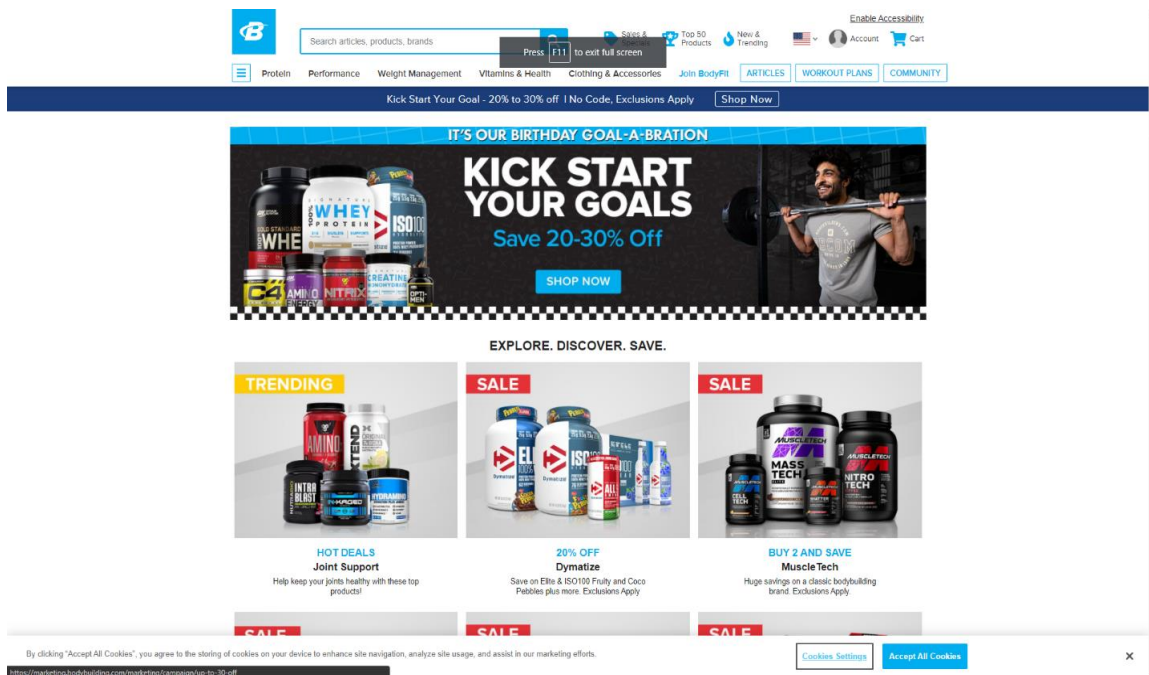


Figure 1.3: Bodybuilding home page

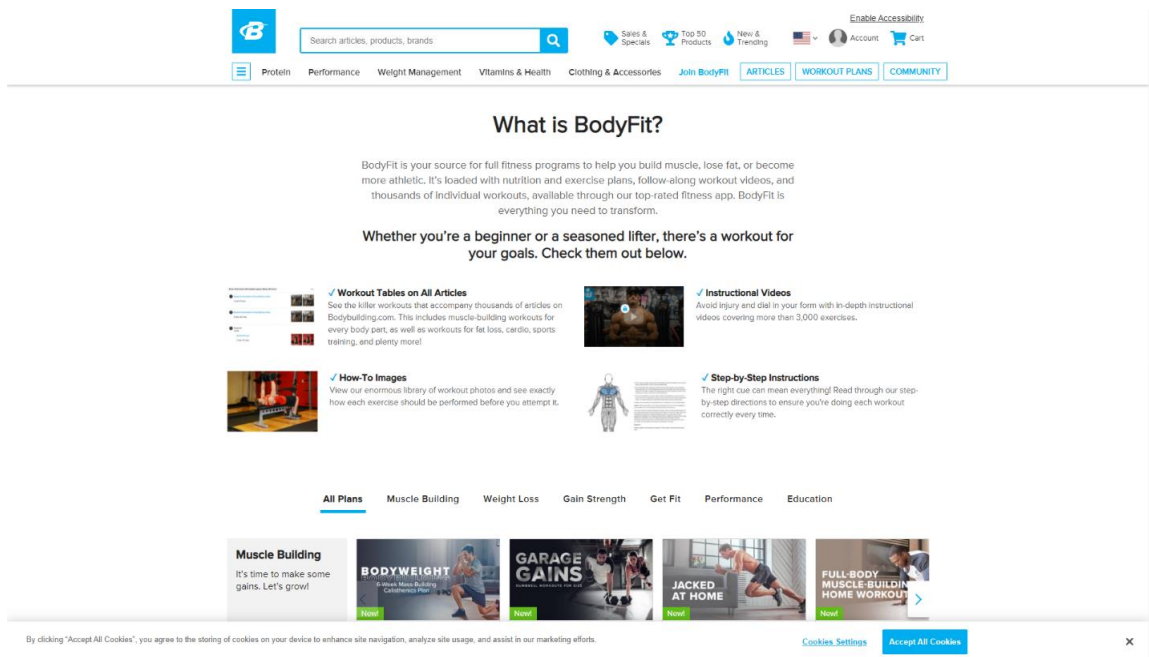


Figure 1.4: Bodybuilding Body Fit page

In conclusion, Bodybuilding.com is a great source of workout plans and it isn't expensive, but the fact that it's a store for supplements and that it promotes supplements with workout can be confusing to its users.

1.4.3 6weeksixpack

6weeksixpack is basically website that offers a challenge, after signing up, the user receives an email containing the link to the webpage that contains the workout for that day.

Strong features

- the users receive daily emails which keeps them motivated
- each webpage contains a video specific for one day and it doesn't get sent twice

Drawbacks

- Users can't see the entire program from the start, they have to wait until the specific day.
- The workouts are only about abdominal muscles, not the entire body.

In conclusion, 6packsixweek doesn't offer enough workout plans, nor does it connect users with each other, but the fact that users need to tune in each and every day to find out what needs to be done is a huge factor into helping participants into sticking until the end of the program.



Figure 1.5: weeksixpack Home Page

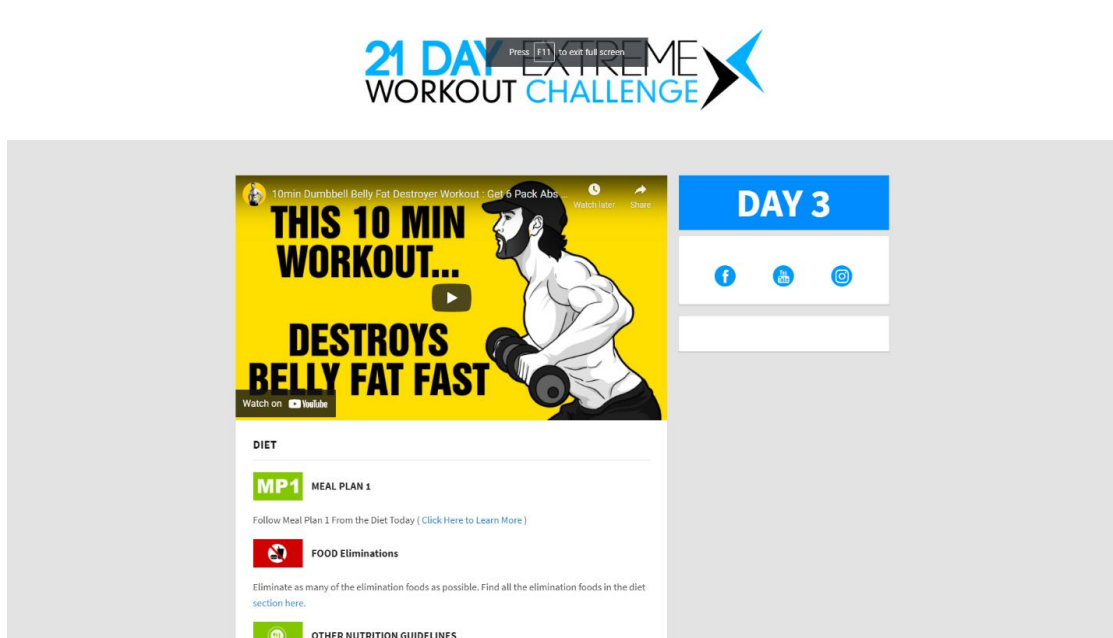


Figure 1.6:sixweeksixpac Home Page

1.5 Criticism of the existing solutions

In this section, I will move on to the criticism of the existing solutions based on comparative research aimed at emphasizing the advantages that my application will bring.

Even though they're all fitness application that provide workout plans for its users, they differ in terms of several criteria which are defined as follow:

- Design
- User experience
- Full-body workouts
- Home workouts
- Interaction between users
- Progress tracking
- Assigning workouts while considering their conditions
- Easy to use

Table 1.1 represents the result of a comparative study between the different solutions based on the criteria defined above

	Fitness blender	BodyBuilding.com	6weeksixpack
Design	V	x	x
User expericence	v	x	x
Full body workouts	v	v	x
Home workouts	v	v	v
Interaction between users	x	v	x
Progress Tracking	x	v	x
Assigning workouts for specific conditions	x	x	x
Easy to use	v	x	v

Table 1.1: Comparison of Existing Apps and current project

1.6 Our solution

After the project study and researches, we opted for creating a web application that responded to its users in a way that the other existing ones never could.

-Fitness planet is going to be a dynamic and interactive web application that allows creating a fitness program and to track users who are undergoing that program

-Fitness planet is going to allow its user to receive a fitness program that meets their goals, and in the same time takes into consideration multiple factors like user's age, gender and available time for workout.

-Every fitness program in Fitness Planet will contain all the needed resources such as videos, articles etc.

-It will also allow its users to connect with other users going through the same fitness program in support groups and to compete to be on top of the leaderboard. These groups will provide needed motivation for its users to stick to the program until the end

-This solution will minimize the costs of hiring a personal trainer and help those who want to work out at home instead of going to the gym.

1.7 Methodologies

The methodology is a rationally organized approach to achieve a result. Among the different existing methodologies, we find the the waterfall approach which is well known to be used in simple projects whose needs are clear and well defined from the beginning, Agile methodologies (Scrum and extreme programming) characterized by their flexibility and used in large project.

1.8.1 Waterfall Approach

The waterfall model is a linear project management approach, where stakeholder and customer requirements are gathered at the beginning of the project, and then a sequential project plan is created to accommodate those requirements. The waterfall model is so named because each phase of the project cascades into the next, following steadily down like a waterfall

The waterfall model has, at least, five to seven phases that follow in strict linear order, where a phase can't begin until the previous phase has been completed. The specific names of the phases vary, but they were originally defined by its inventor in the following way:

-Requirements: The key aspect of waterfall is that all customer requirements are gathered at the beginning of the project.

-Design: The requirement specifications from first phase are studied in this phase and the system design is prepared

-Implementation: With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase

-Integration and Testing: All the units developed in the implementation phase are integrated into a system after testing of each unit

-Deployment: once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

-Maintenance: The customer is regularly using the product during the maintenance phase, discovering bugs, inadequate features and other errors that occurred during production. The production team applies these fixes as necessary until the customer is satisfied.

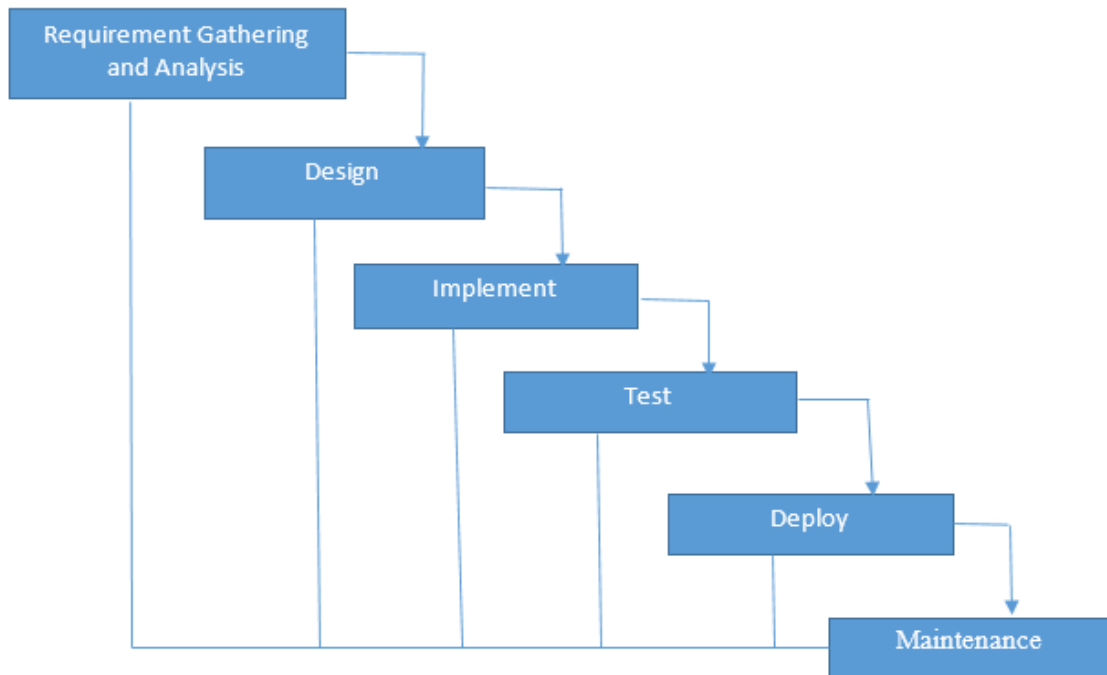


Figure 1.7: Waterfall methodology diagram

1.8.2Scrum

Scrum is an agile development methodology used in the development of Software based on an iterative and incremental processes. Scrum is adaptable, fast, flexible and effective agile framework that is designed to deliver value to the customer throughout the development of the project. The primary objective of Scrum is to satisfy the customer's need through an environment starts from a general idea of what needs to be built, elaborating a list of characteristics ordered by priority (product backlog) that the owner of the product wants to obtain.

