

**UNIVERSITE SAAD DAHLEB BLIDA 1**

**Faculté des Sciences et de Technologie**

Département d'Electronique



**MEMOIRE DE MASTER**

Spécialité : Electronique des Systèmes Embarqués

Creation of a finding missing persons application  
using MERN stack and flutter

Par

**DIBOUCHE Houssam Eddine & BABACI Abdelmalek**

Promoteur

Madames **BOUGHERIRA Hamida & NACEUR Djamilia**

Blida, Juillet 2020

Année Universitaire  
2019-2020

## **Acknowledgement**

***First of all we would like to express our cordial gratefulness  
to Almighty ALLAH for HIS Kindness,  
for which thing we successfully completed our project.***

***While working on this project we have received many invaluable  
help from a large number of people  
we would like to take this opportunity to express our deepest  
gratitude  
to everyone who helped us.***

***We felt grateful to express our boundless honor and respect to  
our supervisor professor Boughrira Hamida  
for her deep knowledge and keen interest in the field of software  
development that influenced us to carry out of this project.  
Her endless patient helps, friendly support, which have guided us  
throughout our work and showed the path of achievement.***

***We would like to express our heartiest gratitude  
to our professor Naceur Djamila  
for her support and provides more information.***

***Also grad to other faculty members, the staff of the Electronic  
Department, Saad Dahleb Blida University  
and at last but not the least We must acknowledge with due  
respect the constant support and patience  
of our family members for completing this project.***

## Table of Contents

Acknowledgement .....	
Table of Contents .....	
Figures list.....	
Tables list .....	
Abbreviations .....	
Abstract .....	
Introduction .....	1
Chapter 1 Web Application Architecture .....	3
1.1. Introduction .....	3
1.2. Background .....	3
1.2.1. Overview .....	3
1.2.2. Definitions and basic description .....	3
1.2.3. History .....	7
1.2.4. TCP/IP Application Services .....	7
1.2.5. Web Versions .....	9
1.2.6. The Semantic Web .....	12
1.3. Theory .....	13
1.3.1. MVC.....	13
1.3.2. Programming Languages .....	15
1.3.3. Web Application Frameworks .....	15
1.4. Conclusion .....	16
Chapter 2 Project technologies and structure.....	17
2.1. Introduction .....	17
2.2. Web application development.....	17
2.2.1. Various Fields .....	17
2.2.2. Restful Api .....	19

2.2.3. Single Page Application.....	20
2.3. Full Stack .....	21
2.3.1. Overview .....	21
2.3.2. Full Stack JavaScript.....	22
2.4. Tools.....	24
2.4.1. Visual Studio Code .....	24
2.4.2. Postman .....	24
2.4.3. MongoDB Compass.....	24
2.5. MERN Stack .....	25
2.5.1. Overview .....	25
2.5.2. Mongo DB .....	25
2.5.3. Node.js .....	27
2.5.4. Express.js .....	28
2.5.5. React.js .....	28
2.6. Mobile App Integration.....	29
2.6.1. Native Apps.....	29
2.6.2. Hybrid Apps.....	29
2.6.3. Cross Platform Software .....	30
2.7. Real-Time Functionality .....	31
2.7.1. WebSockets.....	31
2.7.2. Socket.io.....	31
2.8. Conclusion .....	32
Chapter 3 Find Me System Analysis And Design .....	33
3.1. Introduction .....	33
3.2. Preliminary study .....	33
3.2.1. System Description .....	33
3.2.2. System Functionality .....	34

3.2.3. Project Requirements .....	35
3.3. Structural/ Procedural design .....	35
3.3.1. Flowchart diagram .....	35
3.3.2. Object Oriented Design.....	36
3.4. Project Model .....	37
3.4.1. Data Flow Diagram (DFD) .....	37
3.4.2. DFD Level 0.....	38
3.4.3. DFD Level 1.....	38
3.4.4. DFD Level 2.....	39
3.4.5. DataBase Architecture .....	53
3.5. Conclusion .....	59
Chapter 4 Implementation and results .....	60
4.1. Introduction .....	60
4.2. Setting Up The Project .....	60
4.2.1. Setting Up The Back-end .....	60
4.2.2. Initiating The Back-end Project .....	60
4.2.3. Setting Up The React Application .....	67
4.3. User Interfaces .....	72
4.3.1. Public Pages .....	72
4.3.2. Private Pages .....	80
4.4. Limitations .....	91
4.5. Conclusion .....	91
Conclusion .....	91
Biographies .....	93

## Figures list

Figure 1.1: Most Popular Web Browsers 2000-2020 .....	6
Figure 1.2: Web 1.0.....	10
Figure 1.3: Web 2.0 architect.....	11
Figure 1.4: Web 3.0.....	12
Figure 1.5: MVC Architecture .....	14
Figure 2.1: Web application model.....	18
Figure 2.2: REST API.....	20
Figure 2.3: Comparison of traditional page lifecycle and SPA lifecycle .....	21
Figure 2.4: Full Stack JavaScript .....	23
Figure 2.5: Block diagram of The Project Architecture .....	24
Figure 2.6: MERN stack architecture .....	25
Figure 2.7: Node.js event loop .....	27
Figure 2.8: Advantages of Flutter .....	31
Figure 3.1: Flowchart diagram for the application .....	36
Figure 3.2: Find Me Use Case Diagram .....	37
Figure 3.3: FindMe Zero Level DFD.....	38
Figure 3.4: FindMe First Level DFD.....	39
Figure 3.5: JWT to access the application server.....	41
Figure 3.6: Authentication Second level DFD.....	42
Figure 3.7: Account management Second level DFD .....	42
Figure 3.8: ‘Track’ management Second level DFD .....	43
Figure 3.9: Second Level DFD of inter-user communication system .....	44
Figure 3.10: Machine Learning and Deep Learning Amount of Data.....	46
Figure 3.11: Various Dimensions of TensorFlow.....	48
Figure 3.12: Manual Search Second Level DFD .....	51
Figure 3.13: Automatic Search Mechanism Second Level DFD.....	52
Figure 3.14: Data Base Schema Assosiations Diagram .....	53
Figure 4.1: Back-End package.json File .....	61
Figure 4.2: Back-End Project app.js File .....	62
Figure 4.3: Regiter Route Source Code .....	63
Figure 4.4: ‘Create-Track’ Route Source Code .....	63
Figure 4.5: Chat Routes Source Code .....	64

Figure 4.6: Register Middleware Source Code .....	65
Figure 4.7: ‘Create-Track’ Middleware Source Code .....	66
Figure 4.8: ‘Send-Message’ Middleware Source Code .....	67
Figure 4.9: Index.js Source Code.....	68
Figure 4.10: Front-End Project app.json File.....	69
Figure 4.11: Register Method Source Code .....	70
Figure 4.12: ‘Create-track’ Method Source Code.....	71
Figure 4.13: ‘Load-Messages’ Method Source Code .....	72
Figure 4.14: Large Screen Landing Page User Interface .....	73
Figure 4.15: Small Screen Landing Page User Interface .....	73
Figure 4.16: Automatic Search User Interface.....	74
Figure 4.17: Signup User Interface .....	75
Figure 4.18: Email Confirmation User Interface .....	76
Figure 4.19: Login User Interface.....	77
Figure 4.20: Forgot Password User Interface .....	78
Figure 4.21: Reset Password User Interface .....	79
Figure 4.22: Error Page User Interface .....	80
Figure 4.23: Other Users Track User Interface.....	81
Figure 4.24: Owner Track User Interface .....	82
Figure 4.25: Temporary Track User Interface .....	82
Figure 4.26: Found Person Track User Interface .....	83
Figure 4.27: Filter User Interface.....	83
Figure 4.28: Right SideBar Component User Interface .....	84
Figure 4.29: Home User Interface Displayed On Large Screen .....	85
Figure 4.30: Home User Interface Displayed On Small Screen .....	85
Figure 4.31: My Tracks User Interface .....	86
Figure 4.32: Create Track User Interface.....	87
Figure 4.33: They Found Me User Interface .....	88
Figure 4.34: Notifications User Interface .....	88
Figure 4.35: Chat User Interface On Large Screen.....	89
Figure 4.36: Chat User Interface On Small Screen.....	89
Figure 4.37: Profile User Interface .....	90
Figure 4.38: Danger Zone User Interface .....	91

**Tables list**

Table 1: User Data Model Schema .....	54
Table 2: Track Data Model Schema .....	55
Table 3: TrackIsFound Data Mode Schema .....	56
Table 4: TempUser Data Model Schema .....	56
Table 5: Image Data Model Schema.....	57
Table 6: Chat Data Model Schema .....	57
Table 7: Message Data Model Schema.....	58
Table 8: Notification Data Model Schema .....	58
Table 9: NotificationGenerator Data Model Schema .....	59



## Abbreviations

ABBREVIATIONS	DEFINITIONS
API	<b>A</b> pplication <b>P</b> rogramming <b>I</b> nterface
APP	<b>A</b> pplication
ARPA	<b>A</b> dvanced <b>R</b> esearch <b>P</b> rojects <b>A</b> gency
ASP	<b>A</b> ctive <b>S</b> erver <b>P</b> ages
AWS	<b>A</b> mazo <b>n</b> <b>W</b> eb <b>S</b> ervices
<i>BBS</i>	<b>B</b> ulletin <b>B</b> oard <b>S</b> ystems
<i>CEO</i>	<b>C</b> hief <b>E</b> xecutive <b>O</b> fficer
CGI	<b>C</b> ommon <b>G</b> ateway <b>I</b> nterface
CNN	<b>C</b> onvolutional <b>N</b> eural <b>N</b> etworks
CPU	<b>C</b> entral <b>P</b> rocessing <b>U</b> nit
CSS	<b>C</b> ascading <b>S</b> tyle <b>S</b> heets
DFD	<b>D</b> ata <b>F</b> low <b>D</b> iagram
FIR	<b>F</b> irst <b>I</b> nformation <b>R</b> eport
FTP	<b>F</b> ile <b>T</b> ransfer <b>P</b> rotocol
DOM	<b>D</b> ocument <b>O</b> bject <b>M</b> odel
GPL	<b>G</b> NU <b>P</b> ublic <b>L</b> icense
GPU	<b>G</b> raphics <b>P</b> rocessing <b>U</b> nit
GUI	<b>G</b> raphical <b>U</b> ser <b>I</b> nterface
HTML	<b>H</b> yper <b>T</b> ext <b>M</b> arkup <b>L</b> anguage
HTTP	<b>H</b> yper <b>T</b> ext <b>T</b> ransfer <b>P</b> rotocol

IBM	<b>I</b> nternational <b>B</b> usiness <b>M</b> achines
IOT	<b>I</b> nternet <b>O</b> f <b>T</b> hings
IMAP	<b>I</b> nternet <b>M</b> essage <b>A</b> ccess <b>P</b> rotocol
IP	<b>I</b> nternet <b>P</b> rotocol
JSON	<b>J</b> ava <b>S</b> cript <b>O</b> bject <b>N</b> otation
JSP	<b>J</b> ava <b>S</b> erver <b>P</b> ages
LAMP	<b>L</b> inux, <b>A</b> pache, <b>M</b> ySQL, <b>P</b> HP/Perl/Python
MVC	<b>M</b> odel <b>V</b> iew <b>C</b> ontroller
MERN	<b>M</b> ongoDB/ <b>E</b> xpress JS/ <b>R</b> ect JS / <b>N</b> ode JS
MIT	<b>M</b> assachusetts <b>I</b> nstitute of <b>T</b> echnology
MSN	<b>M</b> icrosoft <b>N</b> etwork
MySql	My(the name of co-founder Michael Widenius's daughter) <b>S</b> tructured <b>Q</b> uery <b>L</b> anguage
NPM	<b>N</b> ode <b>P</b> ackage <b>M</b> anager
OS	<i><b>O</b>perating <b>S</b>ystem</i>
PAAS	<b>P</b> latform <b>A</b> s <b>A</b> <b>S</b> ervice
PARC	<b>P</b> alo <b>A</b> lto <b>R</b> esearch <b>C</b> enter
PERL	<b>P</b> ractical <b>E</b> xtraction and <b>R</b> eport <b>L</b> anguage
PHP	<b>P</b> ersonal <b>H</b> ome <b>P</b> age
POP	<b>P</b> ost <b>O</b> ffice <b>P</b> rotocol
RDBMS	<b>R</b> elational <b>D</b> atabase <b>M</b> anagement <b>S</b> ystem
RDF	<b>R</b> esource <b>D</b> escription <b>F</b> ramework