The framework is based on the division of a project in time boxed, called "sprints". Sprints can last from a few hours to a month (with a two -week preference). Each sprint begins with an estimate followed by operational planning. The sprint ends with a demonstration of what has been completed.

The Scrum methodology is easily explained by the figure below.

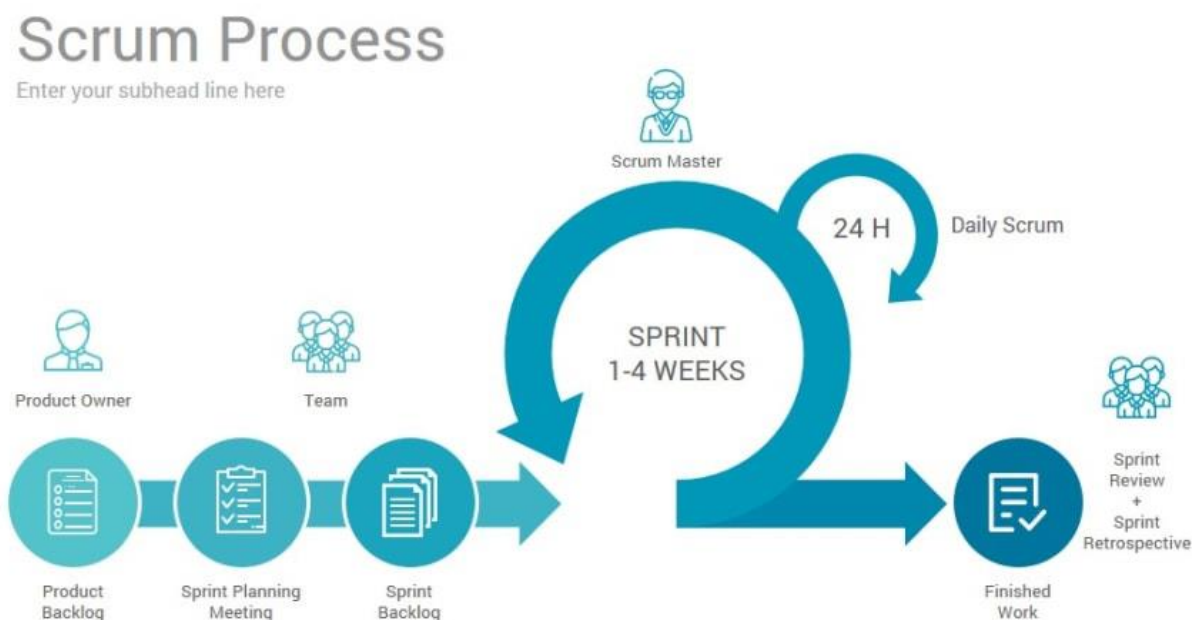


Figure 1.8: Scrum Process Diagram

The basic principles of Scrum are the following:

- Determining the functionalities needed in order to create the backlog of the product
- Defining the priorities of the functionalities and choosing which ones will be carried out in each iteration
- Work on the functionalities in iterations called Sprint
- Each sprint always delivers a functional partial product called increment

Scrum Roles

- **Product Owner:** This person is responsible for defining the work and then prioritizing those tasks. They are clear on the goals of the project, as well as those of the customer market and organization.
- **Scrum Master:** He plays the role of facilitator and guardian of the correct application. He is a product owner service. He facilitates interactions between members of the Scrum team. He acts

on the development process (development definitions of the duration of sprints, the agenda of scrum meetings ...)

- Development team: the development team is the heart of the scrum team, as they're the ones responsible for doing the actual project work. Each member of the team has a skill that, together with other team members, combines to tackle all the needs of executing the project.

Our choice

We chose the Scrum Methodology as it is the closest methodology to how we planned our tasks and timeline.

1.9 conclusion

In the chapter, we presented the host company as well as the general context of the work, followed by a study of the existing solution that are close to our application, and we defined our work methodology and finally we described the different technologies used in this project.

Chapter 2 : Specification and Conception

2.1 Introduction

After presenting the general context of our project, we will dedicate this chapter to present our project in a formal way. We will do a detailed and complete analysis of different needs related to our application which will include:

- specifying the functional and non-functional requirements of our system.
- an identification of the main actors that are going to interact with our application.
- presentation of the global use case diagram
- presentation of different detailed use case diagrams

2.2 Functional requirements

In the following, we will present the functional requirements of our project.

Functional requirements express the actions that the system must execute in response to user requests. In our case, our application must respond to the following needs:

Registration

The application must allow users to register in order to access the application

Authentication

It allows the user to connect using his email address in order to benefit from the application's features.

Consulting workout

It allows the user to open his account every day, and check his workout for that day.

Chat

It allows the user to connect and chat with other users.

Consulting support groups

It allows the user to consult available support groups and to join them

Creating support groups

It allows the user to create support groups and to invite people to these groups or make them public.

Settings:

It allows the use to consult or update settings

Goal:

It allows the user to consult or choose his goal.

Creating workout

It allows the Admin to create a workout

Consulting users

It allows the Admin to consult all the users

Changing user's workout

It allows the admin to consult or change the user's workout

2.3Non-Functional requirements

We identify non-functional needs which are operational constraints that count on the performance of the application. They are summarized as follows:

- **Scalability and maintenance:** The application must be able to adapt to any change on the implementation side (updating frameworks) in order to guarantee its evolution and flexibility. Also, it must offer a clearly readable, understandable code and modular.
- **Extensibility:** The system must be scalable and must take into account the possibility of its extension by adding new features.

- **Security:** The Application must ensure the security of personal information
- **Performance:** The modules must optimize the treatments and reduce the execution time.
- **Ergonomics:** The application is user-friendly, simple to use, ergonomic and adapted to the use
- **Reusability:** The components of the system can be reused for the development of a different applications.
- **Modularity:** The application is well structured as modules to ensure better readability, a reduction in the risk of error and the possibility of selective tests

2.5 Identification of Actors

An actor is an entity external to the system. It represents a person or other computer system waiting for one or more services offered by an interface access. It interacts with the system by sending or receiving messages

The actors that will interact with the system are:

Administrator: This is the person who manages the entire application which include managing users, and workouts.

User: He's the basic user that will benefit from the features of the application.

2.4 Use-case diagrams

A use case diagram is a behavior diagram that visualizes the observable interactions between actors and the system under development. The diagram consists of the system, the related use cases and actors.

In this section, we present the needs of our system in a formal way using the Unified Modeling Language (UML) use case diagrams. These various features are described in the diagrams below.

2.4.1 Global use-case diagram:

The Figure describes in a general way the actors of the system as well as the functionalities. A detailed description of the different use-case will follow.

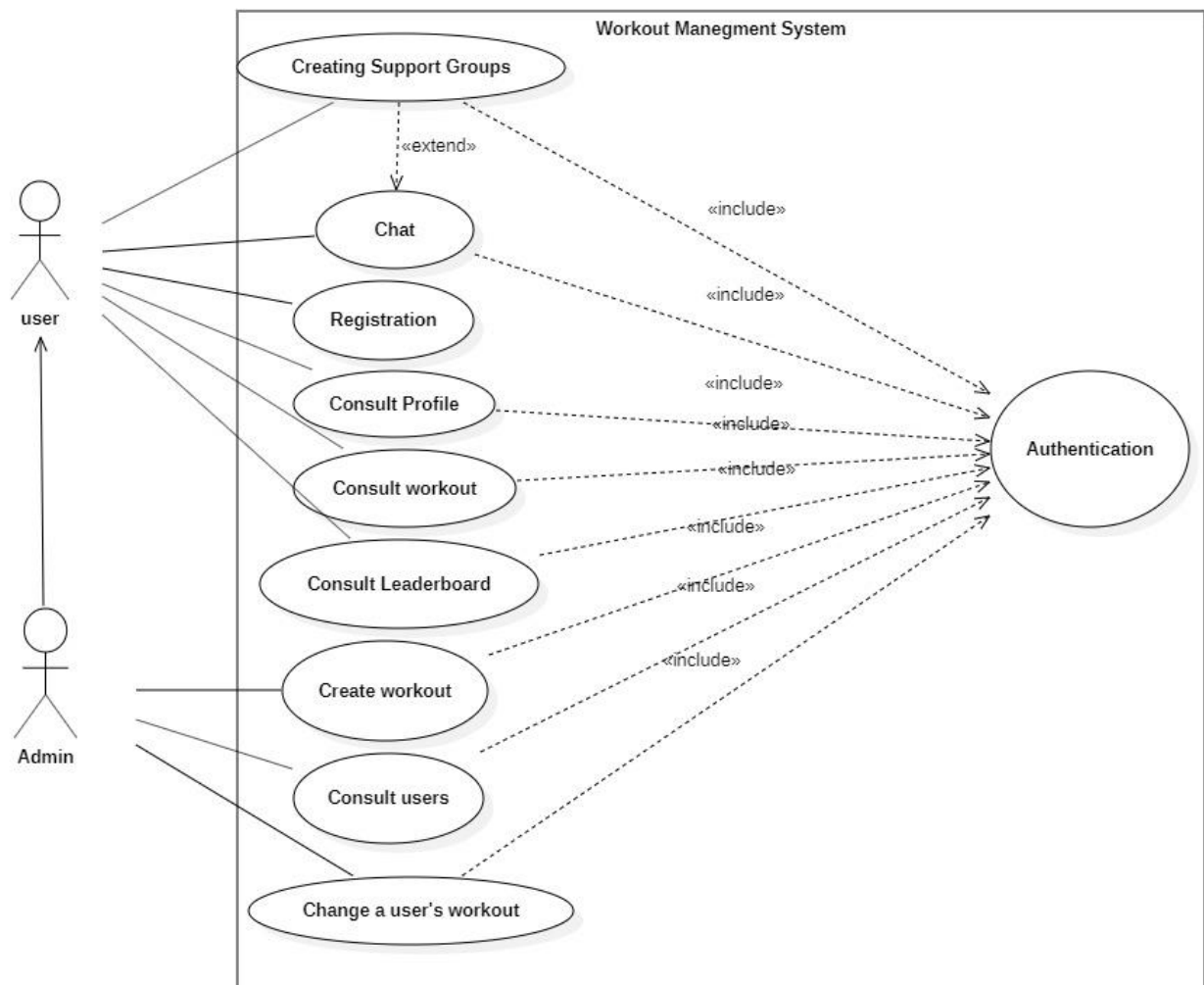


Figure 2.1: General use case diagram

2.4.2 Authentication use-case diagram:

Figure represents the use case diagram of authentication. The user will be able to have access to the application with his email and password as well as restore access to the application through Restore Password Action in case of forgotten password.

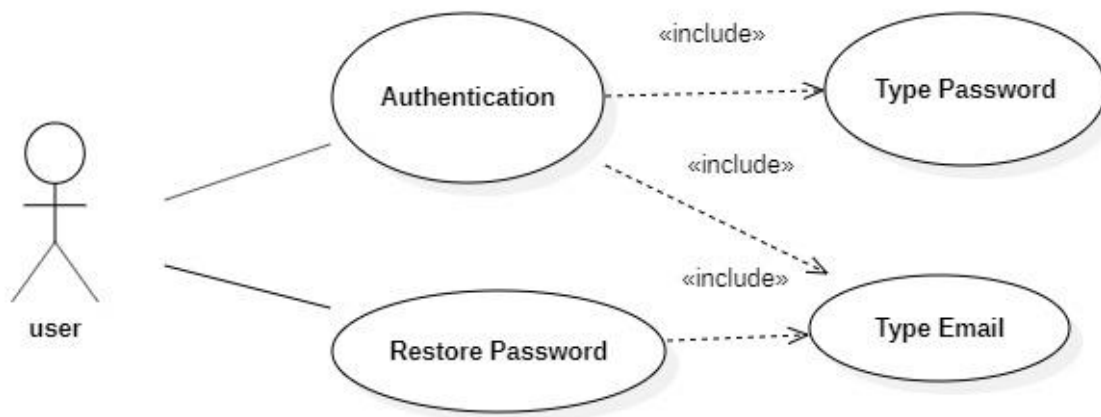


Figure 2.2: Authentication use case diagram

Table 2.1 describes in details the authentication use case scenario:

Actor	Admin, User
Objective	Get authenticated and access to the application's features
Pre-condition	Registered email and password
Post condition	Redirection to Dashboard Page
Nominal Scenario	1-User request the authentication interface 2- the application displays the interface 3-user fills in the necessary and valid fields 4-System checks the data entered and displays the dashboard Page
Alternative scenario	1-User enters incorrect data 2-System displays an error message 3-Resumption of stage 3 of the nominal scenario

Table 2.1: Authentication use case scenario

Table 2.2 describes in details the restore password scenario:

Actor	User, Admin
Objective	Restore Password
Pre-condition	Valid key
Pre-condition	The user has already registered
Post-condition	The user's password changes
Nominal Scenario	1-User requests the Forgotten password interface 2-System displays the interface 3-User fills in the necessary and valid fields 4-The system sends an email with a password reset token and displays a message success 5-User retrieves the token and use it to change his password
Alternative scenario	1-User enters an incorrect email address 2-System displays an error message 3-Resumption of stage 3 of the nominal scenario

Table 2.2: Restore password use case scenario

2.4.3 Workout Management use-case diagram

Figure 2.3 represents the use case diagram of workout management: Any user will be able to consult the workout page to see his exercises for that specific day. Admins l'll be able to create, modify or delete workouts.

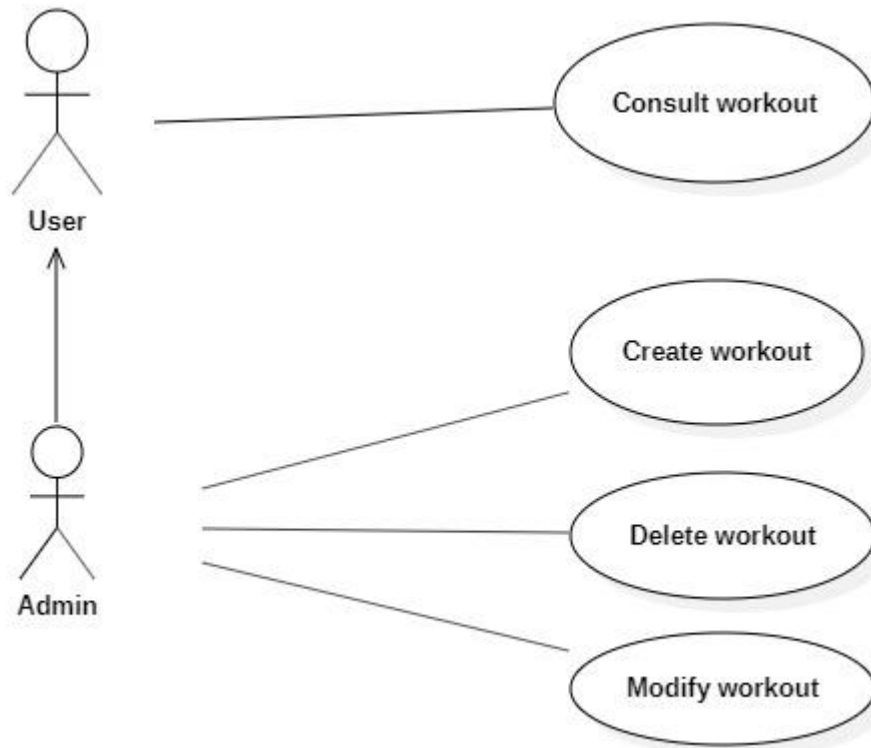


Figure 2.3: Workout Management Use-Case Diagram

Table 2.4 describes in details the consult workout scenario:

Actor	User
Objective	Consult Workout
Pre-condition	The user is authenticated
Pre-condition	The user has already chosen his goals and set his conditions
Post-condition	The user is shown the workout
Nominal Scenario	1-User requests the Workout interface 2-System displays the interface 3-User retrieves his daily workout
Alternative scenario	1-User requests the Workout 2-System displays the interface for choosing the goal 3-User sets the goal and conditions

	3-Resumption of stage 2 of the nominal scenario
--	--

Table 2.4: Consult workout use case scenario

Table 2.5 describes in details the create workout scenario:

Actor	Admin
Objective	Create a workout
Pre-condition	The admin needs to be authenticated
Post condition	The workout is saved and the admin is redirected to the interface containing the list of workouts with the new workout added
Nominal Scenario	1-User request the create workout interface 2- the application displays the interface 3-user fills in the necessary and valid fields 4-System checks the data entered 5-The system saves the workout and resets the create workout interface's fields.
Alternative scenario	1-User enters already existent workout name 2-System displays an error message 3-Resumption of stage 3 of the nominal scenario

Table2.5: Create workout use case Scenario

Table 2.6 describes in details the create workout scenario:

Actor	Admin
Objective	Modify a workout
Pre-condition	The admin needs to be authenticated
Post condition	The modification is saved and the admin is redirected to the interface containing the list of workouts with the new workout added
Nominal Scenario	1-User request the workout list interface 2- the application displays the interface 3-The user choses the workout to modify 3-user fills in the necessary and valid fields

	4-System checks the data entered 5-The system saves the modification and redirect the user to the workout list interface
Alternative scenario	1-User doesn't enter the modification 2-User enter invalid information

Table 2.6: Modify workout use case scenario

Table 2.7 describes in details the delete workout scenario:

Actor	Admin
Objective	Delete a workout
Pre-condition	The admin needs to be authenticated
Post condition	The workout is deleted and the admin is redirected to the interface containing the list of workouts without the workout that got deleted
Nominal Scenario	1-User request the create workout interface 2- the application displays the interface 3-User choose the workout to be deleted 5-The system deletes the workout and the admin is redirected to the interface containing the list of workouts without the workout that got deleted
Alternative scenario	1-User doesn't delete the workout

Table 2.7: Delete Workout use case scenario

2.4.4 Chat Management use case Diagram

Figure 2.4 represent the use case diagram of chats Management. After authentication, the user can look up and start a conversation with him as well create a group chat with multiple user which server as a support group.

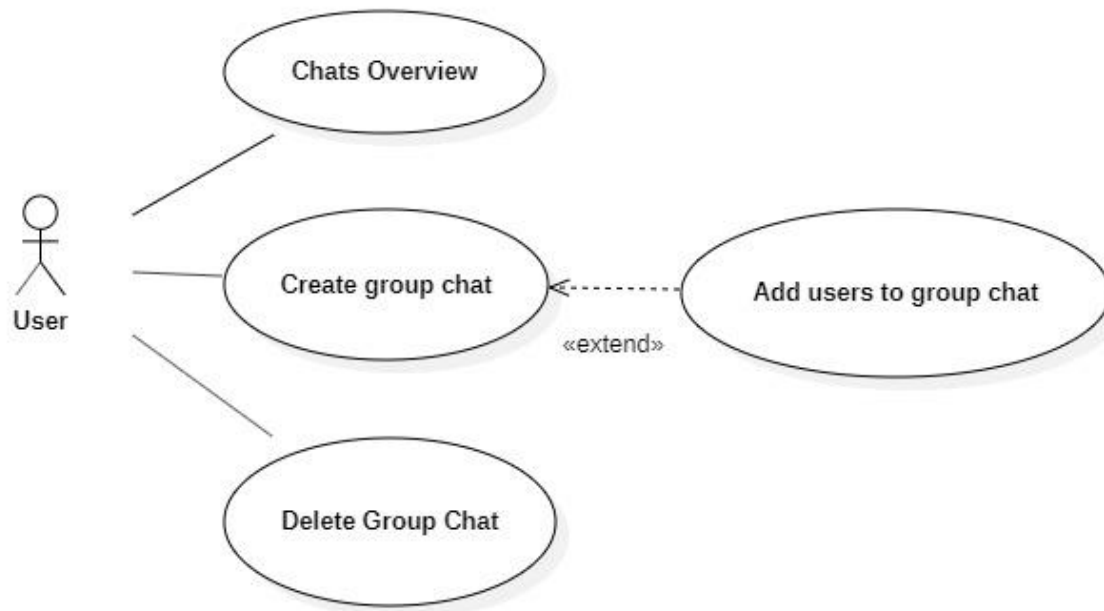


Figure 2.4: Chats Management Use Case Diagram

Table 2.8 describes in details the chats overview scenario:

Actor	User
Objective	Consult Chats
Pre-condition	The user is authenticated
Pre-condition	The user has already chosen his goals and set his conditions
Post-condition	The user is shown the list of chats
Nominal Scenario	1-User requests the chats interface 2-System displays the interface
Alternative scenario	1-User didn't start any chat 2-The user is redirected to an empty chats interface

Table 2.8: Consult Chats Use Case Scenario

Table 2.9 describes in details the Create Group Chat Scenario:

Actor	User, Admin
Objective	Start a chat
Pre-condition	The user needs to be authenticated
Post condition	The user can send messages to the other user in the chat
Nominal Scenario	1-User request the chats interface 2- the application displays the interface 3-user fills in the name of the user 4-System shows users with relative name 5-User choose who he wants to start a conversation with 5-The system create the conversation and the user is redirected to the conversation interface
Alternative scenario	1-User enters a wrong username 2-System doesn't display any users 3-User enters the correct username 3-Resumption of stage 4 of the nominal scenario

Table2.9: Create workout use case Scenario

Table 2.9 describes in details the create group chat scenario:

Actor	Admin
Objective	Create a group chat
Pre-condition	The user needs to be authenticated
Post condition	The group chat is saved and the users within it can chat with each other
Nominal Scenario	1-User request the group chat interface 2- the application displays the interface 3-user fills in the group chat name field 4-System creates the chat group
Alternative scenario	1-User enters already existent group chat name

Table2.9: Create workout use case Scenario

Table 2.91 describes in details the add users to group chat scenario:

Actor	Admin
Objective	Add users to group chat
Pre-condition	The user needs to be authenticated
Pre-condition	The group chat is already created
Post condition	The users are added to the group chat
Nominal Scenario	1-User chooses the group chat 2- The user requests the add users' interface 3-System displays the add users interface 4-Users enter the name of the users to add 5-The users are added to the group chat
Alternative scenario	1-User enters inexistant user name 2-System doesn't show any user 3-Resumption of stage 4 of the nominal scenario

Table2.9: Create workout use case Scenario

Conclusion

Throughout this chapter, we presented the functional and non-functional needs related to our application, as well as the main functionalities such as workout and chat management, and in the end, we presented the different use-case diagrams and scenarios of those functionalities.

Chapter 3 :Design and Architecture

3.1 Introduction

The design and Architecture are two of the most important building blocks because proper project understanding is necessary from the outset. In this chapter, we will elaborate the global architecture of the project and present conceptual aspects by providing different diagrams such as sequence and class diagrams.

3.2 Design Pattern

Design patterns are solutions to general problems that software developers faced during software development. These solutions were obtained by trial and error by numerous software developers over quite a substantial period of time

3.2.1 MVC MERN Stack

MVC is a design pattern stands for Model View Controller that consists on the separation of the application into three Components (Layers) which aims to isolating the logic of the application from the presentation layer, as well as prohibiting direct access to the data stored by these layers.

In our project we used the MERN stack, which stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack.

The structure of our project is described as follows:

Model

The central component of the pattern. It is the application's dynamic data structure, independent of the user interface.^[5] It directly manages the data, logic and rules of the application.

In our project, it's the MongoDB modals that represent how data is stored in the mongoDB database.

View

The view is a visual representation of the data- like a chart, diagram, table, form.

The view contains all functionality that directly interacts with the user - like clicking a button, or an enter event.

In our project, the presentation part is implemented in the React.js in the format of components.

Controller.

The controller connects the model and view. The controller converts inputs from the view to demands to retrieve/update data in the model.

The controller receives input from view, uses logic to translate the input to a demand for the model, the model grabs the data, the controller passes data from the model back to the view for the user to see in a nice display.

In our project it's a JavaScript file the contains the needed logic to manipulate the data coming from the database and to handle requests given by the Express.js router.

See figure 3.1,3.2 and 3.3

```
JS workoutControllers.js M X
controllers > JS workoutControllers.js > [0] getWorkoutById > Workout.findOne() callback
1 import Workout from "../models/Workout";
2 export const getAllWorkouts = () => {
3   Workout.find({}, (err, Workout) => {
4     if (err) {
5       res.send(err);
6     }
7     res.json(Workout);
8   });
9 };
10
11 export const getWorkoutById = (ref) => {
12   Workout.findOne({ Workoutref: ref }, (err, workout) => {
13     if (err) {
14       res.send(err);
15     }
16     res.json(workout);
17   });
18 };
19
20 export const addNewWorkout = (workout) => {
21   const { name, days, Description } = workout;
22 }
```

Figure 3.1: The Controller

```
JS Workout.js M X
client > src > components > UI > Content_pages > JS Workout.js > Workout
1 import React, { useEffect, useState } from "react";
2 import "../css/Workout.css";
3 import Task from "../Task";
4 import ReactPlayer from "react-player";
5 import { useSelector } from "react-redux";
6
7 function Workout(props) {
8   const state = useSelector((state) => state);
9   const workoutName = state.auth.person.workout;
10  const day = state.auth.person.day;
11  const workoutData = state.workout.payload.daysDB;
12  const currentWorkout = workoutData[day];
13  console.log("i am the data :", workoutData);
14  const [paragraph, setParagraph] = useState("");
15  const [everything, setEverything] = useState({});
16
17
18  return (
19    <div>
20      <h1>Welcom to day {props.day}</h1>
21      <div className="second">
22        <p className="paragraph">{paragraph}</p>
23        <p>{ press on workout name to reveal the corresponding video}</p>
24        <ul className="work">
25          {currentWorkout.map((element) => (
26            <li
27              className="work1"
28              onClick={() => {
29                setCurrentLink(element.Link);
30              }}
31            >
32          )}
33        </ul>
34      </div>
35    </div>
36  );
37 }
```

Figure 3.2: The View

```

JS Workout.js X
models > JS Workout.js > ...
1  const mongoose = require("mongoose");
2
3  const Schema = mongoose.Schema;
4
5  const workoutSchema = new Schema({
6    WorkoutName: {
7      type: String,
8      required: true,
9    },
10   Description: {
11     type: String,
12     required: true,
13   },
14   daysDB: {
15     type: Array,
16     required: true,
17   },
18   Workoutref: {
19     type: String,
20     default: null,
21   },
22 });
23
24 module.exports = Workout = mongoose.model("Workout", workoutSchema);
25

```

Figure 3.3:The model

3.2.2 MVC Flow

The user interacts with the UI, and the controller gets notified via the view. Based on the User interaction the controller modifies certain Models. Models perform some business logic and return the updated model data state to the controller. The controller can then update the UI according to the new data state as received from Model.