REST	<b>R</b> epresentational <b>S</b> tate <b>T</b> ransfer
RSS	<b>R</b> eally <b>S</b> imple <b>S</b> yndication
SMTP	<b>S</b> imple <b>M</b> ail <b>T</b> ransfer <b>P</b> rotocol
SPA	<b>S</b> ingle <b>P</b> age <b>A</b> pplication
SQL	<b>S</b> tructured <b>Q</b> uery <b>L</b> anguage
TCP	<b>T</b> ransmission <b>C</b> ontrol <b>P</b> rotocol
TPU	<b>T</b> ensor <b>P</b> rocessing <b>U</b> nit
UI	<b>U</b> ser <b>I</b> nterface
URL	<b>U</b> niform <b>R</b> esource <b>L</b> ocator
VS	<b>V</b> isuel <b>S</b> tudio code
XHTML	<b>E</b> xtensible <b>H</b> yper <b>T</b> ext <b>M</b> arkup <b>L</b> anguage
XML	<b>E</b> xtensible <b>M</b> arkup <b>L</b> anguage

## Abstract

The phenomenon of kidnapping and loss of people has increased in the recent years, whether mentally retarded persons or children. It is difficult to find them either by the traditional methods or by the modern methods which lack research speed and rely heavily on the traditional research. Our objective is to create a web application using the “MERN stack” and a mobile application works on the two most popular platforms (Android and iOS) using "Flutter" which is the framework of dart based on the automatic search using the images of the missing persons and the face recognition feature to help find them as soon as possible. The users have more freedom that they can also use the traditional and the manual search.

**Keywords:** MERN ; Dart ; Flutter ; JavaScript ; NodeJs ; Missing Persons ; Web Application Mobile Application ; Cross-platform Application ; Andriod ; IOS ; Face-api.js ; Face Recognition ; Track.

## ملخص

ازدادت ظاهرة الاختطاف وضياع الناس شيوعاً في السنوات الأخيرة سواء في ما يتعلق بالأشخاص المتأخرة ذهنياً أو الأطفال. ومن الصعب إيجادهم سواء بالطرق التقليدية أو الطرق الحديثة التي تقتصر لسرعة البحث وتعتمد بشكل كبير على البحث التقليدي.

هدفاً من المشروع هو إنشاء تطبيق الويب باستخدام (MERN stack) والهواتف الذكية على أشهر منصتين (أندرويد و Ios) باستخدام فلاتر إطار عمل اللغة دارت يعتمد على البحث التلقائي باستعمال صور المفقودين عبر خاصية التعرف على الوجوه للمساعدة في إيجاد هؤلاء المفقودين في أقرب وقت كما يمكن استخدام البحث التقليدي واليدوي لمنح حرية أكبر للمستخدمين .

**كلمات مفتاحية :** تطبيق ويب ، تطبيق هاتف ، مسار ، تمييز الوجود ، أشخاص مفقودين ، تطبيق متعدد المنصات ، موقع الكتروني

## Résumé

Le phénomène des enlèvements et des pertes de personnes a augmenté en commun ces dernières années, qu'il s'agisse de personnes mentalement en retard ou d'enfants. Il est difficile de les trouver soit par des méthodes traditionnelles, soit par des méthodes modernes qui manquent de rapidité de recherche et reposent fortement sur la recherche traditionnelle.

Notre objectif du projet est de créer une application Web utilisant « MERN stack » et une application mobile fonctionnant sur les deux plateformes les plus populaires (Android et iOS) en utilisant le langage « DART » du framework « Flutter » basé sur la recherche automatique en utilisant les images des personnes disparues grâce à la fonction de reconnaissance faciale pour aider à retrouver les personnes disparues le plus tôt possible, comme c'était le cas. On peut utiliser la recherche traditionnelle et manuelle pour donner plus de liberté aux utilisateurs.

**Mots clés:** MERN ; Dart ; Flutter ; JavaScript ; NodeJs ; Personnes Disparues ; Application Web ; Application mobile ; Android ; IOS ; Face-api.js ; Reconnaissance Faciale ; Piste .

## Introduction

One of the greatest fears of people is to lose their relatives. Each year many people get lost in Algeria. In some cases, the lost person is found easily, but in some critical cases, missing persons are never reunited with their relatives. Finding a lost person can be a difficult task

The currently available manual system for finding missing persons have a very long procedure and takes more time. More time is required for launching an 'FIR' (First Information Report) in the police station. In addition, more time is required for finding a lost person, and during the manual process, more manpower is needed.

To make the task of finding missing persons easy we developed a web app and a mobile app (cross-platform Web, Ios, and Android) that allow the user to enter the missing persons information and keep a track of it. The user should register oneself to the system using his email or his Facebook account to use all the system features and to keep his missing person on track.

The search mechanism is based on the face recognition feature, the App allows the authenticated user and other users that don't want to register or login to check the missing persons at their or any other area, just they need to provide the missing person picture and in return the system displays the missing persons information if there is a match.

Some existing applications do not show the proper information about the Missing person, which makes it difficult to find out the missing person. Some missing person related websites show only the database of the missing person. To overcome from this, some applications have been developed. But these applications have certain limitations such as:

- The user cannot add a complaint nor the missing person information.
- They display the advertisements, which are collected from newspapers.
- Previous applications use humans to recognize the missing persons.

To overcome these drawbacks we developed a single page application based on the MVC (Model-View-Controller) "FindMe". The MVC pattern separates data and their graphical appearance. The pageflow is controlled by the Controller. The Model is responsible for fetching data (by accessing a database) and the View is responsible for the graphical appearance of the data within Web pages. We used the MERN Stack, mongoDB as

database, Express.js for the server side, and react.js for the Front-end web application and flutter framework for the Front-end mobile. Our application is basically designed to perform all the tasks that the previous systems can perform and all functionalities that are provided by existing applications, as well as it gives additional features to the user which based on recognizing the missing people face using 'Artificial Intelligence' algorithm, Google Tensorflow.

To achieve our work, we have organized our thesis in four chapters as follows:

In the first chapter we will talk about web application architectures, we will start by giving a background on the web and the internet world followed by general concepts on the programming languages and frameworks.

In the second chapter, we will define the basic web development concept then we will explain the full stack meaning and the different web stacks. The main objective of the chapter is to study the different components of the most popular Full Stack JavaScript framework, MERN stack, and describe other project technologies and structure.

In the third chapter, we will describe our system and we will analyze the project using flowcharts and data flow diagrams.

The last chapter contains the result of the project implementation. We present the UI of the project main pages. Then we discuss the project limitations and future enhancements.

# Chapter 1 Web Application Architecture

## 1.1. Introduction

Since our objective is to develop a web application for IOs, and Android, we will describe in this chapter the web application architecture, starting with a general background, which contains general concepts on the web and the internet world. This chapter also provides a detailed information about the TCP/IP services and the different web versions followed by the semantic web. Then we will explain some theory about the programming languages, frameworks, and the mvc architecture.

## 1.2. Background

### 1.2.1. Overview

There is an often-overlooked distinction between the Web and the Internet. The line between the two is often blurred, partially because the Web is rooted in the fundamental protocols associated with the Internet. Today, the lines are even more blurred, as notions of ‘the Web’ go beyond the boundaries of pages delivered to Web browsers, into the realms of wireless devices, personal digital assistants, and the next generation of Internet appliances.[1]

### 1.2.2. Definitions and basic description

#### 1.2.2.1. Internet

The Internet is the global system of interconnected computer networks that uses the Internet protocol suite (TCP/IP) to communicate between networks and devices. It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies.[19]

The Internet carries a vast range of information resources and services, such as the inter-linked hypertext documents and applications of the World Wide Web, electronic mail, telephony, and file sharing.[28]

#### 1.2.2.2. Web

From the very beginnings of Internet technology, there has been a dream of using the Internet as a universal medium for exchanging information over computer networks. Many people shared this dream.



The Web is built on top of core Internet protocols that had been in existence for many years prior to the Web's inception. Understanding the relationship between 'Web technology' and the underlying Internet protocols is fundamental to the design and implementation of true 'Web applications'. [1]

The Web such as global database that user can share information through his device connected to the internet. There are many of resources explain the stages of the Web technology through its development whenever it is become easier for users it is become more complex for developer. The Web is involved from simple to more advance structures.[28]

### **1.2.2.3. Web page**

A web page or webpage is a document, commonly written in HTML, which is viewed in an Internet browser. A web page can be accessed by entering a URL address into a browser's address bar. A web page may contain text, graphics, and hyperlinks to other web pages and files.[19]

A web page is often used to provide information to viewers, including pictures or videos to help illustrate important topics. A web page may also be used as a method to sell products or services to viewers. Multiple web pages make up a website, like our Computer Hope website[1].

### **1.2.2.4. Web site**

A Web site is more than just a group of Web pages that happen to be connected to each other through hypertext links.

There are also architectural concerns. As a site grows in size and becomes more complex, it becomes critically important to organize its content properly. This includes not just the layout of content on individual pages, but also the interconnections between the pages themselves.[19]

### **1.2.2.5. Web applications**

A Web application is something more than just a 'Web site.' It is a client/server application that uses a Web browser as its client program, and performs an interactive service by connecting with servers over the Internet. A Web site simply delivers content from static files. A Web application presents dynamically tailored content based on request parameters, tracked user behaviors, and security considerations.[19]

### 1.2.2.6. Web Servers

In computing, a server is a computer program or a device that provides functionality for other programs or devices, called "clients". This architecture is called the client–server model. Servers can provide various functionalities, often called ‘services’, such as sharing data or resources among multiple clients, or performing computation for a client.[15]

Web servers enable HTTP access to a ‘Web site,’ which is simply a collection of documents and other information organized into a tree structure, much like a computer’s file system. In addition to providing access to static documents, modern Web servers implement a variety of protocols for passing requests to custom software applications that provide access to dynamic content.[15]

### 1.2.2.7. Web browsers

A web browser is a software application for accessing information on the World Wide Web. When a user requests a web page from a particular website, the web browser retrieves the necessary content from a web server and then displays the page on the screen.

A web browser is not the same thing as a search engine, though the two are often confused. For a user, a search engine is just a website, such as Google Search, that stores searchable data about other websites. However, to connect to a website's server and display its web pages, a user must have a web browser installed.[15]

Web browsers are used on a range of devices, including desktops, laptops, tablets, and smartphones.

This is the list of the most popular and the most used web browsers:

- Chrome
- Internet Explorer
- Firefox
- Edge
- Opera

Figure 01 illustrates the most popular web browsers between 2000 and 2020.