The figure 3.1 describes MVC flow:

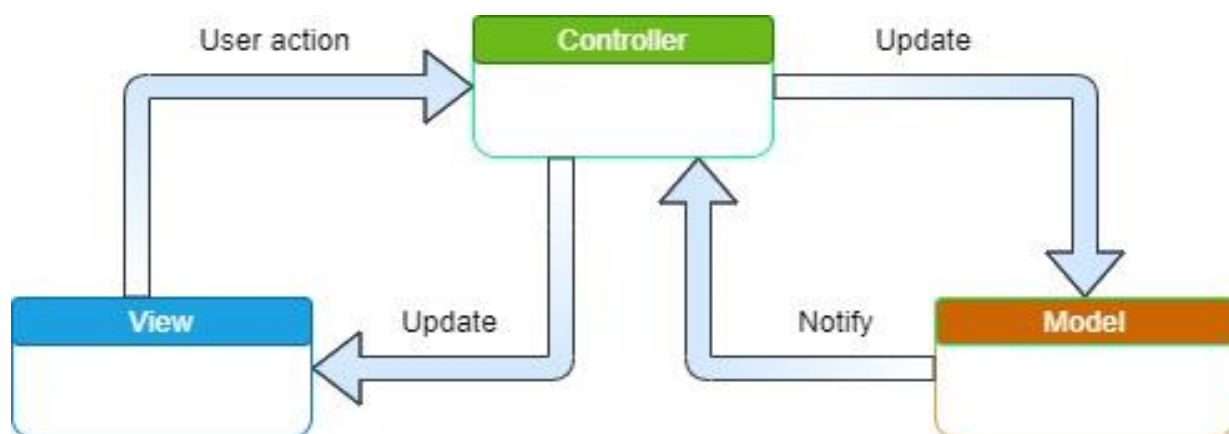


Figure 3.4:MVC flow

3.3Physical Architecture

During the development of our project, we chose the three-tier architecture since it's a well-established software application architecture that organizes application into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, and the data tier (see figure 3.2)

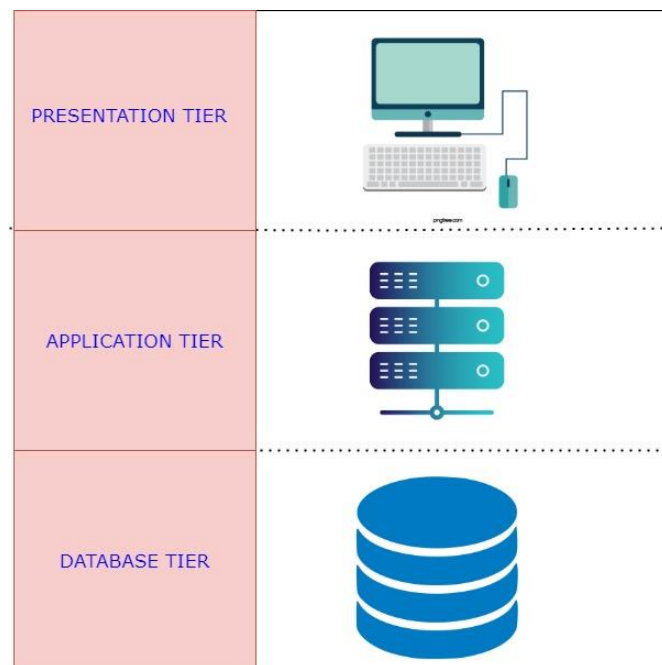


Figure 3.5:Physical Architecture

Client Tier: it's the user interface and communication layer of the application, where the end user interacts with it. Its main purpose is to display information to and collect information from the user.

Application Tier: also known as the logic tier or middle tier, is the heart of the application. In this tier, information collected in the presentation tier is processed-sometimes against other information in the data tier-using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier.

Data Tier:

Sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed.

Benefits of three-tier architecture:

-Separation: the chief benefit of three-tier architecture is that because each tier runs on its own infrastructure, each tier can be developed simultaneously by separate development team, and can be updated or scaled as needed without impacting the other tiers.

-Faster development: Because each tier can be developed simultaneously by different teams, an organization can bring the application to market faster, and programmers can use the latest and best languages and tools for each tier.

Improved scalability: Any tier can be scaled independently of the others as needed.

Improved reliability: An outage in one tier is less likely to impact the availability or performance of the other tiers.

Improved security: Because the presentation tier and data tier can't communicate directly, a well-designed application tier can function as a sort of internal firewall, preventing SQL injections and other malicious exploits.

3.4 Deployment Diagram

A deployment diagram in the Unified Modeling Language (UML) models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components exist, what software components run on each node, and how the different pieces are connected (See figure 3.3)

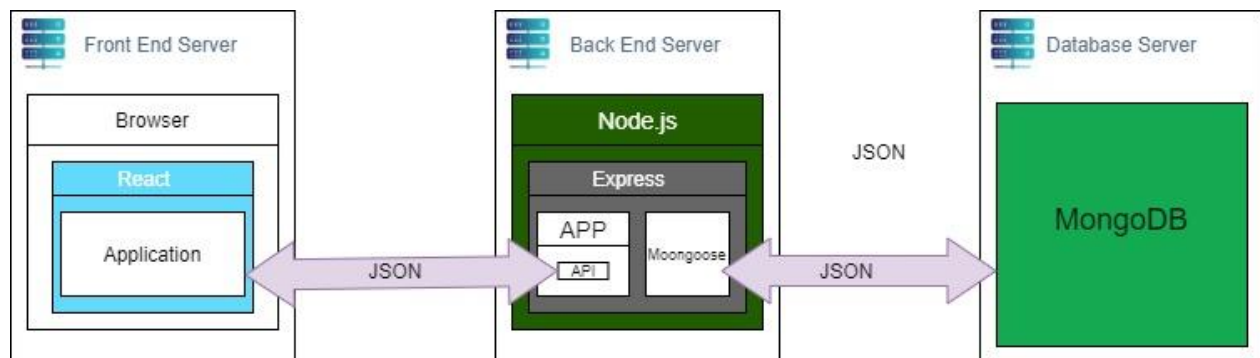


Figure 3.6:Deployment Diagram

3.5 Logical Architecture

Choosing an architecture for an application is a paramount step in the process of making the application before moving on to project design.

Our Web application is composed of 3 parts the Front-End, the Back-End and the Database. The Front-End is realized with React.js Framework combined with Redux library (React-Redux) for state management and Axios which uses the Rest APIs implemented in the Back-End in which we used Express.js.

Figure 3.4 represent our application's software architecture:

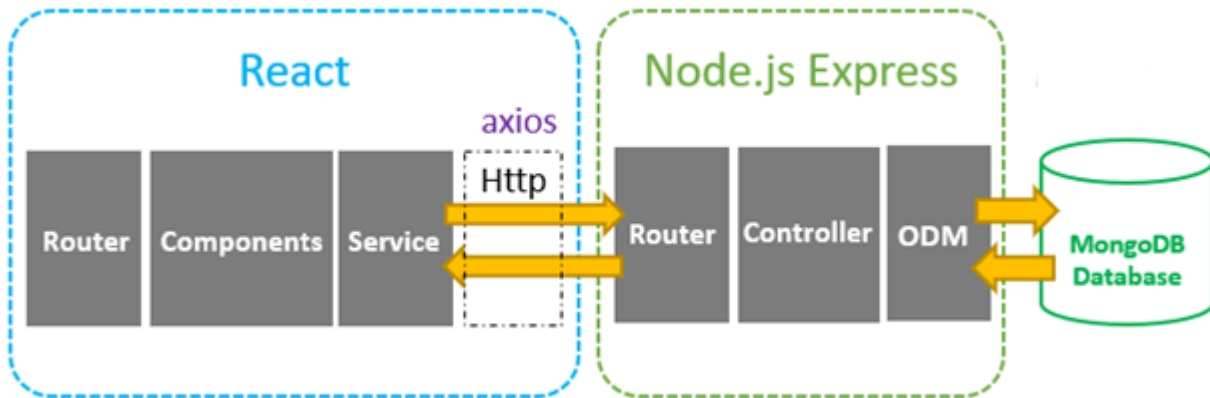


Figure 3.7: Logical Architecture Diagram

Express communicates with the database MongoDB (NoSQL database) using the mongoose. Everything in the database has a representation in the form of manipulable objects called Models to simplify access and operations performed on the database and this representation is done by mongoose ODM

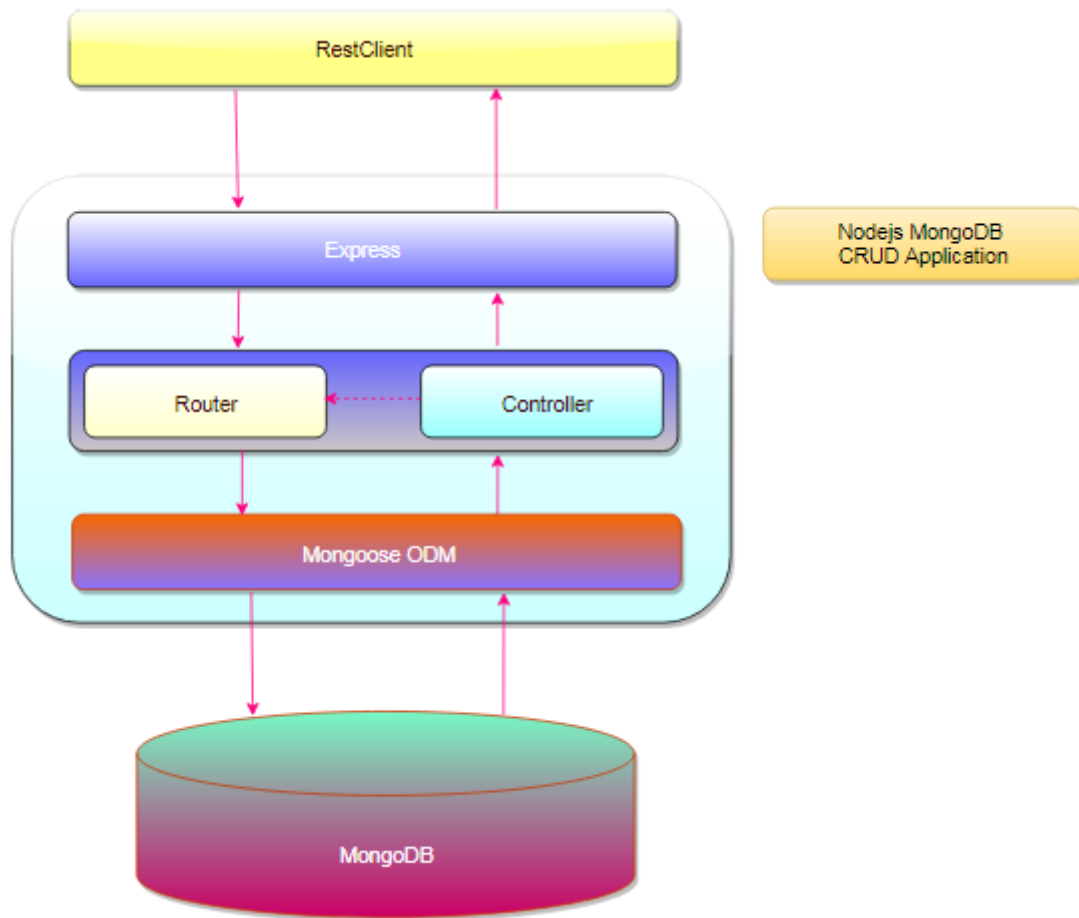


Figure 3.8:BackEnd diagram

In this way, every collection in the MongoDB is represented by a modal and the express controller contains the logic concerning the actions performed by the user that affect those collections.

For the Front-End, we used Redux to define how our component's state was managed.

Redux is a library that acts as a state container and helps to manage the application data flow.it provides a central place for storing data that is used across the React application.

The concepts of Redux is better explained in the figure 3.6:

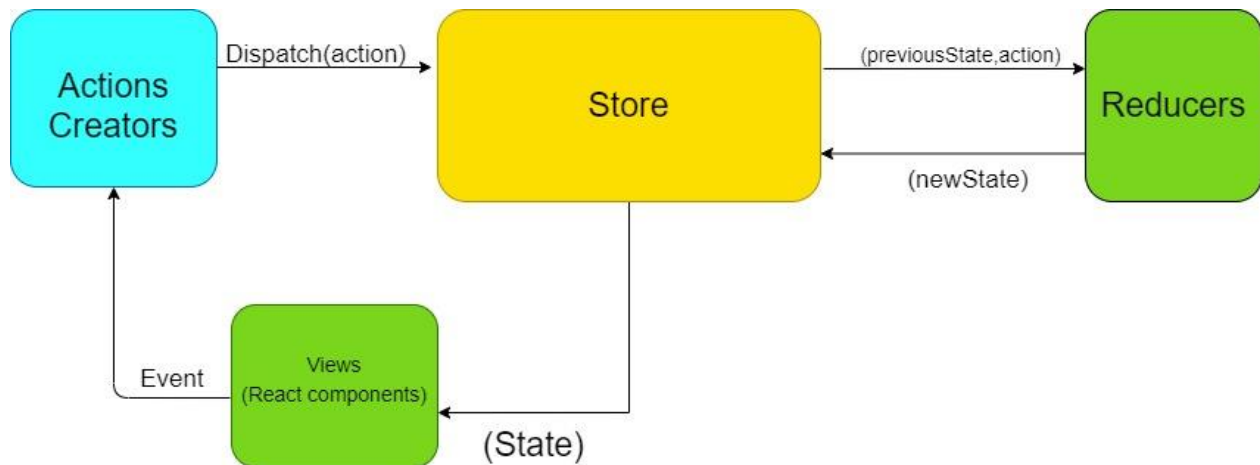


Figure 3.9: Redux diagram

Actions: they are JavaScript objects that contain to properties “type” and “payload”.

Actions Creators: they are functions that dispatch actions after a react component fires an even.

-Store: it’s an immutable object tree in Redux. A store is a state container which holds the application’s state.

-Views: The views are the React component which can be created to depend to the state that the store holds.

-Reducers: it’s a pure function that takes an action and the previous state of the application and returns the new state. The action describes what happened and it is the reducer’s job to return the new state based on that action

In our project we created a chat service through which the user can communicate with other users, we used the mongoDB to save the messages, and we used socket.io, which means when a user receives a message , his conversation refreshes automatically and sees the new message without having to refresh the page.

The socket.io communicated the react directly without passing through the Express.js server.

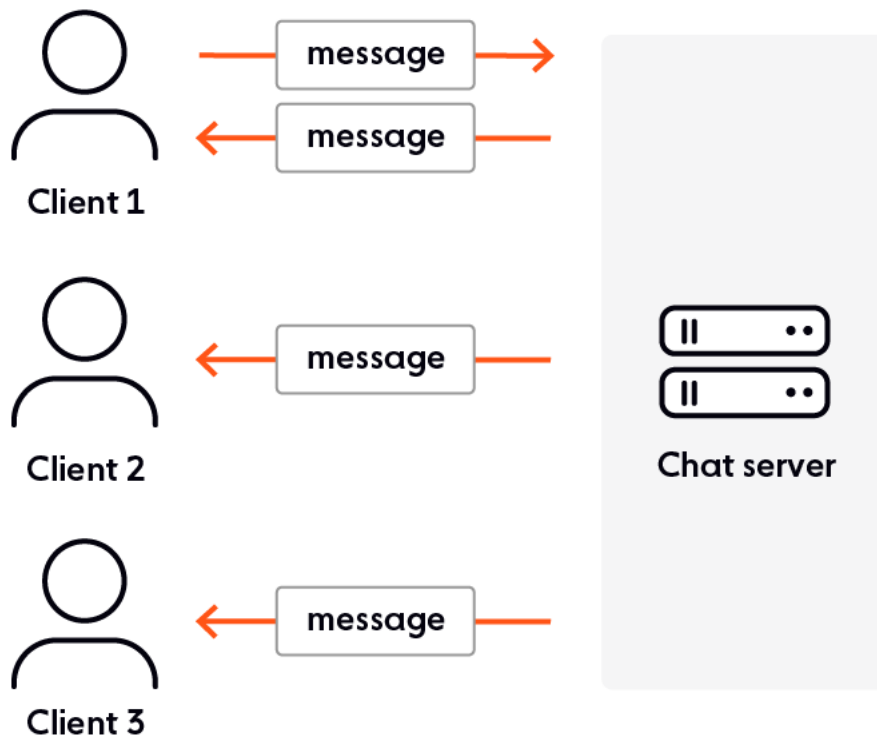


Figure 3.10: Socket.io Diagram

3.6 Sequence diagrams

In this section, we will present the dynamic views of the system using the sequence diagrams for the most important functionalities in our application.

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

3.5.1 Authentication Sequence Diagram

This Authentication Sequence Diagram describes how the user is able to connect to the application and exploit its features. The authentication is successful only if the user had already registered and that he entered the right credentials.

If the user hasn't registered or that he entered incorrect data, he gets notified by an error message. Otherwise, the system verifies his credentials in the database and then returns a token containing his information which will be saved in local storage and then the user is redirected to the dashboard.

Figure 3.6 illustrated the sequence diagram of the Authentication scenario:

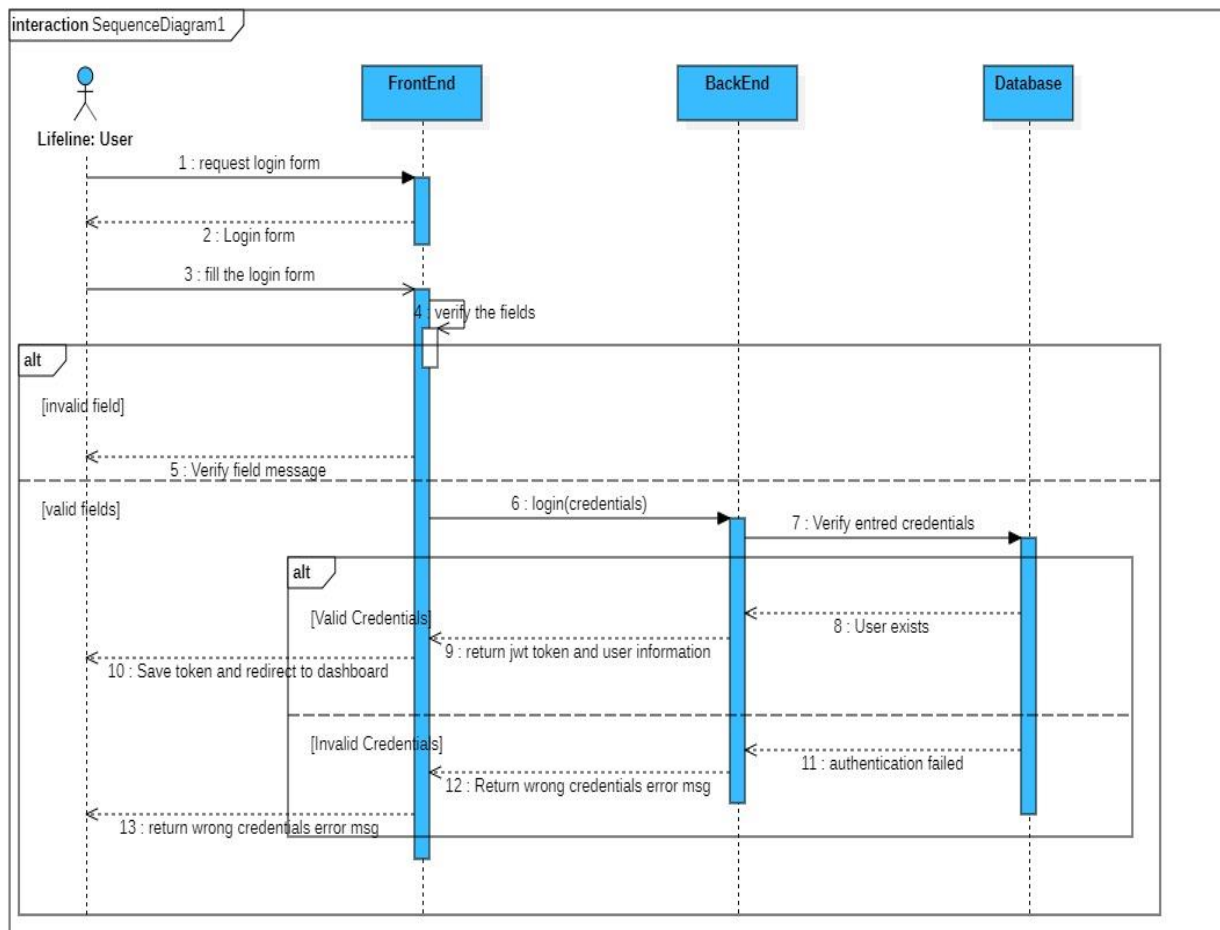


Figure 3.10: Authentication Sequence Diagram

3.5.1 User Management Sequence Diagram

The use case user management allows the admin to consult the list of users as well as promote any user to admin or delete him. If these operations are successful, the system displays a success message.

If there's no users, the system displays an empty list.

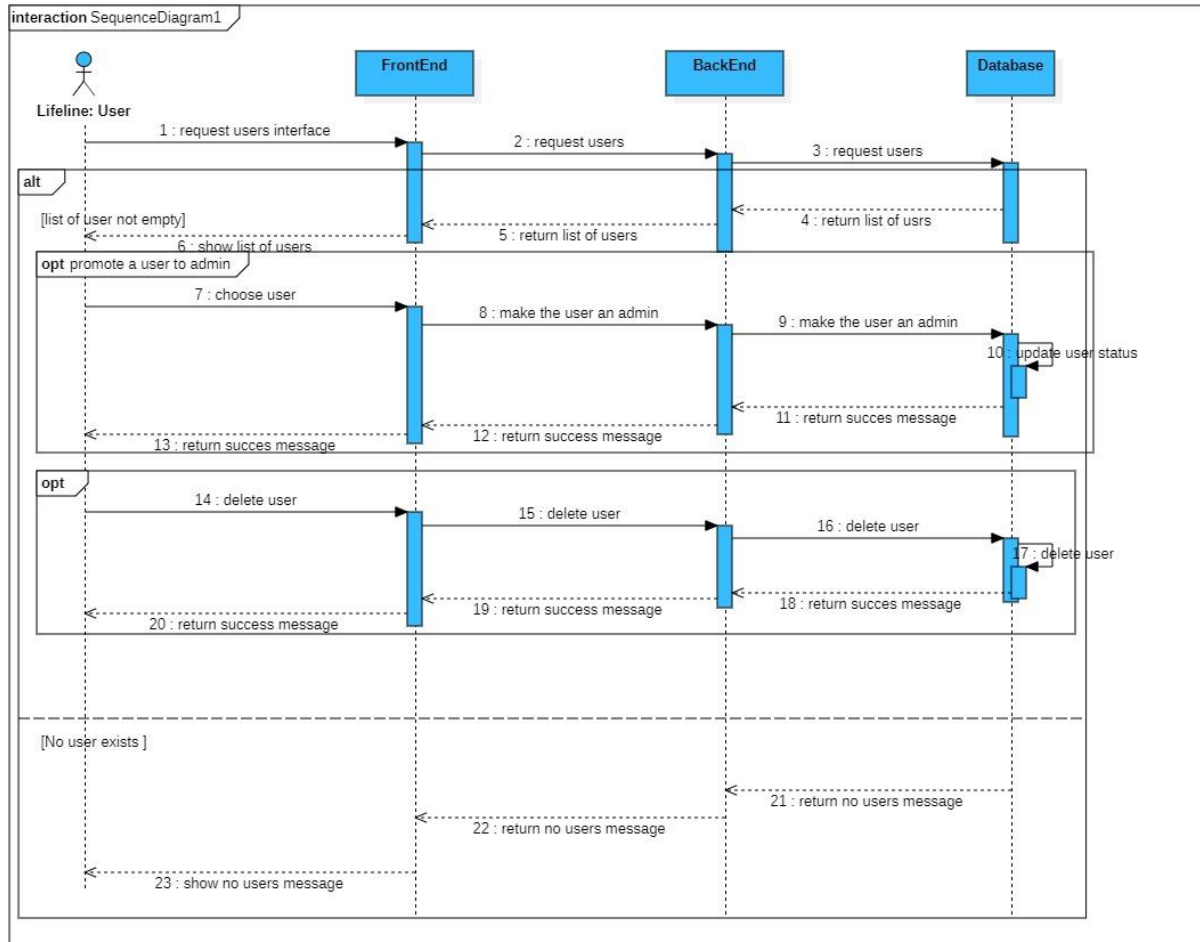


Figure 3.11:User Management Sequence Diagram

3.5.1 New Chat Sequence Diagram

The new chat use case allows a user to start a new conversation with any other user. The operation is only carried out if both of the users already registered.

If the user inserts a user name that doesn't exist, the system returns an error message to check the name field, otherwise it creates a new conversation and redirects the user to that conversation.

Figure 3.7 represents the sequence diagram of the New Chat scenario:

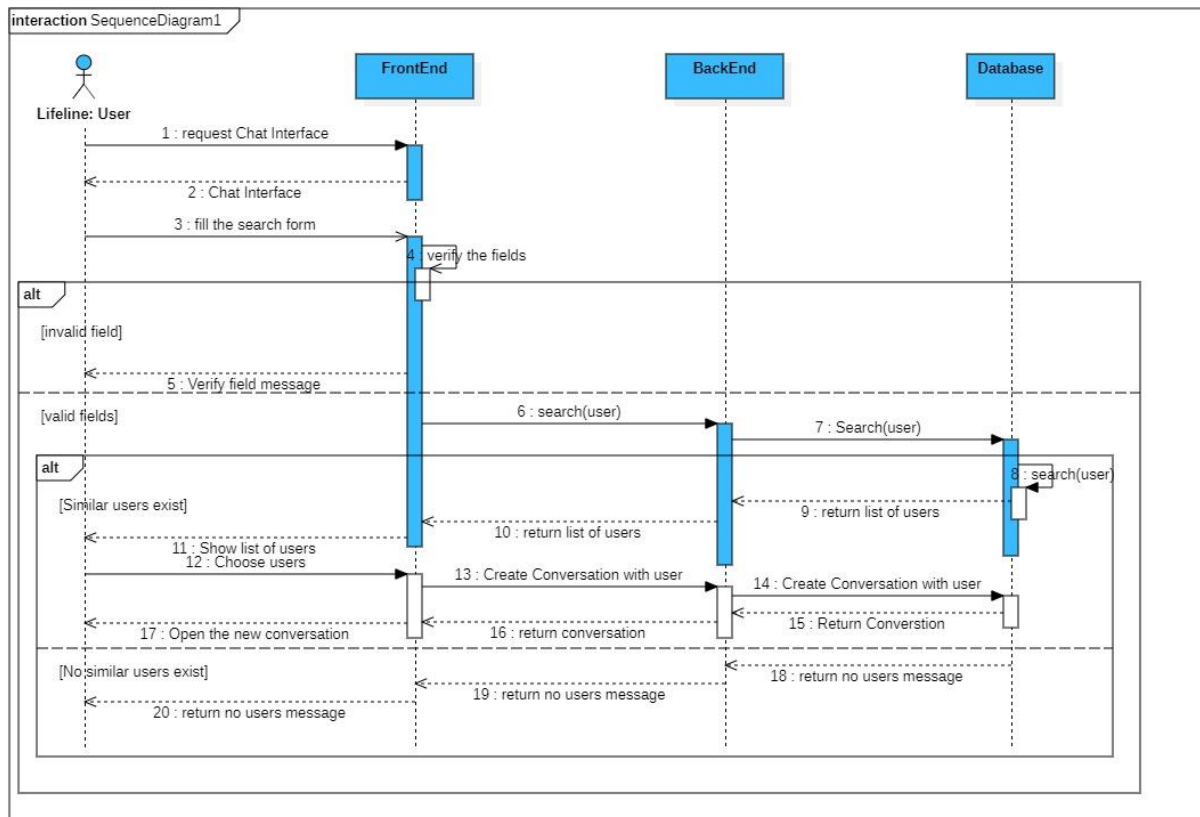


Figure 3.12:Create New Conversation Sequence Diagram

3.6 General Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling.

Since our project follows the MVC design pattern, we shaped the class diagram accordingly.