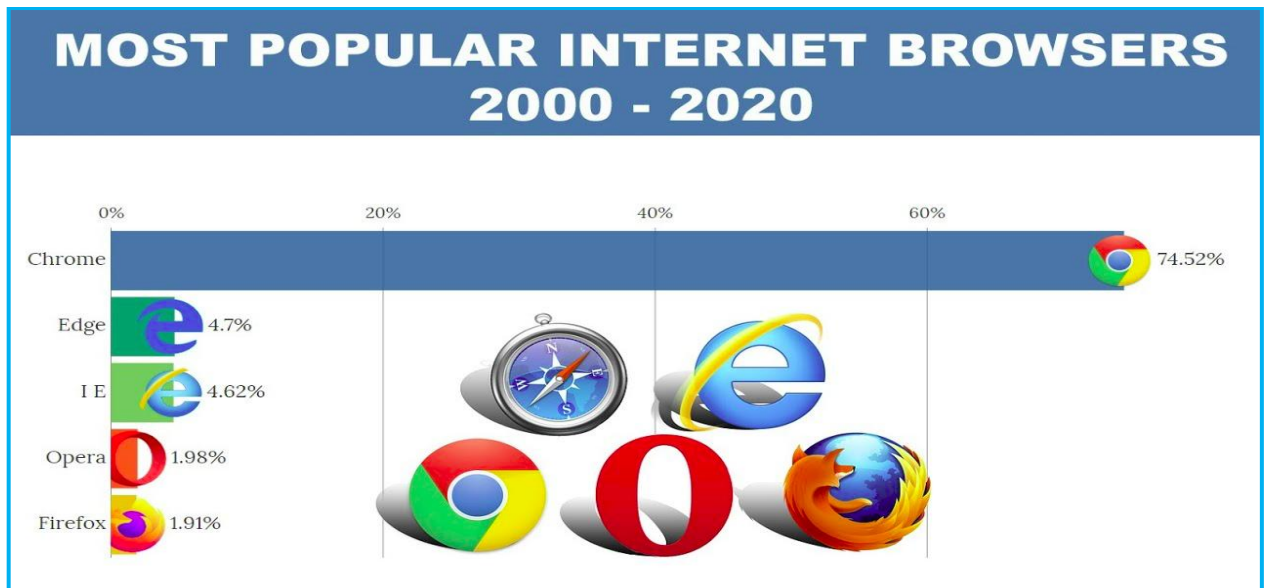


Figure 1.1: Most Popular Web Browsers 2000-2020

### 1.2.2.8. Internet Protocols

#### 1.2.2.8.1. TCP/IP

TCP/IP, is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP can also be used as a communications protocol in a private computer network (an intranet or an extranet).[3]

TCP/IP specifies how data is exchanged over the internet by providing end-to-end communications that identify how it should be broken into packets, addressed, transmitted, routed and received at the destination.[6]

TCP/IP requires little central management, and it is designed to make networks reliable, with the ability to recover automatically from the failure of any device on the network.

#### 1.2.2.8.2. HTTP

HTTP is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access by a mouse click or by tapping the screen in a web browser.[29]

#### 1.2.2.8.3. Public Pages

We say that a page is public when the user can see it without logging in. Most pages are only viewable by authenticated users. A limited number of pages are public so that users

do not have to log in to view them, such as the landing page, the error page, and the register and login pages.

#### **1.2.2.8.4. Private Pages**

We say that a page is private when the user can only see it if he is logged in and authenticated. The private pages are available only to internal personnel or to registered users or partners with passwords. In web applications, most pages are private.

#### **1.2.3. History**

The roots of Web technology can be found in the original Internet protocols (known collectively as TCP/IP), developed in the 1980. These protocols were an outgrowth of work done for the United States Defense Department to design a network called the ARPANET.[6]

The ARPANET was named for ARPA, the Advanced Research Projects Agency of the United States Department of Defense. It came into being as a result of efforts funded by the Department of Defense in the 1970 to develop an open, common, distributed, and decentralized computer networking architecture.[3]

#### **1.2.4. TCP/IP Application Services**

##### **1.2.4.1. Telnet**

The Telnet protocol operates within the Application layer. It was developed to support Network Virtual Terminal functionality, which means the ability to ‘log in’ to a remote machine over the Internet. The latest specification for the Telnet protocol is defined in Internet RFC 854.[6]

Telnet clients are configured by default to connect to port 23 on the server machine, but the target port number can be over-ridden in most client programs. This means you can use a Telnet client program to connect and ‘talk’ to any TCP server by knowing its address and its port number.[3]

##### **1.2.4.2. Message forums**

Message forums are online services that allow users to write messages to be posted on the equivalent of an electronic bulletin board, and to read similar messages that others have posted. These messages are usually organized into categories so that people can find the kinds of messages they are looking for.[29]

#### **1.2.4.3. Live messaging**

Today, the vast majority of Internet users eschew command-line interfaces, and the notion of being logged in to a particular system is alien to most people. Thus, a protocol like talk would not work in its original form in today's diverse Internet world. Proprietary 'instant messaging' systems exist, but they are exclusionary, and the intense competition and lack of cooperation between instant messaging providers further limits the degree of interoperability we can expect from them.[20]

#### **1.2.4.4. E-mail**

Electronic mailing lists provided communities where people with like interests could exchange messages. These lists were closed systems, in the sense that only subscribers could post messages to the list, or view messages posted by other subscribers. Obviously, lists grew, and list managers had to maintain them. Over time, automated mechanisms were developed to allow people to subscribe without human intervention. These mailing lists evolved into message forums, where people could publicly post messages, on an electronic bulletin board, for everyone to read.[3]

The transmission of electronic mail is performed through the SMTP protocol. The reading of electronic mail is usually performed through either POP or IMAP.

#### **1.2.4.5. File servers**

For years before the existence of the Internet, files were shared using BBS. People would dial in to a BBS via a modem, and once connected, they would have access to directories of files to download (and sometimes to 'drop' directories into which their own files could be uploaded). Various file transfer protocols were used to enable this functionality over telephone dialup lines.[12]

To facilitate this functionality over the Internet, the File Transfer Protocol (FTP) was created.

An FTP server operates in a manner similar to an e-mail server. Commands exist to authenticate the connecting user, provide the user with information about available files, and allow the user to retrieve selected files.[8]

FTP servers also allow users to traverse to different directories within the server's local file system, and to upload files into those directories.

## 1.2.5. Web Versions

### 1.2.5.1. Web 1.0

It represents the basic of Web it was used even 2003, invented by Tim Berners-Lee and it is just the readable site with raw data of the World Wide Web. The user can only Search and read the information through browser, he cannot share and commend on the site.[7]

In other words, it is static information. In Web 1.0, a few of person that must be has knowledge of how the Web pages are designed (interlinked) can create and modified the Web pages compare with large number of users there not necessary to have knowledge about how the Web pages are designed. In the Web 1.0 technique, some companies design applications that allow users download information from the Web but they have not seeing the procedure of how the applications works. Technologies used in Web1.0 are HTML, HTTP and URI.[7]

In addition, other protocols used in web1.0 like XML, XHTML and CSS. There are combined technologies between server and client such as ASP, PHP, JSP, CGI, and PERL. The server side uses JavaScript, VBscript and flash on the client. Web 1.0 it is very slow and the user need to refresh the site every time when new information added to the web pages. The web1.0 problem it just works one direction. In other word, the user cannot post or modify the web page.[18]

Figure 02 illustrates the web1.0.

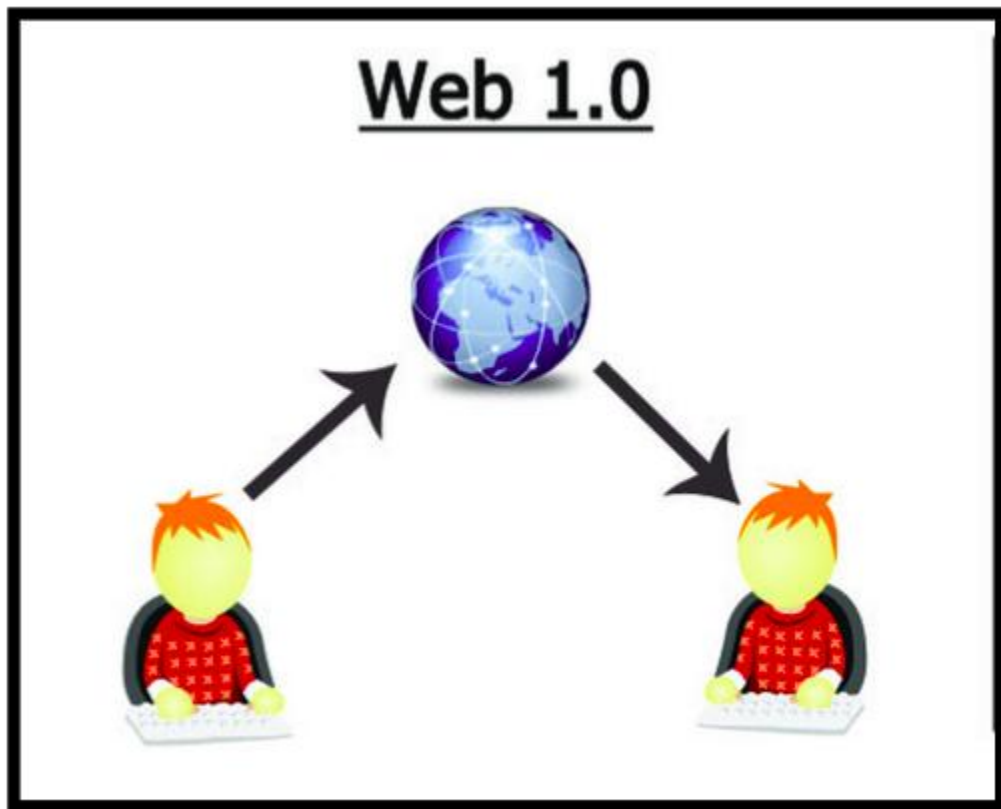


Figure 1.2: Web 1.0

### 1.2.5.2. Web 2.0

It is the second version of web. In 2004 it presented formally by Dale Dougherty who was vice-president of O'Reilly Media. It is also called the read and write web, it is representing a new method to use the current technologies of internet, and the web could become bidirectional.[7]

Actually, the web1.0 presents to the user accessing possibility to upload and download from the webpage like provider but in limited controlling. In other words, actual interactive of user to allow simply upload as well as download.

The users of web 2.0 have more interaction with less control. Technology infrastructure of web2.0 consist of some rules such as RSS, Atom, RDF witch used by the designer for creation the web 2.0 services, also the web2.0 uses Ajax technology in internet such as JavaScript and XML, DOM, REST, XML and CSS. The web2.0 allows the users the ability to creation social activities and communicates with each other. But these properties also consider issues because the user can be hacked in privacy and personal information security.[7]

Figure 03 illustrates the web2.0 architecture.

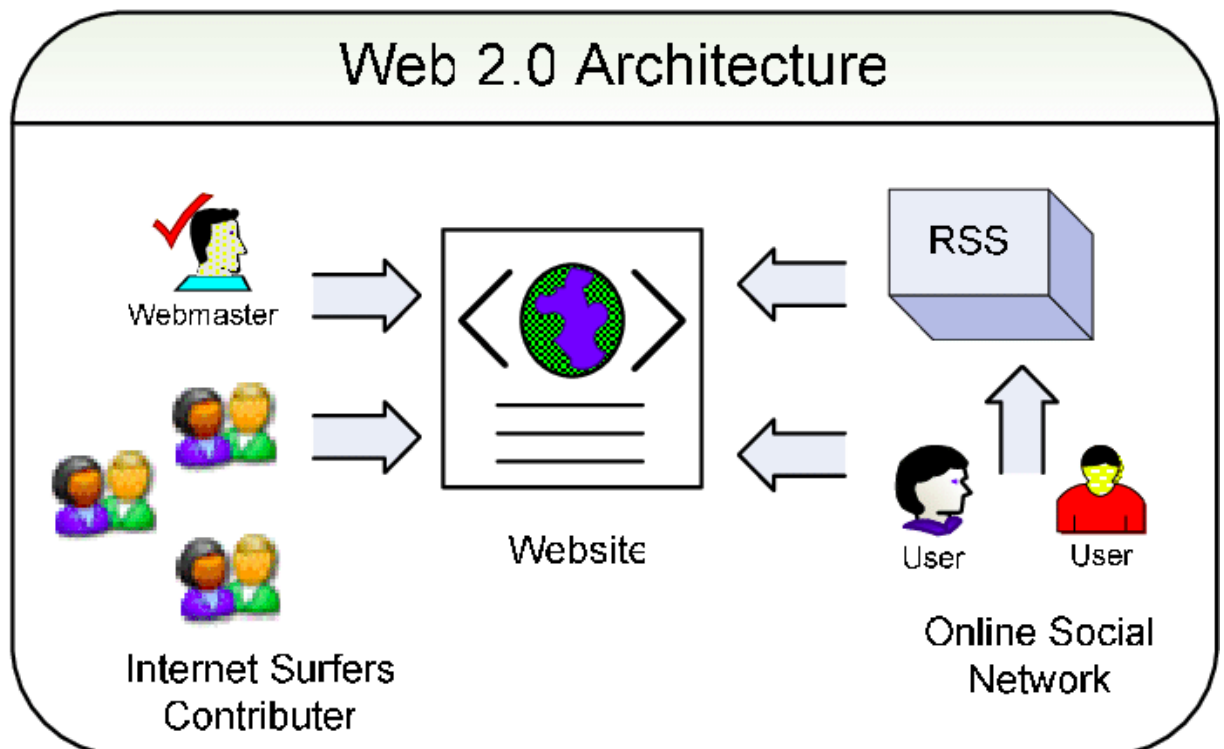


Figure 1.3: Web 2.0 architect

### 1.2.5.3. Web 3.0

It is also the third and current version of web started in 2014 known as executable web that allows user the ability to interact with dynamic applications. In other word, sometimes is called the Semantic Web and personalization.