Figure 3.8 represents the general class diagram of the application:

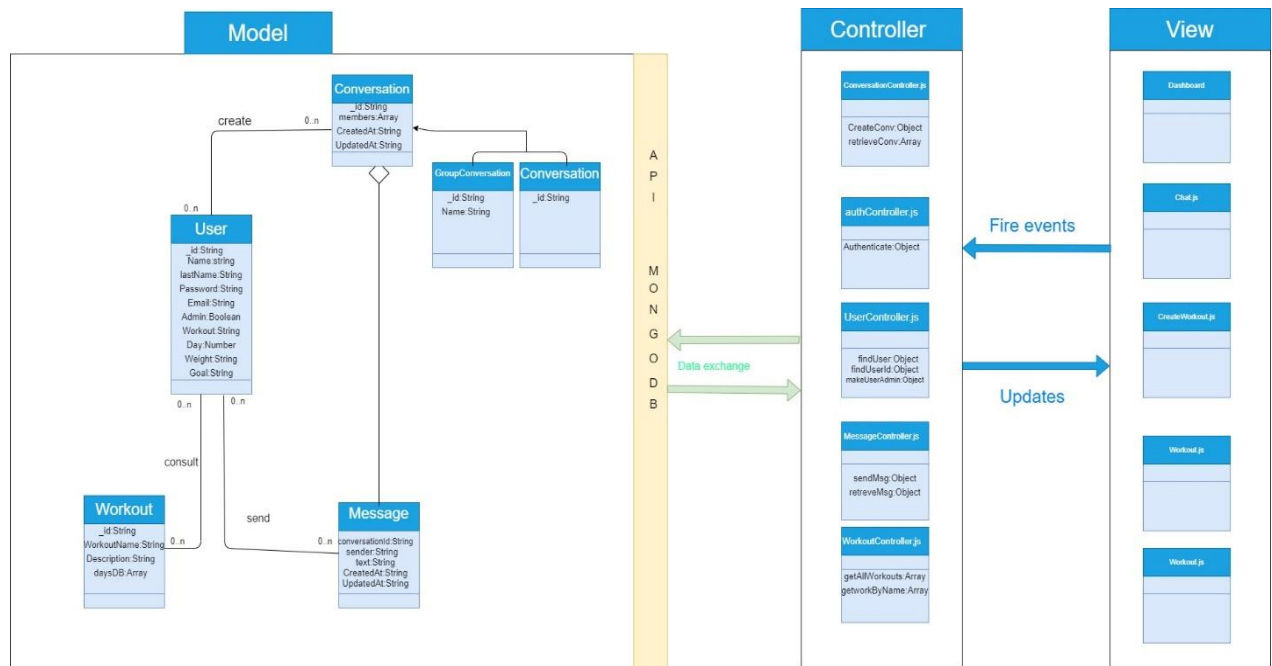


Figure 3.13:: General Class Diagram

3.7 Gantt Diagram

The Gantt diagram is commonly used in the management of a project. it allows to represent the state of progress over time of the different activities (sprints in our case) constituting a project visually. The column on the left lists the tasks to be performed, while the columns on the right represent the time units. Each task is materialized by a horizontal bar, whose position and length represent the start date, the duration and the end date.

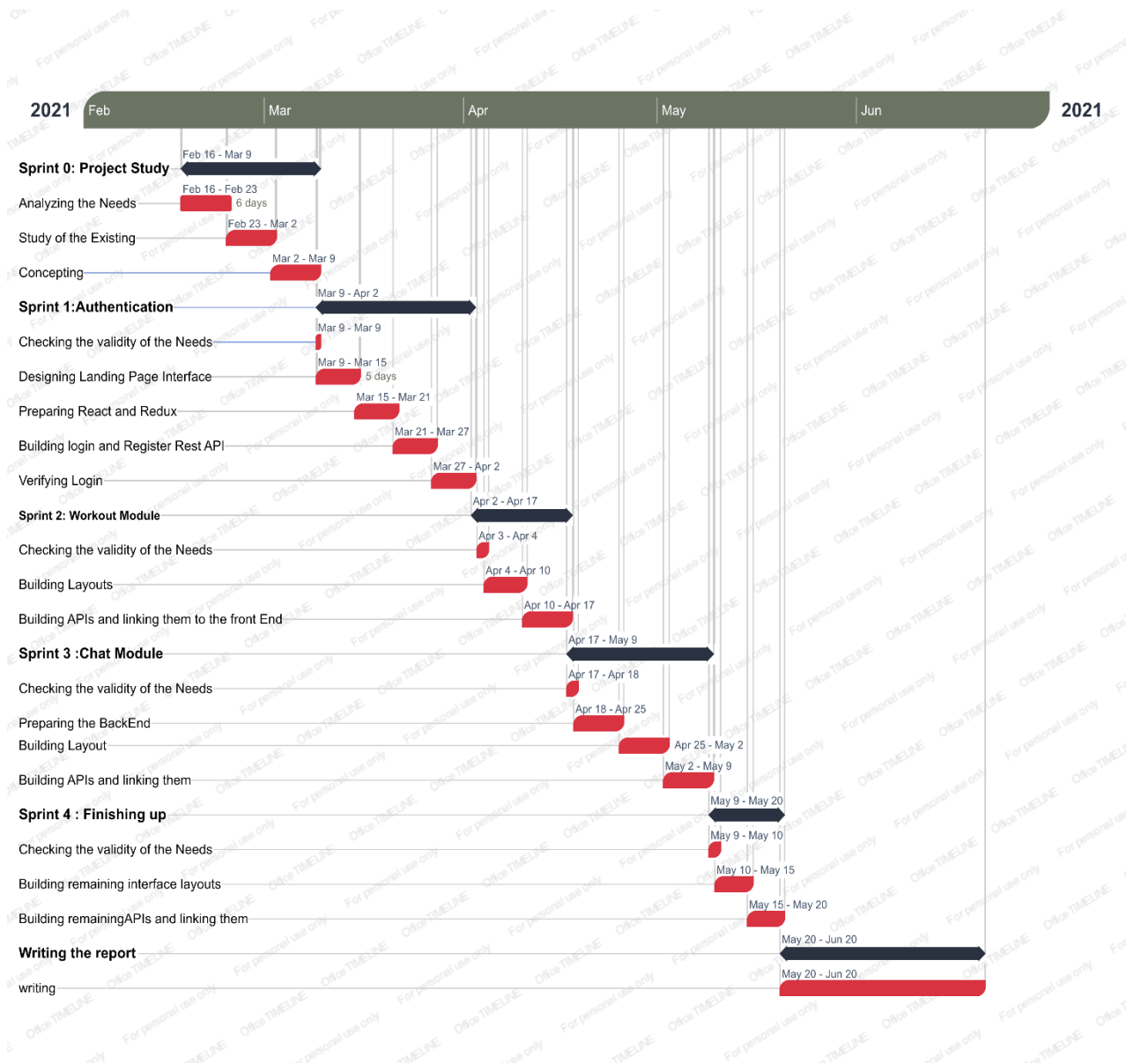


Figure 3.14:Gantt Diagram

3.8 Conclusion

In this Chapter we presented the design pattern that we followed, as well as the physical and logical architecture, then we moved on to present the different sequence diagram, followed by the general class diagram and the Gantt diagram.

Chapter 4 : Realization

4.1 Introduction

In every craftsmanship, the quality of the tools plays a decisive role in the quality of the product, and it's the same in software development. That's why the choice of programming tools needs to be made after an in-depth study of these tools.

After presenting the conceptual aspects of our project in the last chapter, we move on to presenting the realization of our project in this one after we present the tools that helped us bring our idea to life.

4.2 Working environment and tools

In this section we will present the software and hardware environment

4.2.1 Material Environment

During this project, we used a laptop characterized by the following:

- OS: Windows 10
- CPU: 1.70GHz Intel Core i5 four cores
- RAM: 8Go
- Hard Disk: 1000GO HDD.

Device specifications

Device name	DESKTOP-B7T17TR
Processor	Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40 GHz
Installed RAM	8.00 GB
Device ID	DBF733B2-A2BF-4BB2-A51D-CD93E388C482
Product ID	00330-80000-00000-AA681
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 4.1: Computer characteristics

4.2.2 Software Environment

In this section, I will present the technologies chosen for my project:

Front-End

React.js is an open-source, front end JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies.

Component-Based

React allow its users to build encapsulated components that manage their own state, then compose them to make complex UIs.

Declarative

React makes it painless to create interactive UIs. After designing simple views for each state in the application, React will efficiently update and render just the right components when the data changes

Write once, Use everywhere

React components are easily re-usable in different applications.

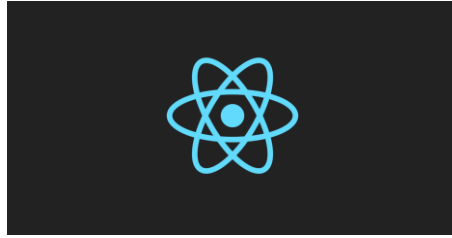


Figure 4.2:ReactJS logo

Redux

Redux is an open-source JavaScript library for managing application state. It is most commonly used with libraries such as React or Angular for building user interfaces. Similar to (and inspired by) Facebook's Flux architecture, it was created by Dan Abramov and Andrew Clark.



Figure 4.3:Redux Logo

Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

As of April 2021, Bootstrap is the tenth most starred project on GitHub



Figure 4.4:Bootstrap Logo

Reactstrap

Reactstrap is a component library for reactjs. It provides inbuilt Bootstrap components that make easy to create UI with its self-contained components that's provide flexibility and inbuilt validations. Reactstrap is similar to Bootstrap, but it has self-contained components. It's easy to use and support Bootstrap 4.

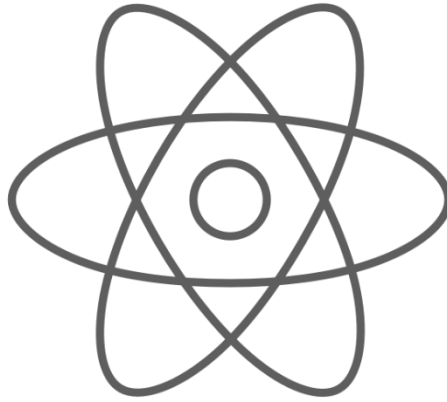


Figure 4.5:Reactstrap Logo

Back-End

Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side Scripting-running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

Node.js represents a “JavaScript everywhere ‘paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.



Figure 4.6:Node.js Logo

Express.js

Express.js, or simply Express, is a back end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.[3] It has been called the de facto standard server framework for Node.js.[4]



Figure 4.7: Express.js logo

MongoDB

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database programming uses JSON-like documents with optional Schemas



Figure 4.8: MongoDB logo

Socket.io

Socket.IO is a JavaScript library for realtime web applications. It enables realtime, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js. Both components have a nearly identical API. Like Node.js, it is event-driven.



Figure 4.9: Socket.io Logo

Visual Studio Code

Visual Studio Code is a source-code editor made by Microsoft^[9] for Windows, Linux and macOS.^[9] Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.



Figure 4.10: Visual Studio Code Logo

StarUml

StarUML is a UML tool by MKLab. The software was licensed under a modified version of GNU GPL until 2014, when a rewritten version 2.0.0 was released for beta testing under a proprietary license. After being abandoned for some time, the project had a revival to move from Delphi to Java/Eclipse and then stopped again.



Figure 4.11: StarUML Logo

Draw.io

diagrams.net (formerly **draw.io**) is free online diagram software. You can use it as a flowchart maker, network diagram software, to create UML online, as an ER diagram

tool, to design database schema, to build BPMN online, as a circuit diagram maker, and more. **draw.io** can import. vsdx, Gliffy™ and Lucidchart™ files .



Figure 4.12:draw.io Logo

Git

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems)



Figure 4.13:Git Logo

Postman

Postman is an interactive and automatic tool for verifying the APIs of your project. **Postman** is a Desktop application for interacting with HTTP APIs. It presents you with a friendly GUI for constructing requests and reading responses. It **works** on the backend, and makes sure that each API **is working** as intended



Figure 4.14:Postman Logo

Microsoft Word

Microsoft Word is a [word processor](#) developed by [Microsoft](#). It was first released on October 25, 1983,^[6] under the name *Multi-Tool Word* for [Xenix](#) systems.^{[7][8][9]} Subsequent versions were later written for several other platforms .



Figure 4.15:Microsoft Word Logo

4.3 Implementation

In this section we will present the accomplished work in our application by presenting screenshots of the main interfaces.

4.3.1 Home Interface

The home interface gives a general presentation of the application and its content and features. The figure 4.15 represents the home page:

Team work Process

You set the goal, we take care of the rest

[Sign up](#)



Figure 4.16:Home page

4.3.2 Registration interface

The registration interface allows the user to create an account in order to gain access to its features. Figure 3.16 represents the registration interface.

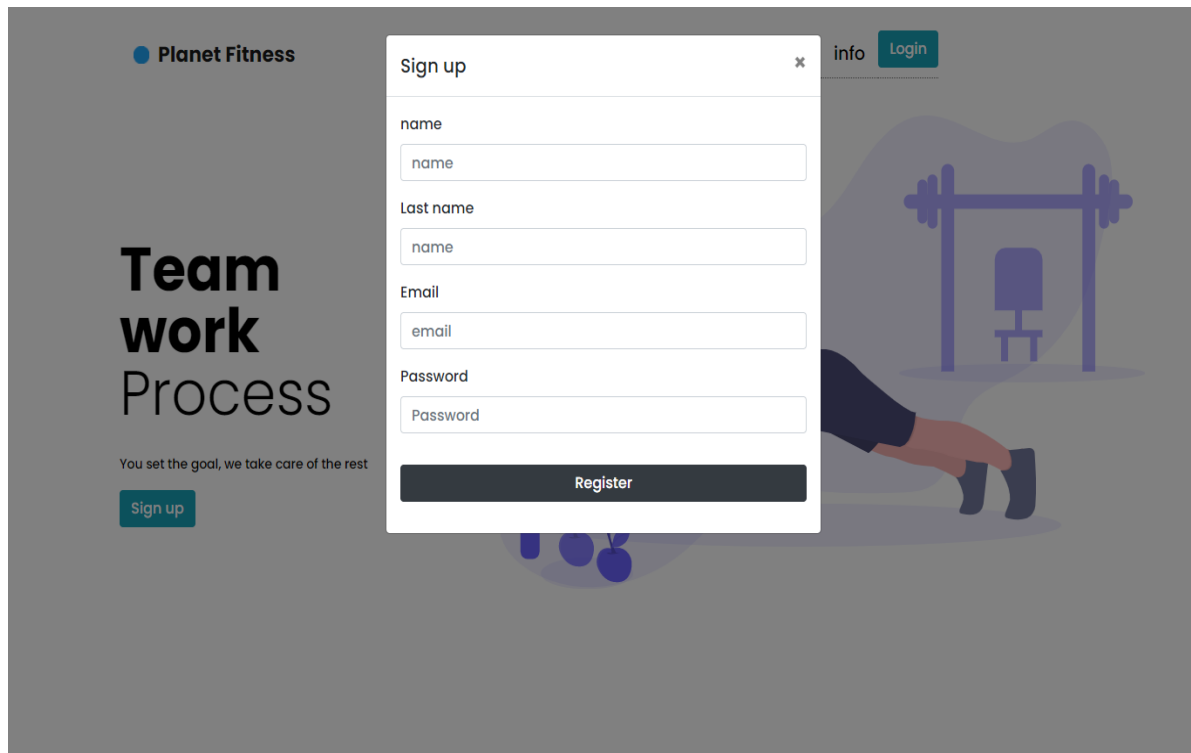


Figure 4.17:Registration Interface

4.3.3 Login Interface

The login interface allows the user to gain access to the system after a successful validation of his credentials. Figure 4.17 represents the login interface.