Web 3.0 will be a complete reinvention of the web. Conrad Wolfram theory about web3.0 tries to make the computer be able thinking and more intelligence for search about new data instead the humans . Web 3.0 is a new method that used in various fields on the internet. In other word, convert the web into huge database.

In Web 3.0, proposed to be the computers like human to describe the specific information in high speed and bring the information for the user as meaning of word and do not search for the same word in web. One example of Web 3.0 is Google, which is technology infrastructure of web3.0. [7]

Figure 04 illustrates the web3.0.





Figure 1.4: Web 3.0

#### 1.2.5.4. Portals and Search Engines

A Web portal is a type of sites designed to presents to the user the ability to visit and providing a link to other site. It is created for some purpose like distributed applications, the share information between the users. In other word, it also can be represented as huge database of components that different number of user can uses it at the same time.[20]

The web portal allows the user in search navigation and information integration, also provides some other features business intelligence and distribution games. There are some examples of Web portal like MSN, NAVER.

A search engine is also term can have called search sites that designed to provides to the user capabilities to access the information from any website stored on a server, such as on the World Wide Web. In other word, is a web system uses some criteria to find the information and bring all sites of web that contains this word or phrases.[18]

#### 1.2.6. The Semantic Web

##### 1.2.6.1. Overview

The web becomes day to day larger and the search about any word or phrase is issue because many sites contain this word and many of them do not have the correct information. Tim Berners-Lee proposed the new term is Semantic Web. It is last version of the web that helps us find the exact information that we are searching for by machines

instead of human. Semantic web uses techniques that search depending on meaning of word and what the user think about .

Semantic Web uses some criteria using in search like the location of the user , and previous searches of user by providing the inclusion of semantic content in Web pages. In other word, the Semantic Web uses a technology that allows machines to understand the user and respond the user requests subject to their meaning.[7]

The semantic web has levels and until today all the levels have not completed. Example for semantic web is GOOGLE. The semantic web is not so much a technology as an infrastructure, enabling the creation of meaning through standards, markup languages, and related processing tools. Each layer of the semantic web technology stack provides services to the layer above and draws on the services of the layers below.

### **1.2.6.2. Semantic Web Technology**

The technology of semantic web uses standard semantics for the data around us to give the full meaning. World Wide Web inventor envisions this technology with Linked Data technology. According to Sir Tim Berner-Lee there are relationships between data in all formats and sources. This technology gives the ability for machines to store, manage and return information according to their meaning and their relationships.[7]

### **1.2.6.3. Semantic Web Stack**

It is one of the semantic web layers also known as semantic web cake. The semantic web is representation of the language, the technologies that use semantic web are shown by the stack

## **1.3. Theory**

### **1.3.1. MVC**

#### **1.3.1.1. MVC Architecture**

Model-View-Controller (or MVC) is probably one of the most popular architectures for applications. As with many other cool things in computer history, the MVC model was conceived at PARC for the Small-talk language as a solution to the problem of organizing applications with graphical user interfaces. It was created for desktop applications, but since then, the idea has been adapted to other mediums including the Web.[5]

We can describe the MVC architecture in simple terms:

- Model: the part of our application that will deal with the database or any data-related functionality.
- View: everything the user will see the pages that we are going to send to the client.
- Controller: the logic of our site, and the glue between models and views. Here we call our models to get the data, and then we put that data on our views to be sent to the users.[5]

Figure 05 illustrates the mvc architecture.

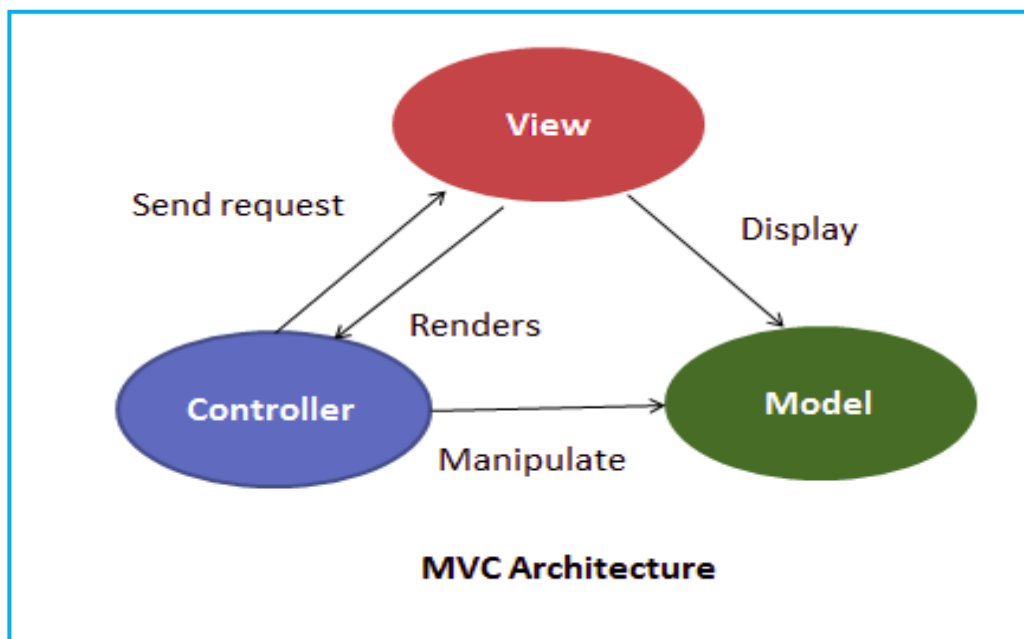


Figure 1.5: MVC Architecture

### 1.3.1.2. MVC Advantages and Drawbacks

The structure allows flexibility since responsibilities are clearly separated. This leads to:

- better and easier code maintenance and re-usability
- easier to coordinate in teams due to the separation
- ability to provide multiple views
- support for asynchronous implementations

However, it has some disadvantages like:

- an increased complex setup process
- dependencies, the changes in the model or controller affect the whole entity.[5]

## **1.3.2. Programming Languages**

### **1.3.2.1. Overview**

A programming language is a formal language comprising a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms.

### **1.3.2.2. High-level Programming Languages**

High-level programming languages are designed to be easy to read and understand. This allows programmers to write source code in a natural fashion, using logical words and symbols.

Many high-level languages are similar enough that programmers can easily understand source code written in multiple languages. Examples of high-level languages include C++, Java, Perl, and PHP. Languages as C++ and Java are called "compiled languages" since the source code must first be compiled in order to run.

Languages as Perl and PHP are called "interpreted languages" since the source code can be run through an interpreter without being compiled. Generally, compiled languages are used to create software applications, while interpreted languages are used for running scripts, such as those used to generate content for dynamic websites.

### **1.3.2.3. Low-level Programming Languages**

Low-level programming languages include assembly and machine languages. An assembly language contains a list of basic instructions and is much more difficult to read than a high-level language. In rare cases, a programmer may decide to code a basic program in an assembly language to ensure it operates as efficiently as possible.

An assembler can be used to translate the assembly code into machine code. The machine code, or machine language, contains a series of binary codes that are understood directly by a computer's CPU.

## **1.3.3. Web Application Frameworks**

A web framework is a website development technology that is used to make the web applications which including web services, web resources,

A web development framework is a set of resources and tools for software developers to build and manage web applications, web services and websites. Web development framework can be built upon a pre-defined infrastructure such as the Linux.

The web development framework also provides the foundations and system level services for software developers to build a content management system for managing digital information on the Web development framework

### **1.4. Conclusion**

In this chapter, we described in details the web applications architecture, which contains the main concepts that any web developer needs. In the next chapter we will present our project structure, which consists mainly in a MERN application and a cross-platform mobile application which was created using dart and flutter framework. We will talk in depth about the languages and the technologies we have used to make these applications.

## **Chapter 2 Project technologies and structure**

### **2.1. Introduction**

This chapter describes the structure of the find-Me application. It provides a detailed information on the different web stacks, and explains all the technologies and the programming languages that we used in the chosen stack.

This chapter also describes the real time integration and all the technologies we have used in it.

### **2.2. Web application development**

#### **2.2.1. Various Fields**

##### **2.2.1.1. Front-end**

Web application development is the combination of the front-end and the back-end development. Front-end web development, also known as client-side development, involves the practice of creating GUI for clients (users) so that the users can interact with the application. It involves the use of primary web technologies and tools such as HTML, CSS, and JavaScript.

HTML is a mark-up language, which provides the structure to a web page. It defines how a web page would look like so it can be considered the skeleton of any web application. CSS, on the other hand, is a style sheet language, which provides style and visual enhancements to the documents written in HTML.[4]

JavaScript is the most advanced language among these technologies. It performs HTML DOM manipulation to provide a dynamic interface to users. Moreover, it provides an interactive interface to the users by creating pop-up messages, validating form inputs, and changing the layout based on events like user-input or mouse clicks. The browser to provide a front-end web interface controls all these technologies.[11]

##### **2.2.1.2. Back-end**

Back-end web development, also known as server-side development, involves the development of computer programs and databases to serve the client. A web application in its primary days did not need to have a front-end but a functioning server-side application was enough for it to be considered a web application.[9]

Several changes have been made in this field since then. Today's sophisticated web applications cannot run without both the front-end and back-end services. Back-end technologies usually consist of the programming languages such as PHP, Ruby, Python, Java, Node.js, and different frameworks.[4]

### 2.2.1.3. Back-end/Front-end communication

A web application, in its most elementary form, sends an HTTP request to a server to establish connection, and the server sends an HTTP response to the client. A typical example of communication in a web application is illustrated in figure 6.

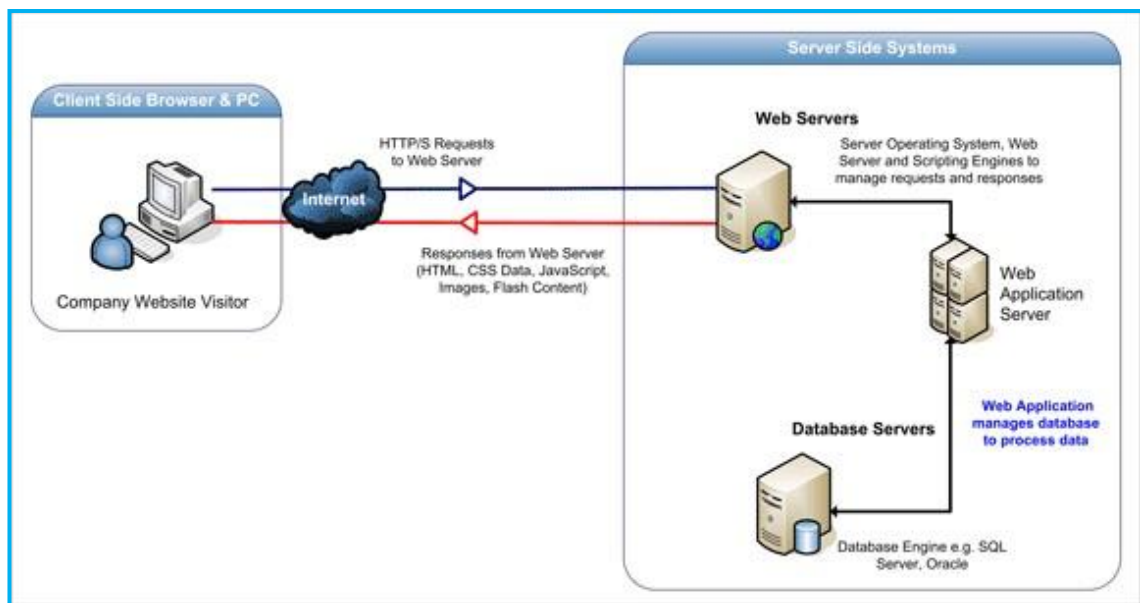


Figure 2.1: Web application model

Figure 01 illustrates the communication among the three layers of a web application model. The first layer is a client-side web browser, the second layer is a server-side dynamic content generator, and the third layer is a database server.[9]

A user sends an initial request using the HTTP protocol through the browser over the internet to the server. The web server then processes the request by accessing the database server and retrieving the requested data.[11]

The web server then sends the response to the user over the internet through the browser. The response usually contains the data requested by the user.