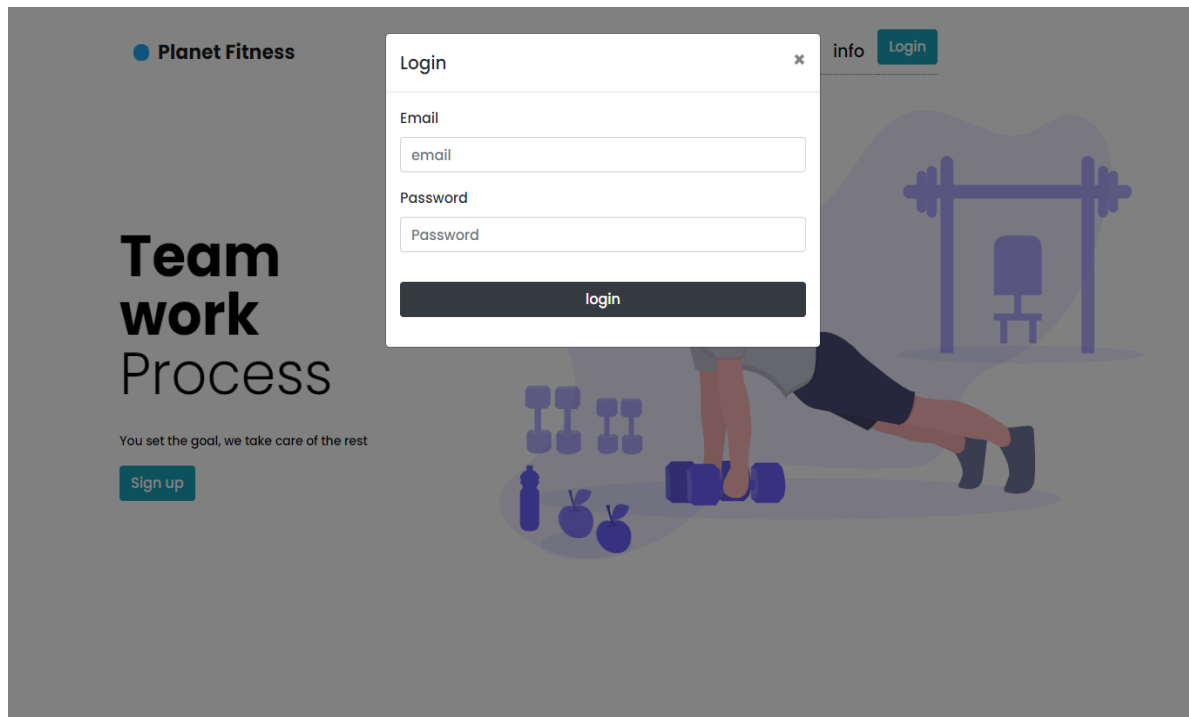


Figure 4.18:Login interface

4.3.4 Admin Dashboard interface

Figure 4.18 represents the admin dashboard which gives the admin more the functionalities than the normal user:

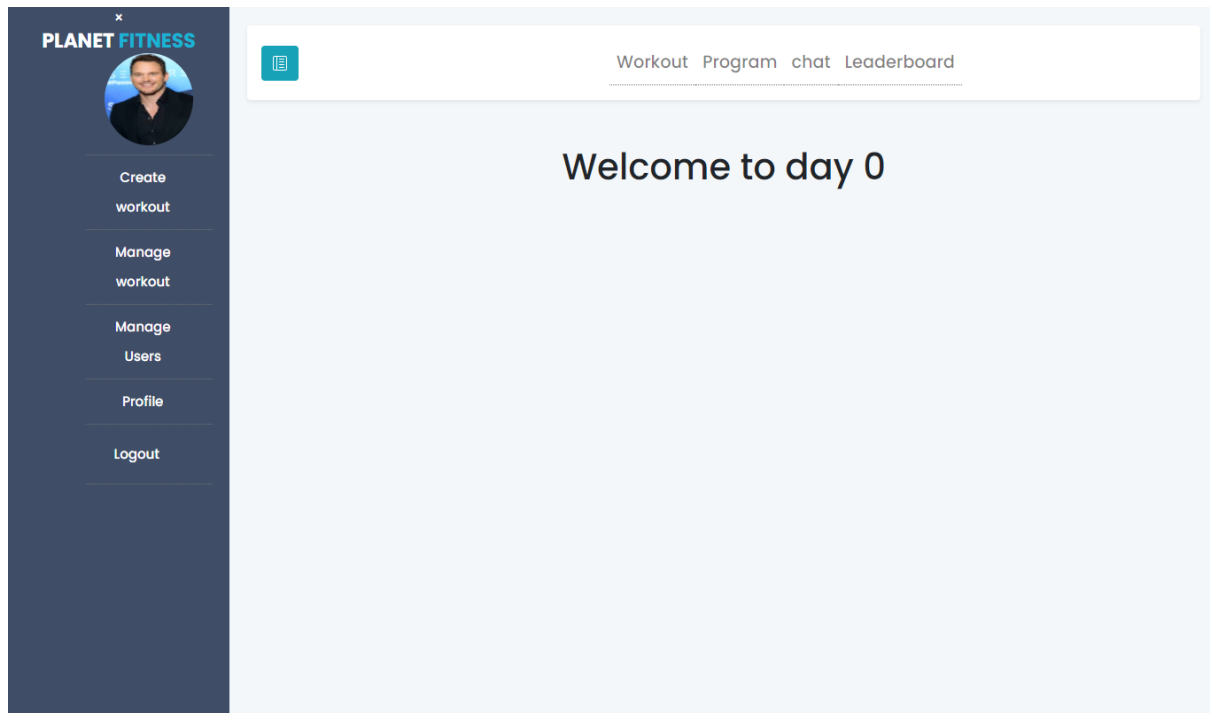



Figure 4.19:Admin Dashboard

4.3.5 Create workout interface

the create workout interface permits the user to fill the needed fields, after that he can press the submit button so the workout is saved in the database. Figure 4.19 and figure 4.20 represent the create workout interface.

PLANET FITNESS



Create workout

Manage workout

Manage Users

Profile

Logout

Workout

Program

chat

Leaderboard

Create Workout

Workout Name

Hard Abs

Duration

10

Workout Reference

11121


Workout description

This workout is for hard workers, high intensity 6 days a week

day 1

Figure 4.20:Create workout 1

PLANET FITNESS



Create workout

Manage workout

Manage Users

Profile

Logout

Workout

Program

chat

Leaderboard

day 1

Push up	5	20	Video Link	Save
Chin up	10	5	Video Link	Save
Add	Save			

day 2

plank	1	30	Video Link	Save
pull up	3	10	Video Link	Save
Add	Save			

day 3

Exercise name	Sets	Reps	Video Link	Save
---------------	------	------	------------	------

Figure 4.21:Create Workout 2

4.3.6 Manage Workout Interface

Figure 4.3.6 represents the Manage Workout Interface which allows the admin the consult all the workouts and delete anyone he chooses.

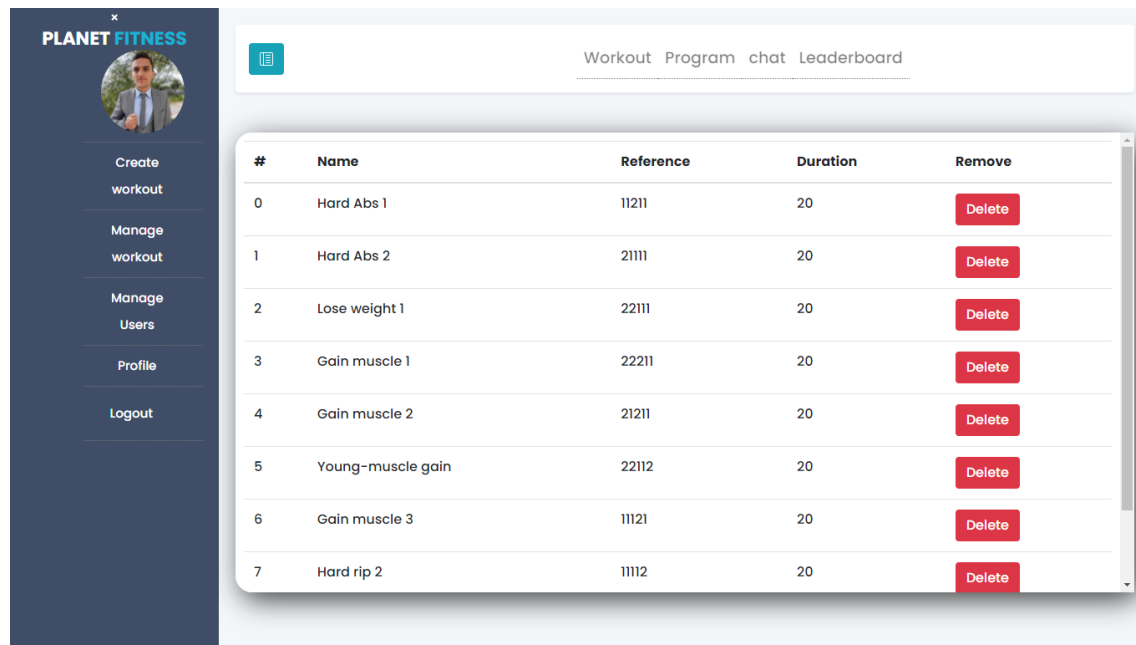


Figure 4.22: The Manage Workout Interface

4.3.7 Manage Workout Interface

Figure 4.3.7 represents the Manage Users Interface which allows the admin to consult all the users and delete anyone he chooses

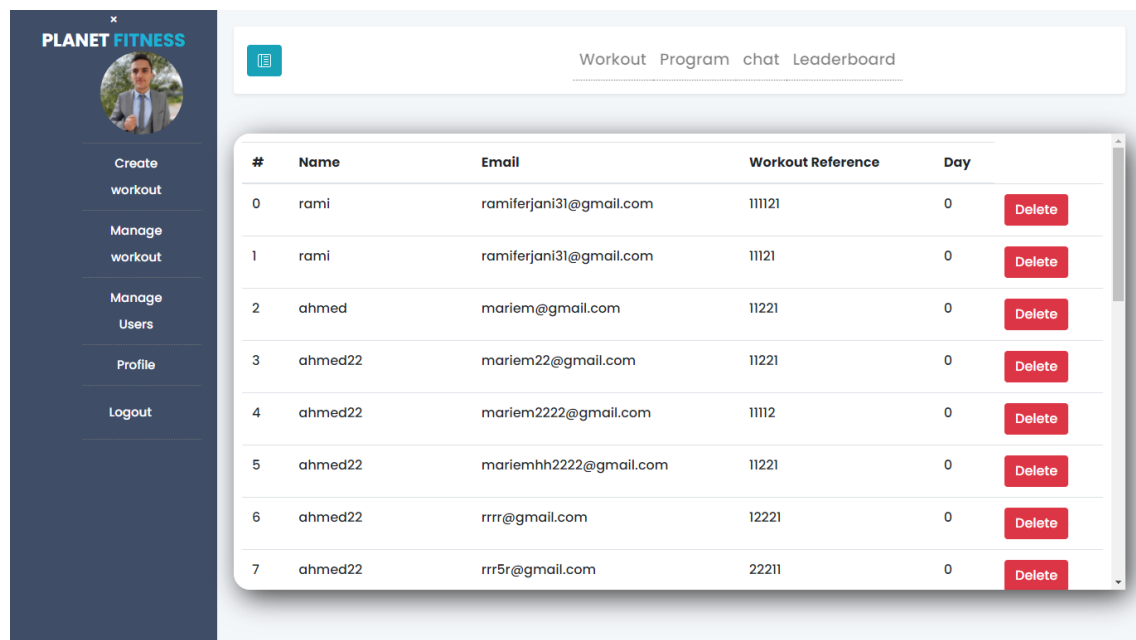


Figure 4.23: Manage Users Interface

4.3.8 Chat Interface

Figure 4.3.8 represents the Chat Interface, the user can type the name of the user he wants to chat with, the interface shows a list of users with similar name, the user then clicks on the user he chooses.

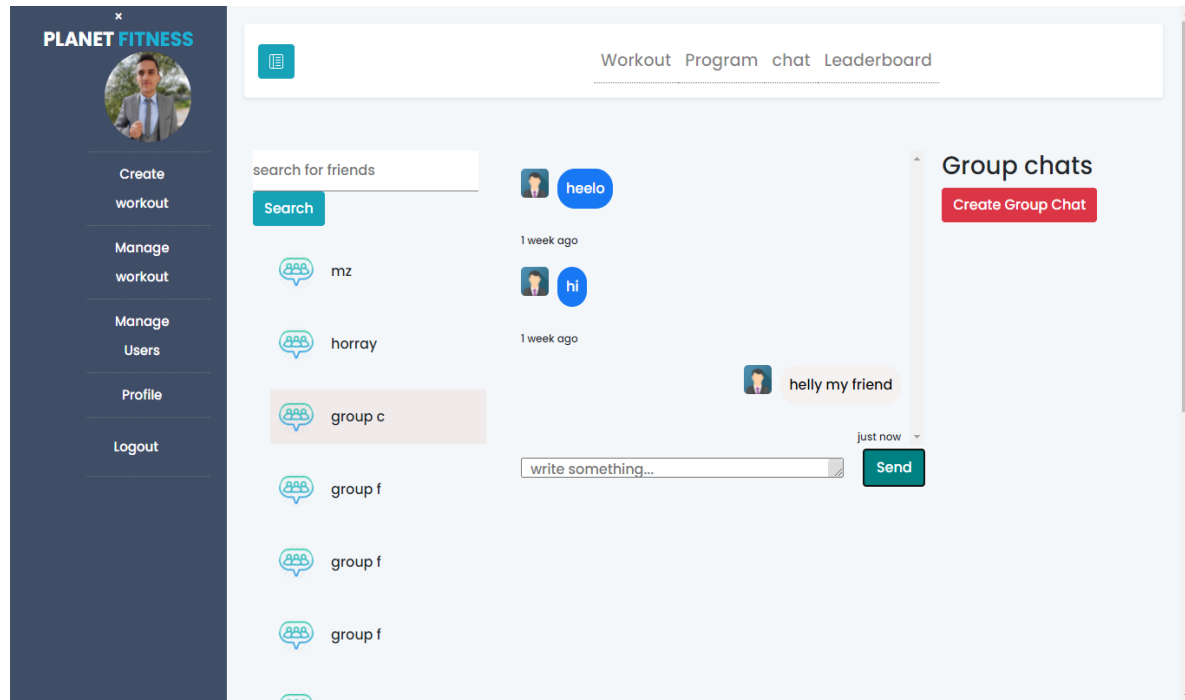


Figure 4.24: Chat Interface

4.3.8 Create Group Chat Interface

Figure 4.24 represent the Create Group Chat Interface, the user chooses the name of the group chat, and then he adds the users and submit.

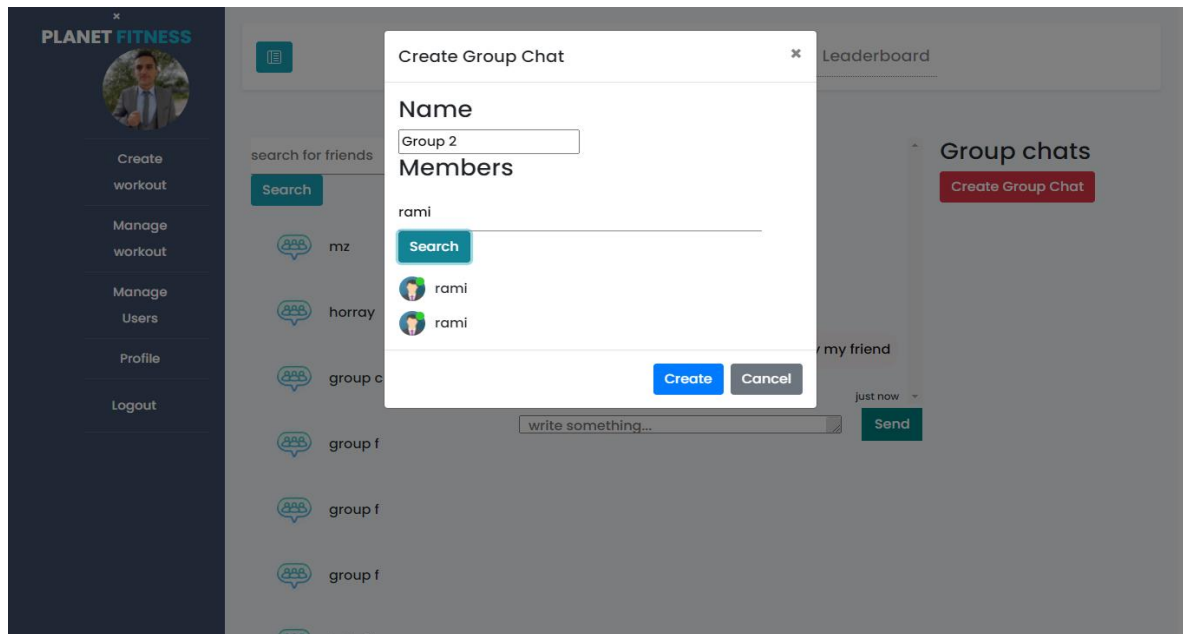


Figure 4.25: Create Group Chat Interface

4.3.8 Leaderboard Interface

Figure 4.25 represents the Leaderboard Interface; any user or admin can check the leaderboard which contains the list of all users ordered by the points earned by completing each day's workout.

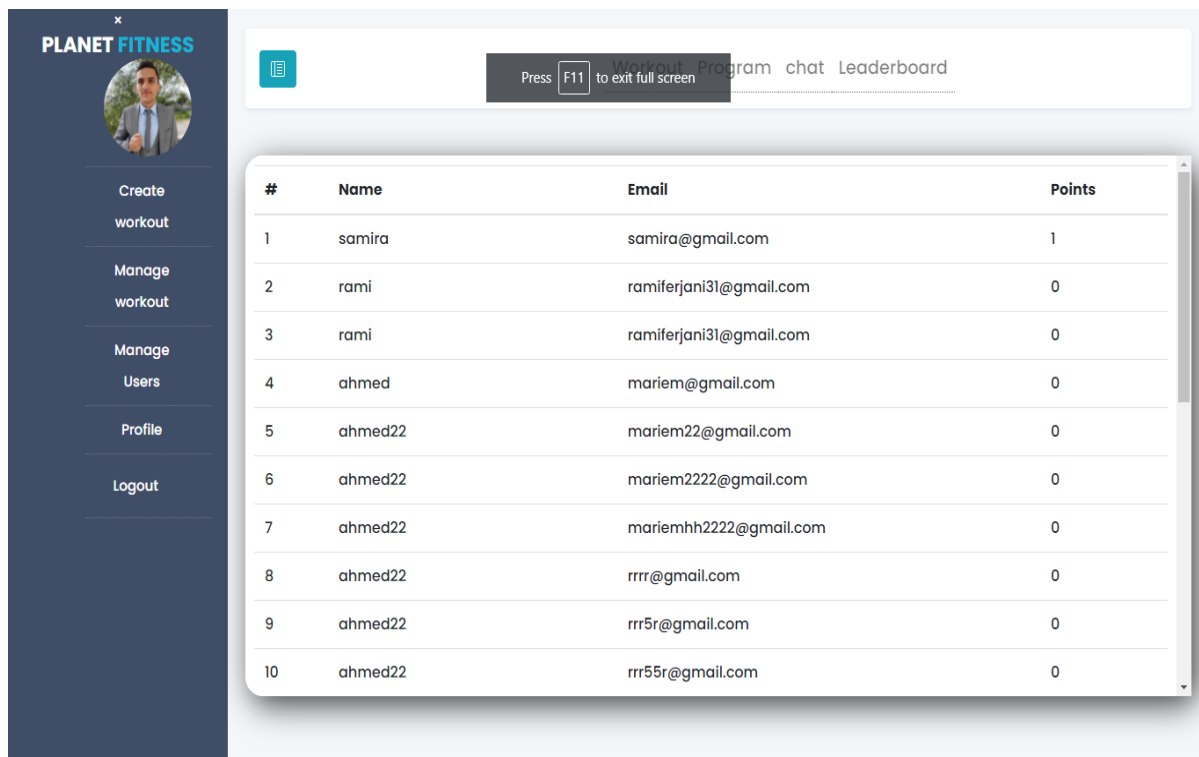
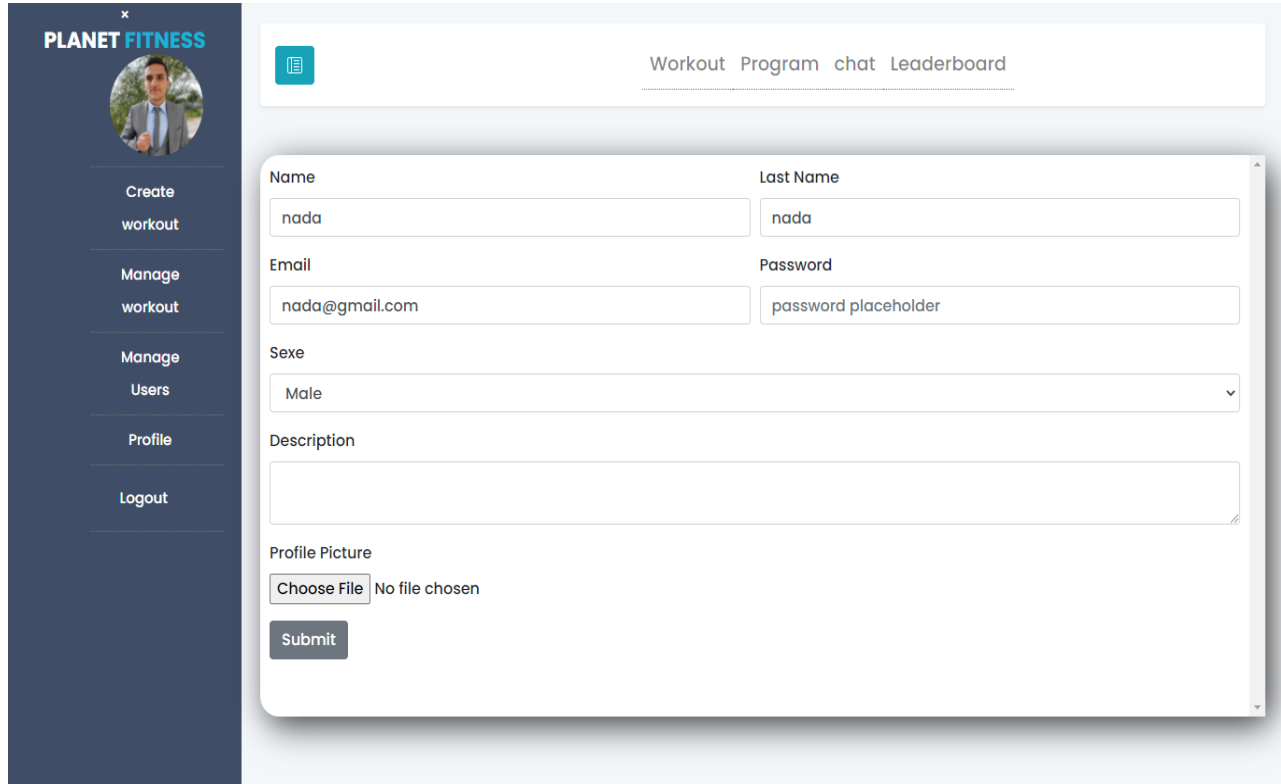


Figure 4.26: Leaderboard Interface

4.3.9 Profile Interface

Figure 4.26 represents the Profile Interface; the user can modify his profile pictures and info using this interface.



The screenshot displays the 'Profile Interface' of the Planet Fitness application. On the left is a dark blue sidebar with the 'PLANET FITNESS' logo at the top, followed by a circular profile picture of a man. Below the picture are five menu items: 'Create workout', 'Manage workout', 'Manage Users', 'Profile', and 'Logout'. The 'Profile' item is currently selected. The main content area has a light blue header with a teal icon and a navigation bar containing 'Workout', 'Program', 'chat', and 'Leaderboard'. The profile form itself is a white box with a light blue border. It contains several input fields: 'Name' (with 'nada' entered), 'Last Name' (with 'nada' entered), 'Email' (with 'nada@gmail.com' entered), and 'Password' (with 'password placeholder' entered). There is a 'Sexe' dropdown menu set to 'Male' and a 'Description' text area. At the bottom, there is a 'Profile Picture' section with a 'Choose File' button (labeled 'No file chosen') and a 'Submit' button.

Figure 4.27:Profile Interface

Conclusion

In this chapter, we described the development environment in which we developed our application, as well as the used frameworks. Followed by an overview of the work carried out through screenshots of the application's interfaces.

General conclusion and perspectives

The period of internship in the Mega-DEV company was quite rewarding for me. As future software developer, this internship has allowed me to learn new technologies, new methodologies and new working environment.

The main objective of the project is to create a workout management system as per the requirements and specifications developed with Mega-DEV. The solution englobes different features such as Workout Management, Chat...

The benefits of this work go beyond the programming knowledge as per the use of modern technologies such as React.js and Express.js frameworks, in fact the Agile project management approach used is a topnotch methodology that allowed us to ensure that bugs were fixed in due time and wave was tailored through every iteration.

Even though the achieved result satisfies the requirements, theres room for improvements, such as adding a mobile version of the application, and to add more interactions between the users.

Among the different lessons picked up during this work, is how to plan ahead and manage the time effectively to achieve the tasks before the deadlines.

Webography

1. Three tier architecture. Accessed 2021-05-15
<https://www.ibm.com/cloud/learn/three-tier-architecture>
2. Visual Studio Code: Accessed 2020-06-11. URL:
https://en.wikipedia.org/wiki/Visual_Studio_Code
3. Redux: Accessed 2021-04-30. URL:
[https://en.wikipedia.org/wiki/Redux_\(JavaScript_library\)](https://en.wikipedia.org/wiki/Redux_(JavaScript_library))
4. Redux: Accessed 2021-04-30. URL:
[https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))
5. Reactstrap. Accessed 2021-02-30. url: <https://www.c-sharpcorner.com/article/reactstrap-in-reactjs/>
6. Express.js: Accessed 2021-03-22. URL: <https://en.wikipedia.org/wiki/Express.js>
7. StarUML: Accessed 2021-04-05. URL: <https://en.wikipedia.org/wiki/StarUML>
8. Draw.io: Accessed 2021-04-12. URL: <https://app.diagrams.net/>
9. Git: Accessed 2021-05-30. URL: <https://en.wikipedia.org/wiki/Git>
10. Redux: Accessed 2021-06-12. URL: <https://www.axelerant.com/resources/team-blog/api-testing-with-postman>
11. Agile methodology: Accessed 2021-02-25.
[://en.wikipedia.org/wiki/Agile_software_development](https://en.wikipedia.org/wiki/Agile_software_development)
12. React Design pattern Accessed: <https://www.educative.io/blog/react-design-patterns-best-practices>
13. <https://en.wikipedia.org/wiki/Socket.IO>