## **2.2.2. Restful Api**

### **2.2.2.1. Overview**

REST is an architectural style used in web development in order to create web services. REST only defines the principles on which a web service is developed for the client- server communication. It is not a set of rules (protocols) for creating web services. Any web services or APIs that are designed with the REST architecture are called RESTful APIs, or just REST APIs. REST provides good performance, scalability, and reliability in a distributed computing system.[4]

### **2.2.2.2. Basic design principles**

An implementation of REST APIs must follow at least four basic design principles:

- Use of HTTP methods: REST APIs must follow the HTTP methods explicitly. They must use GET to retrieve a resource from the server, POST to create a resource, PUT to modify or update a resource, and DELETE to delete a resource.
- Stateless communication: Communication between the client and the server must be stateless, meaning that every request from the client must contain all the information required for the server to process them. The server should not require any stored data to process the request.
- Use of directory-structure like URIs: REST APIs must use the URIs that are straightforward, properly structured, and easily understood.
- Data transfer in XML or JSON: The data transferred between the client and the service-exchange must be in XML or JSON format.

### **2.2.2.3. Communication On the Client-side**

REST web services must have a clear separation of client-side logic and server-side logic. A uniform interface separates clients and servers, which allows developers to work on the individual part of web application and improve one without affecting another.[4]

Clients and intermediaries should be able to cache server responses to avoid reuse of stale data in response to future requests. Clients also cannot assume a direct



connection to the server. In most cases, intermediaries between the client and the server serve the request-response cycle.

Figure 02 illustrates the Rest Api data exchange.

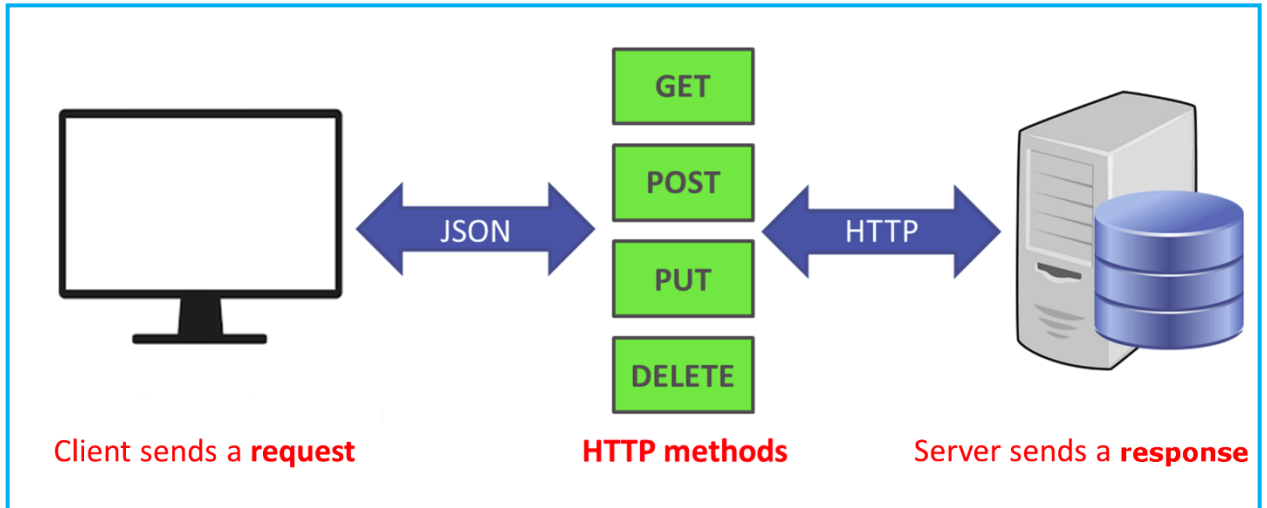


Figure 2.2: REST API

### 2.2.3. Single Page Application

Single Page Application is a web application, which fits into a single web page. In contrast to the traditional full-page loads, An SPA loads all the resources required to navigate throughout the web application on the first page load.

It then dynamically changes the contents as the user interacts with the application, so no full-page request will ever be made again. However, URLs are updated in the address bar of the browser with a hash tag following the name of the resources accessed.[9]

Figure 03 illustrates the distinction between the lifecycle of a traditional web page and an SPA web page.

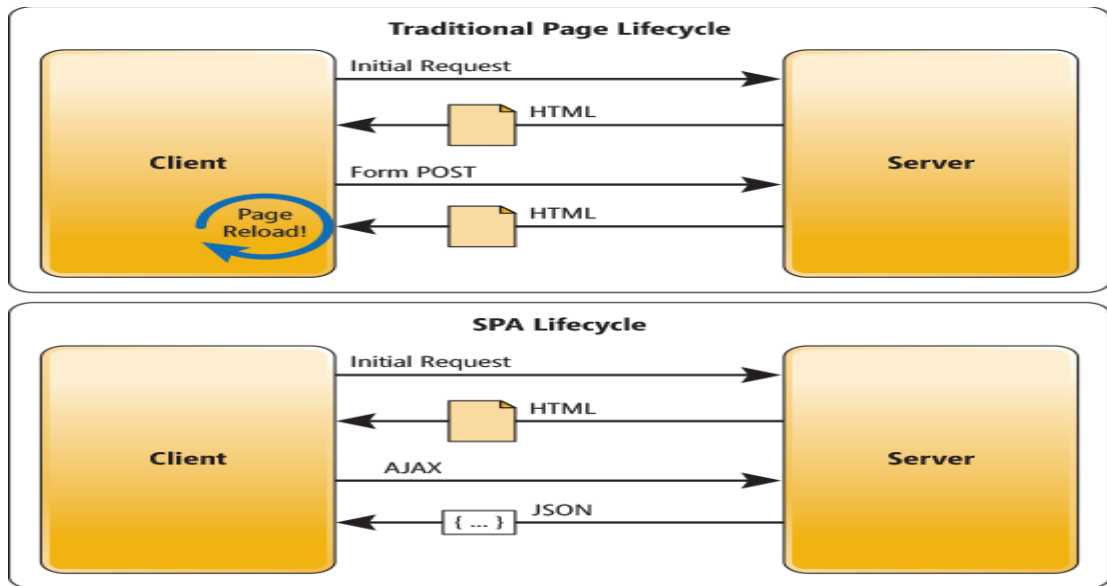


Figure 2.3: Comparison of traditional page lifecycle and SPA lifecycle

## 2.3. Full Stack

### 2.3.1. Overview

Various web and native applications are developed using ‘stacks’ of various technologies. The word ‘stack’ was first referred to the LAMP stack, Linux as the OS, Apache as the http server, MySQL the relational database and PHP as the programming language on which the application is developed.[4]

However, multipage applications are on the decent and single page applications are more popular because of their seamless user experience and lighter server calls which makes rendering the part of page easier without refreshing the page. So, the front-end frameworks or libraries those can produce single page applications are on demand. React being one of them.

MongoDB. NoSql databases are also called Not-Only-Sql because they also support SQL like query languages. MySQL (SQL) stores data in rows and columns, NoSQL databases store their data in different structures: key-value pairs, wide columns, graphs, or documents. Simplicity, scalability, flexibility, availability and speed of NoSql, databases like MongoDB have gained fame.[11]

Popular languages like PHP, Ruby and Python are among the early server side languages. Since, Javascript has made it possible to create dynamic front end, its development has also had an impact on back-end, in the form of NodeJS and ExpressJS.[9]

## 2.3.2. Full Stack JavaScript

### 2.3.2.1. JavaScript

JavaScript is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative styles.[4]

### 2.3.2.2. Rise of Full Stack JavaScript

Developers started to realize that use of two separate languages in the development of the client and the server was complicating the tasks of web programming. Several attempts were made to unify the two sides by creating client components on the server and compiling them into JavaScript, but they failed. The only solution to this problem was the implementation of JavaScript on the server-side, and Node.js was introduced.[9]

Node.js is actually the backbone of Full Stack JavaScript web development. It finally put the power of JavaScript on the server with the idea of non-blocking programming paradigm. Node.js became popular in a short time due to its easy-to-use components. It allowed the developers to quickly set up a server and start building applications on top of it. Several frameworks started to emerge to facilitate Node.js implementation such as Express and Connect.js. Express became the most prominent one. Node.js ecosystem continued to expand and a package manager like 'npm' was introduced.[4]

Figure 04 illustrates the full stack JavaScript with different technologies.

## Full Stack JavaScript

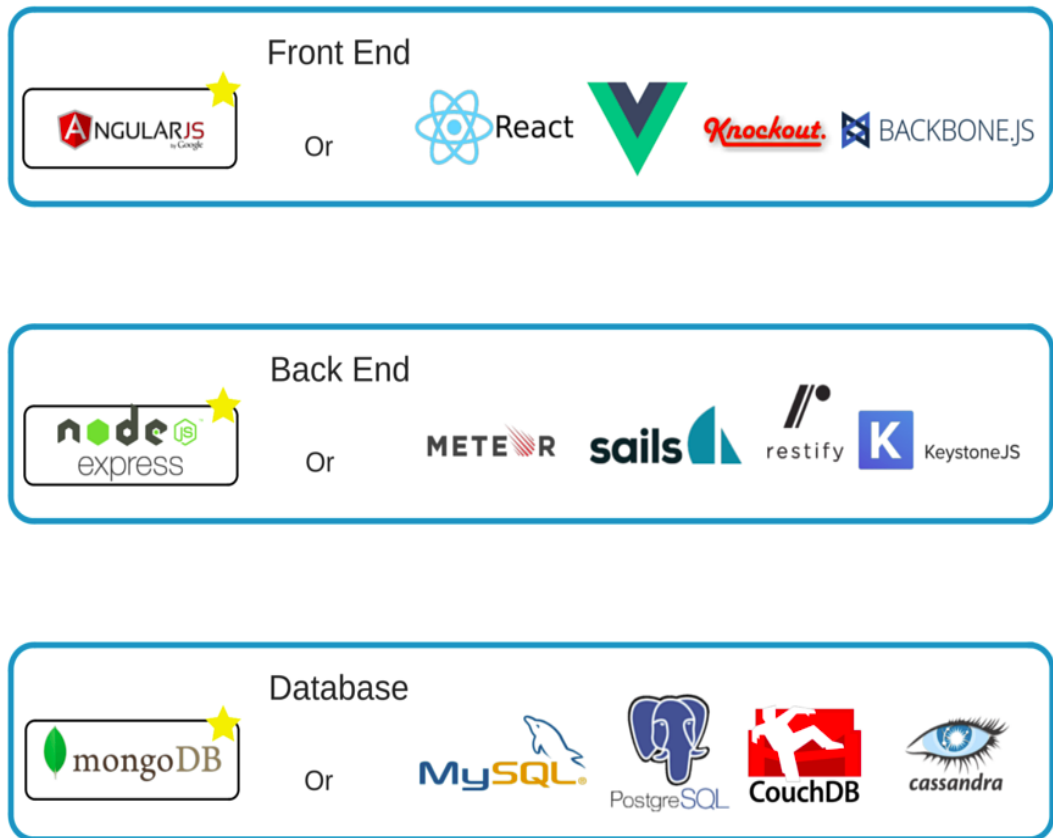


Figure 2.4: Full Stack JavaScript

### 2.3.2.3. Using The MERN Stack

Figure 05 illustrates our project architecture. We use the MERN Stack, mongoDB as database, the Back-end app with express and the Front-end app with react.js. To be described in detail in the next chapter.[9]

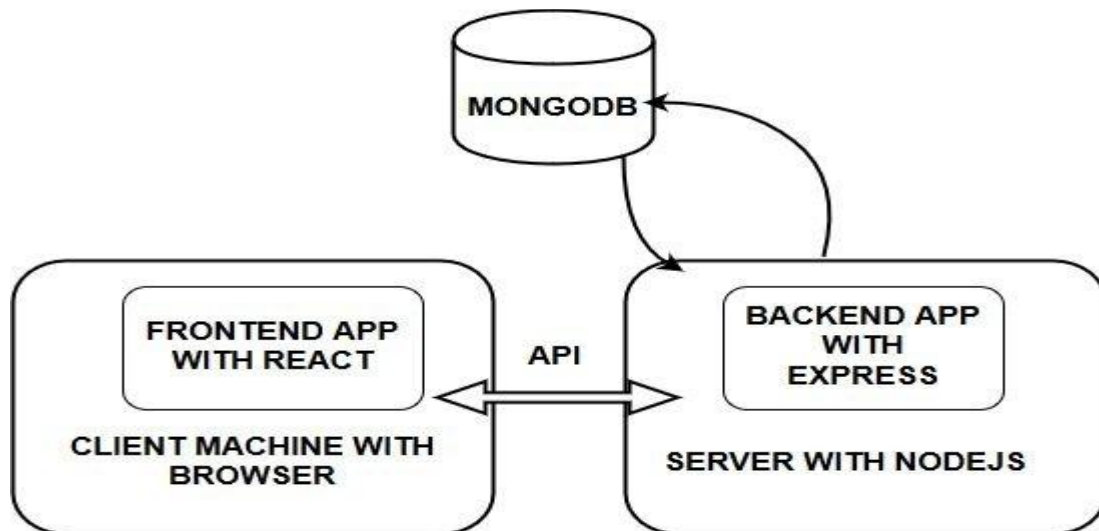


Figure 2.5: Block diagram of The Project Architecture

## 2.4. Tools

### 2.4.1. Visual Studio Code

VS Code is a popular source code editor by Microsoft. It can be used with many languages but since it is based on Electron, one of the Node.js frameworks, MERN development is easy and efficient. It is free for developers use and is open source under MIT license. Built-in terminal support makes it much easier for web-development.[11]

### 2.4.2. Postman

**Postman** is a scalable API testing tool. It started in 2012 as a side project by Abhinav Asthana to simplify API workflow in testing and development. API stands for Application Programming Interface, which allows software applications to communicate with each other via API calls[11].

### 2.4.3. MongoDB Compass

MongoDB Compass is the Graphic User Interface of MongoDB. It helps in analyzing the data content without the requirement of any prior knowledge of the query syntax in MongoDB. You can use it to explore the business data as a visual representation in the required environment. In addition, you can also use this Compass to manage indexes, optimize the performance of a query, and also to implement the validation of the document.[11]

## 2.5. MERN Stack

### 2.5.1. Overview

MERN stack as briefly discussed above consists of 4 independent frameworks and libraries, Mongo DB, Express JS, React JS and Node, hence the abbreviation MERN. Figure 06 described the individual aspects of MERN stack.[2]

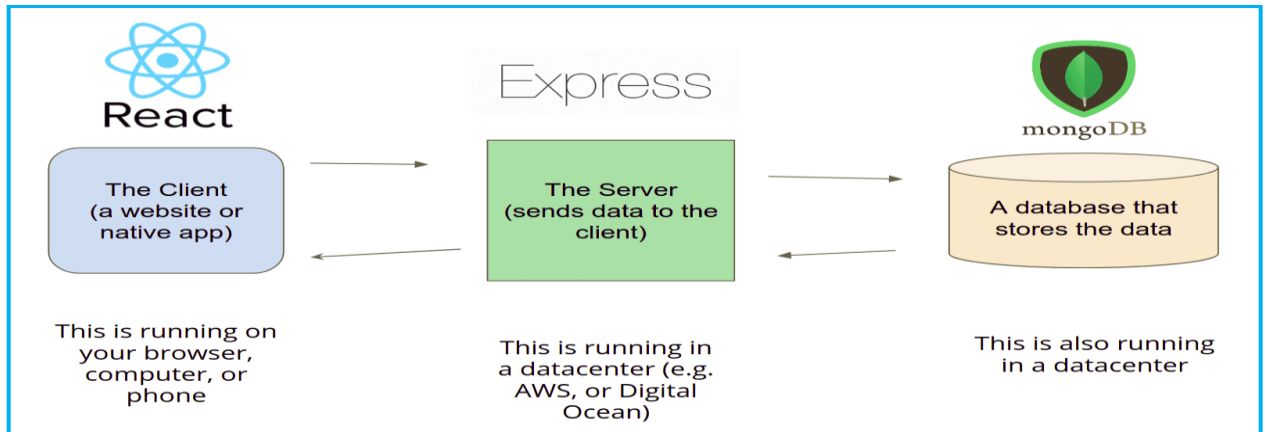


Figure 2.6: MERN stack architecture

### 2.5.2. Mongo DB

#### 2.5.2.1. NoSQL Database

Ever since the inception of the SQL database in 1980 by IBM based on the relational database model invented by Edgar Codd in 1973, relational databases, also known as RDBMS have been the predominant method of storing digital data. Historically the most popular databases have been Microsoft SQL Server, Oracle Database and MySQL.[2]

In SQL data is represented as rows in an Excel-like tables where each unique attribute of the schema is represented by a column in the table. The attributes are limited to SQL's predetermined set of basic data types such as integer, string, and date, which causes issues when the data that requires storing is irregular or dynamic.[22]

One solution that arose to solve this lack of flexibility is NoSQL, also known as 'Not Only SQL'. The term was coined by Carlo Strozzi in 1998. The NoSQL model offers more fluidity by not requiring a distinct schema, and by scaling horizontally. where the resources of a system consisting of multiple separate hardware units connected to a single logical cluster are increased by adding additional nodes to the cluster. Because each node in the cluster is running a copy of the software, additional nodes can be added without an interruption to the system's operation.[16]

NoSQL databases are commonly divided into four categories:

- Document databases
- Key-value databases
- Column databases
- Graph databases

NoSQL databases tend to have faster performance for read and write operations by not requiring joins to relate data and by dropping much of the overhead found in SQL operations.

#### **2.5.2.2. JavaScript Object Notation**

JSON is a format commonly used with browsers, which makes MongoDB an excellent fit for storing data for web applications, since the same JSON data can be transported between the browser and the database without having to be converted into another format between them, as is the case with SQL.[25]

JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.[2]

JSON is built on two structures:

- A collection of name/value pairs.
- An ordered list of values.

#### **2.5.2.3. Basic Information**

MongoDB is an open-source NoSQL database under the document database category, created in 2007 in the U.S by Dwight Merriman, Eliot Horowitz and Kevin Ryan. Mongo operates under the GPL and is written in C++. As the most popular non-relational database at present by a significant margin.

The SQL style of storing the entry divides the types of data that constitute the entry into 5 different schemas which are connected to the single entity through the unique identifiers (primary keys), whereas in MongoDB the same data is stored in a single JSON object, also

referred to as a ‘document’ containing all the inner components in further nested objects as attributes of the main container object.[25]

### 2.5.3. Node.js

#### 2.5.3.1. Basic Information

Node.js is an open source platform that utilizes Google Chrome’s JavaScript runtime V8 Engine. Node.js can be characterized as being to JavaScript what the JRE is to java. Node.js compiles and executes JavaScript code inside of a virtual machine and thereby enables JavaScript code to be run on the server side.[26]

#### 2.5.3.2. The Node.js event loop

The event loop is what allows Node.js to take advantage of multi-threaded system kernels while still maintaining non-blocking I/O. What is meant by I/O in the context of Node.js is usually accessing external resources like disks or network resources, which are the most expensive due to the time they take to complete.[22]

Figure 07 illustrates the Nodejs event loop.

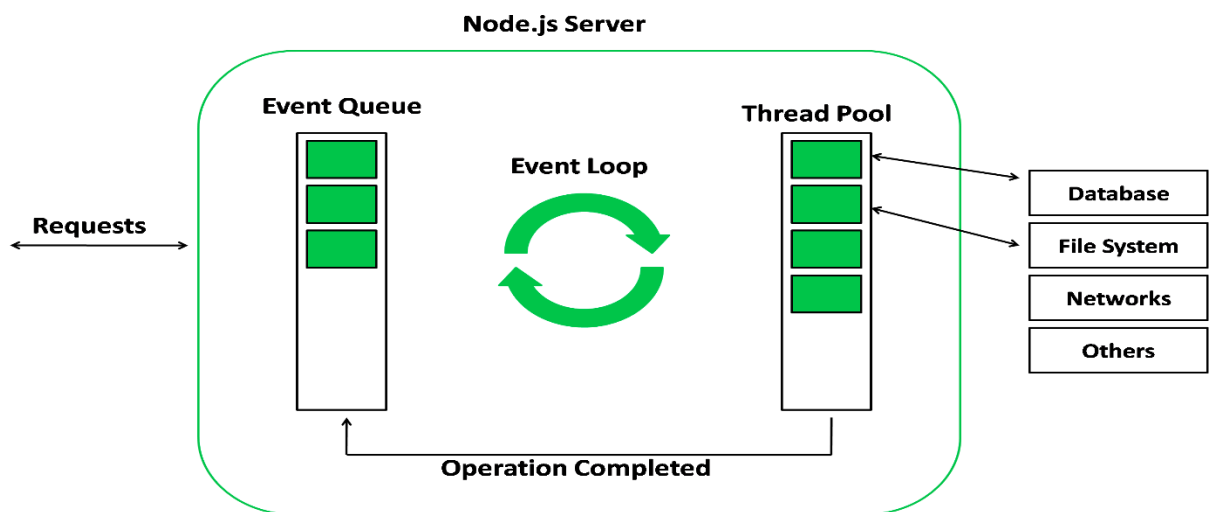


Figure 2.7: Node.js event loop

The Node.js process is a loop that performs polling and blocking calls to the system kernel on a constant basis while active. The operating system kernel in turn notifies Node.js when an operation is complete, after which its associated callback function is added to the poll queue for eventual execution. Node.js exits when it runs out of events to process, but in the context of a web server where listening for new requests is in and of



itself a series of events, the process can be conceptualized essentially as a closed while-loop, repeating its internal phases continually.

The Node.js event loop is internally implemented by a C library called libuv, which is a multi-platform open source support library for base level asynchronous I/O management. Libuv is primarily designed for its use in Node.js and can take advantage of the native polling queue mechanisms of each operating system to achieve high performance levels and effective offloading of I/O operations to the system kernel whenever possible.[2]

### **2.5.3.3. Node Package Manager**

NPM is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry.[2]

NPM comes with Node.js upon installation and is accessed from the command line. Installing packages can be done globally or locally through the npm install command. The code block itself is then fetched from the NPM registry via HTTP and saved into a node\_modules folder, which is created if it does not exist. The name of the package on the website always corresponds to its name in the registry, making it directly accessible using the install command 'npm install <package name>'. [2]

### **2.5.4. Express.js**

ExpressJS is the web application framework of Javascript that runs the back-end application code. Express runs as a module within the Node.js environment. It can handle the routing of requests to the right parts of the application. With a myriad of HTTP utility methods and middleware, creating a robust API is quick and easy with the use of Express.js.[26]

### **2.5.5. React.js**

React is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. It runs in web browsers. React breaks front-end application down into components. Each component can hold its own state and a parent can pass its state down to its child components and those components can pass changes back to the parent with callback functions.[22]

React components are typically implemented using JSX that is an extension of JavaScript that allows HTML syntax to be embedded within the code. React can be used as a base in the development of single-page or mobile applications. Complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API.

React.js emerged as a solution to the problem, delivering the application components in an initial JavaScript bundle and then efficiently managing renders to the DOM, allowing for easily reusable and customizable HTML views.[16]

## **2.6. Mobile App Integration**

### **2.6.1. Native Apps**

Native apps are what comes to most of our minds when we think of mobile apps and are downloaded from the App Store or Google Play. What distinguishes native apps from mobile web and hybrid apps is that they are developed for specific devices. For instance, Android apps are written in Java and iPhone apps are written in Objective-C.[26]

The advantage of choosing a native app is that it is the fastest and most reliable when it comes to user experience. Native apps can also interact with all of the device's operating system features, such as the microphone, camera, contacts lists, etc. However, a bigger budget is required if you want to build your app for multiple platforms (i.e. iPhones and Android) and to keep your native app updated.[22]

### **2.6.2. Hybrid Apps**

A hybrid app combines elements of both native and web applications. Hybrid apps can be distributed through the app stores just like a native app, and they can incorporate operating system features. Like a web app, hybrid apps can also use cross-compatible web technologies.[25]

Hybrid apps are typically easier and faster to develop than native apps. They also require less maintenance. On the other hand, the speed of a hybrid app will depend completely on the speed of the user's browser. This means hybrid apps will almost never run as fast as a native app runs.

The advantage of hybrid apps is that you can build them on a single base, which allows you to add new functionalities to multiple versions of your app. With native apps, you will need to replicate every new feature you want to introduce for each platform.[16]

### **2.6.3. Cross Platform Software**

#### **2.6.3.1. Overview**

Cross-platform software is a type of software application that which works on multiple operating systems or devices, which are often referred to as platforms. A platform means an operating system such as Windows, Mac OS, Android or iOS. When a software application works on more than one platform, the user can utilize the software on a wider choice of devices and computers.[2]

#### **2.6.3.2. Dart**

Dart is an object-oriented programming language developed by google. Whilst technically not restricted, it is primarily used for creating front-end user interfaces for the web (with angularDart or flutter for Web) and mobile apps (flutter).

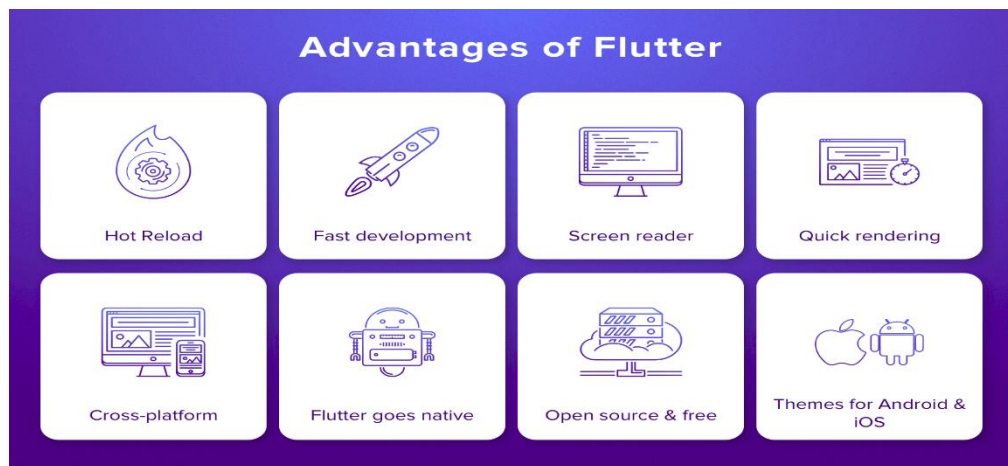
It is under active development, compiled to native machine code (when used for building mobile apps), inspired by modern features of other programming languages (mainly Java, JavaScript, C#) and strongly typed.

As already mentioned, dart is a compiled language. That means, that your code is not executed like you write it, but instead, a compiler parses and transforms it to machine code.

#### **2.6.3.3. Flutter**

Flutter is actually a combination of two things an sdk (software development kit) and a framework built. Flutter aims to make cross platform development (of mobile apps) a breeze. The goal is to allow you to have one code-base that generates apps for multiple platforms (like iOS and android).

Figure 08 illustrates the Advantages of flutter.



*Figure 2.8: Advantages of Flutter*

To achieve this goal, flutter is compiled to native machine code (to achieve good performance) and offers many utility tools and features to speed up your development work.

## **2.7. Real-Time Functionality**

### **2.7.1. WebSockets**

Modern web applications are incorporating real-time capabilities, which enable the application to continuously present the user with recently updated information. Unlike traditional applications, in real-time applications the common roles of browser and server can be reversed since the server needs to update the browser with new data, regardless of the browser request state. This means that unlike the common HTTP behavior, the server won't wait for the browser's requests. Instead, it will send new data to the browser whenever this data becomes available.[22]

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection.

The WebSocket API is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.

### **2.7.2. Socket.io**

Socket.IO enables real-time, bidirectional and event-based communication. It works on every platform, browser or device, focusing equally on reliability and speed. Sockets have

traditionally been the solution around which most real-time chat systems are architected, providing a bi-directional communication channel between a client and a server.

Socket.IO is a JavaScript library that helps improving work with WebSockets. It consists of two parts, server part (for Node.JS) and client part (for web browsers). Both of them have similar APIs based on event-driven architecture. Socket.IO allows using additional features such as sending data to a large number of sockets at the same time (broadcasting) or storing the data.[22]

## **2.8. Conclusion**

In this chapter, we talked about our project structure and the technologies we have used in it. We presented the various web stacks especially the JavaScript stacks. We compared these stacks languages, frameworks and we decided to use the MERN stack to make the FindMe application

In the next chapter, we will analyze our project and we will describe the Find Me system logic using diagrams and flowcharts.

## Chapter 3 Find Me System Analysis And Design

### 3.1. Introduction

In this chapter, we will start with a preliminary study on the FindMe application. Then we will describe the system structural and the project model using the first three levels of the data flow diagram.

This chapter provides a detailed explanation of the authentication, the search mechanism, the account management, the track manipulation and the chat mechanism. It also explains the database architecture of the FindME system.

### 3.2. Preliminary study

#### 3.2.1. System Description

As we mentioned in the introduction, we have made a Web application (MERN app) and a cross platform application work on both iOS and Android that help find the missing people that is called 'FindMe'.

Starting with the MERN app, 'FindMe' Web is prompted with a landing page where users can enter their details in the sign-up form or they can also sign in with Facebook to be authorized to use 'FindMe'. After signing in they are routed to the 'home-page' where each user sees other users public 'tracks' (track is a collection of an information about the missing person) and the people that are found in the last 24 hours. They have the option of filtering the 'tracks' to see them based on the location and if the 'track' is found or not. Here the user can search automatically using a photo of someone; he can find this option in the landing page because you do not need to be authorized to look for someone.

In 'FindMe' there is a navigation bar that allows the users to navigate easily, the first link 'Home' is for homepage, the second link 'My Tracks' takes to 'my-tracks-page' where the user can create, edit and delete his own 'track', this 'track' contains a photo and an information about this lost person and the user who wants to find him. 'They Found Me' link takes to a page that display the related 'tracks' to the found persons where the user can confirm or delete this temporary 'track'. All the 'tracks' that are related to this person will be deleted.

The navbar contains three links represented by icons; the notification link takes to the 'notification-page' where we notify the user that a person in his 'tracks' is found. The chat link takes to the 'chat-page' where the user send and receive messages and contact other

user, every conversation is related to a 'track'. If the 'track' is deleted this conversation will be blocked. There is a logout link and an account link that takes the user to a page where he can edit his email, password and delete his account.

### **3.2.2. System Functionality**

In this section, we present our system functionality and all the actions that the user can do. The system actions are divided into 4 parts. Note that 'FindMe' is a real time application.

#### **3.2.2.1. Account Actions**

The account actions:

- Register (sign-up).
- Login with email or Facebook.
- Logout.
- Password recovery using the account email.
- Change account email or password.
- Delete account.

#### **3.2.2.2. Track Actions**

The 'track' actions:

- Create 'track'.
- Edit 'track'.
- Delete 'track'.
- Automatic search.
- Confirm Temporary 'track'.
- Delete temporary 'track'.

#### **3.2.2.3. Chat Actions**

The chat actions:

- Send message.
- Remove message.
- Block conversation

#### **3.2.2.4. Under the Hood Actions**

The 'under the hood' actions:

- Get the user account.
- Send validation emails.
- Get the user 'tracks'.
- Get the users public 'tracks'.
- Filter the users public 'tracks'.
- Get the people that are found in the last 24 hours.
- Get the conversation messages.
- Get the conversations last messages.
- Notify the users.

### **3.2.3. Project Requirements**

The project requires:

- Single page Web application.
- The application should be built with the MVC architecture.
- Node.js backend with Express and MongoDB.
- ReactJS front-end.
- Good application structure so our app can grow.
- Responsive Web application.
- Cross platform application work on iOS and Android.

Based on the requirements, the Web application was built in the MVC architecture. There is a complete separation between the server-side and the client-side logic. The model is implemented in the server, and the view and the controller are implemented in the client.

Moreover, the application (web and mobile) does not communicate directly to the server, but uses REST APIs to feed the data. The Web application uses the routes created in ReactJs to navigate throughout the web pages but all data are served by updating a single page.

## **3.3. Structural/ Procedural design**

### **3.3.1. Flowchart diagram**

This is a visual representation of the sequence of steps and decisions needed to perform a process. Figure01 represents the sequence of activities a user undergo when using the application based on its functionalities.



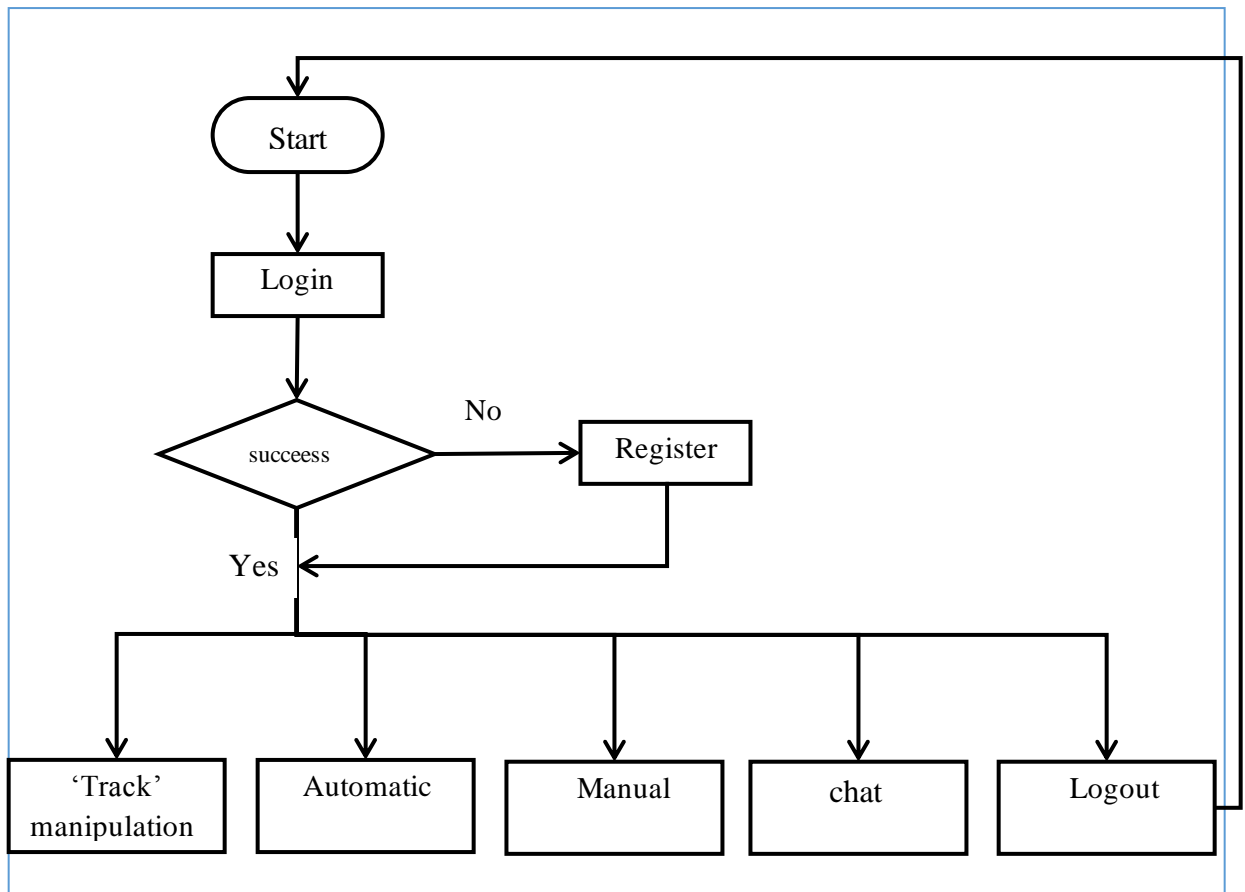


Figure 3.1: Flowchart diagram for the application

### 3.3.2. Object Oriented Design

Use cases share different kinds of relationships. Defining the relationship between two use cases is the decision of the software analysts of the use case diagram.

A relationship between two use cases is basically modeling the dependency between the two use cases.

The reuse of an existing use case by using different types of relationships reduces the overall effort required in developing a system.

A use case diagram is usually simple. It summarizes in details the system's users and their interactions with the system. It does not show the detail of the use cases:

- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case. Figure represents the Find-Me use case diagram.

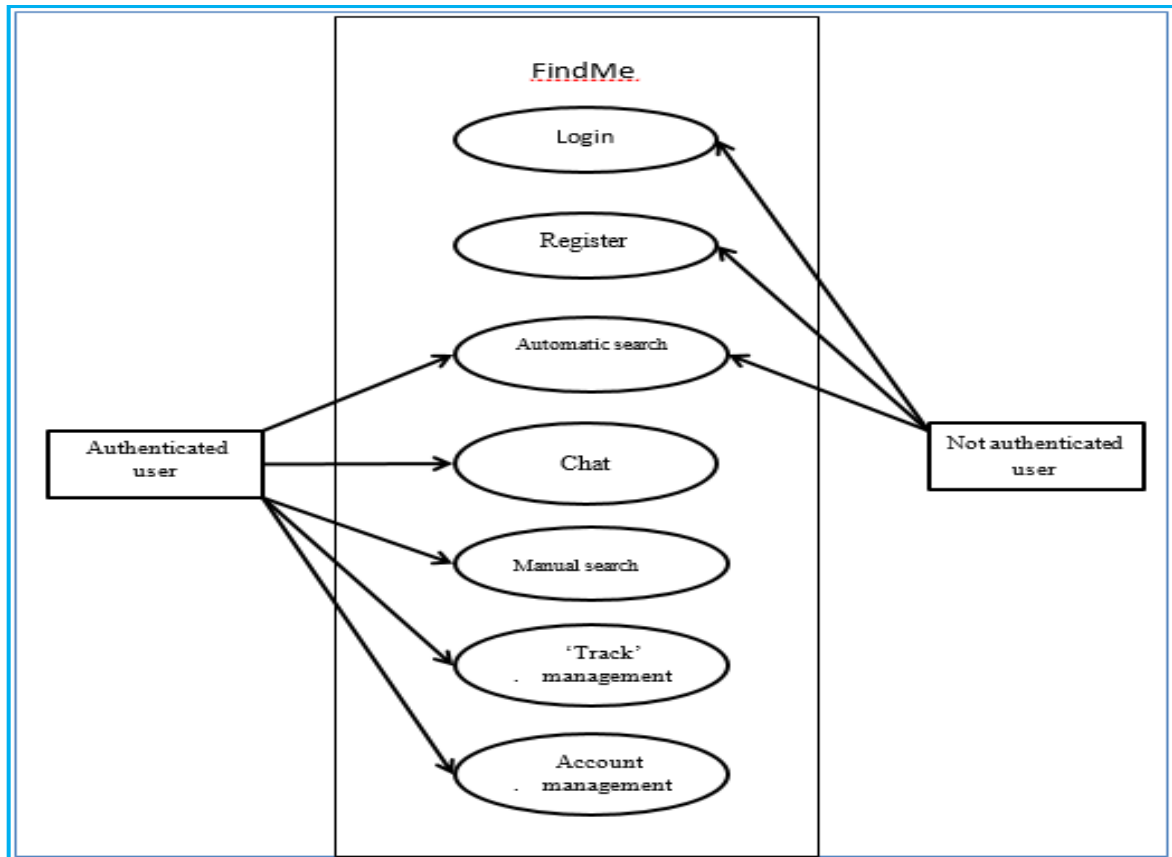


Figure 3.2: Find Me Use Case Diagram

### 3.4. Project Model

#### 3.4.1. Data Flow Diagram (DFD)

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multilevel DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO.

That is why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

### 3.4.2. DFD Level 0

A level 0 data flow diagram, also known as a context diagram, shows a data system as a whole and emphasizes the way it interacts with external entities. This DFD level 0 example shows how such a system might function within a typical retail business.

Figure 03 illustrates our Project (FindMe) zero level DFD.

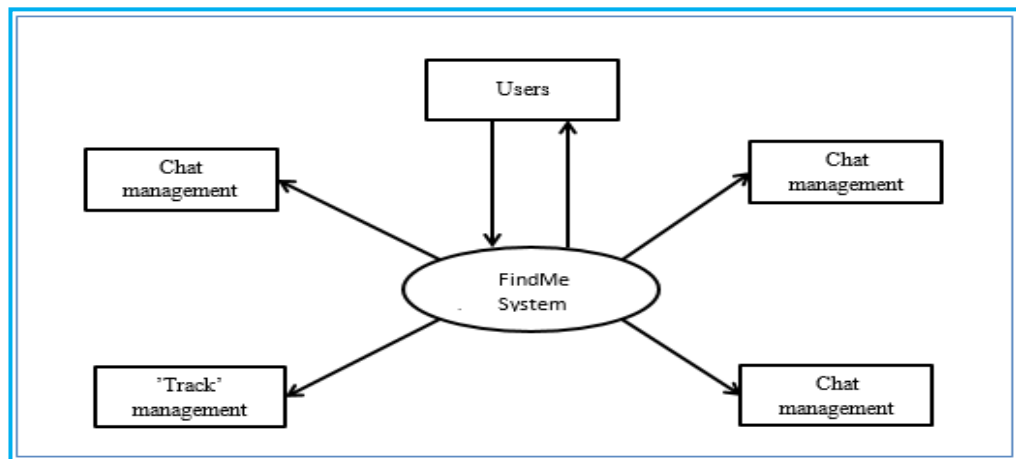


Figure 3.3: FindMe Zero Level DFD

### 3.4.3. DFD Level 1

A level 1 data flow diagram is more detailed than a level 0 DFD but not as detailed as a level 2 DFD. It breaks down the main processes into sub-processes that can then be analyzed and improved on a more intimate level.

Figure 04 illustrates our Project (FindMe) first level DFD. The visitor is just a not authenticated user.

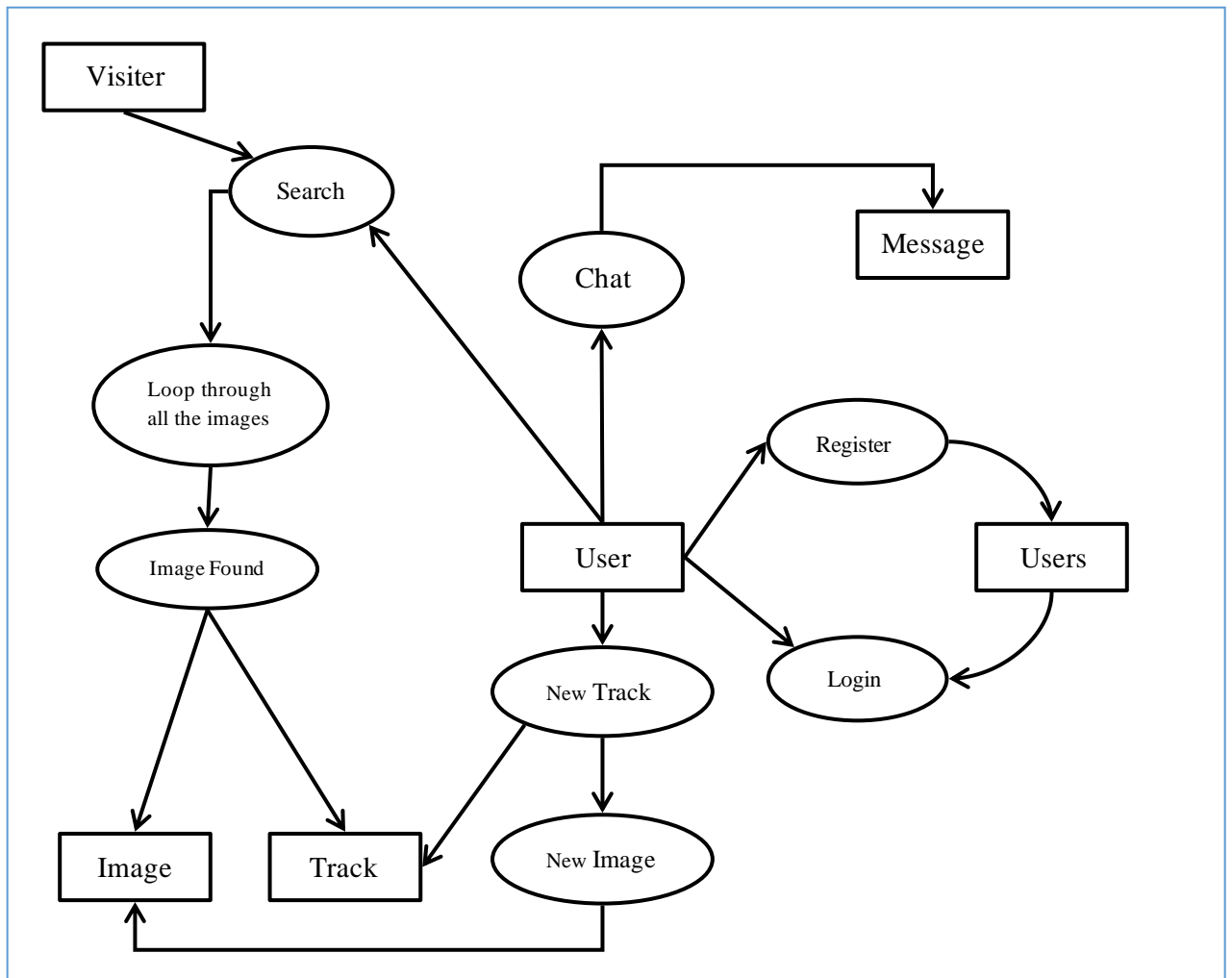


Figure 3.4: FindMe First Level DFD

### 3.4.4. DFD Level 2

#### 3.4.4.1. Overview

A level 2 data flow diagram offers a more detailed look at the processes that make up an information system than a level 1 DFD does. It can be used to plan or record the specific makeup of a system.

#### 3.4.4.2. Authentication

##### 3.4.4.2.1. Definition

In computing, authentication is the process of verifying the identity of a person or device. A common example is entering a username and password when you log in to a website. Entering the correct login information lets the website know:

- Who you are.

- That it is actually you accessing the website.

While a username/password combination is a common way to authenticate your identity, many other types of authentication exist. For example, you might use a four or six-digit passcode to unlock your phone. A single password may be required to log on to your laptop or work computer. Every time you check or send email, the mail server verifies your identity by matching your email address with the correct password. Your web browser or email program often saves this information so you do not have to enter it each time.

Biometrics may also be used for authentication. For example, many smartphones have a fingerprint sensor that allows you to unlock your phone with a simple tap of your thumb or finger. Some facilities have retinal scanners, which require an eye scan to allow authorized individuals to access secure areas. Apple's Face ID authenticates users by facial recognition.

#### **3.4.4.2.2. JSON Web Token (JWT)**

JWT is a JSON web object, which helps creating a safe data communication between two parties, which in application's context are the application's server, and the end-user. The token is composed of a header, a payload and a signature.

Header provides information how signature is to be computed. Payload is the data stored in the token, like the user information. The payload is encrypted or hashed to produce a signature. It is used for the authentication of users in the application.

Figure 05 illustrates the way JWT attaches with the application to provide security and privacy to the contents.

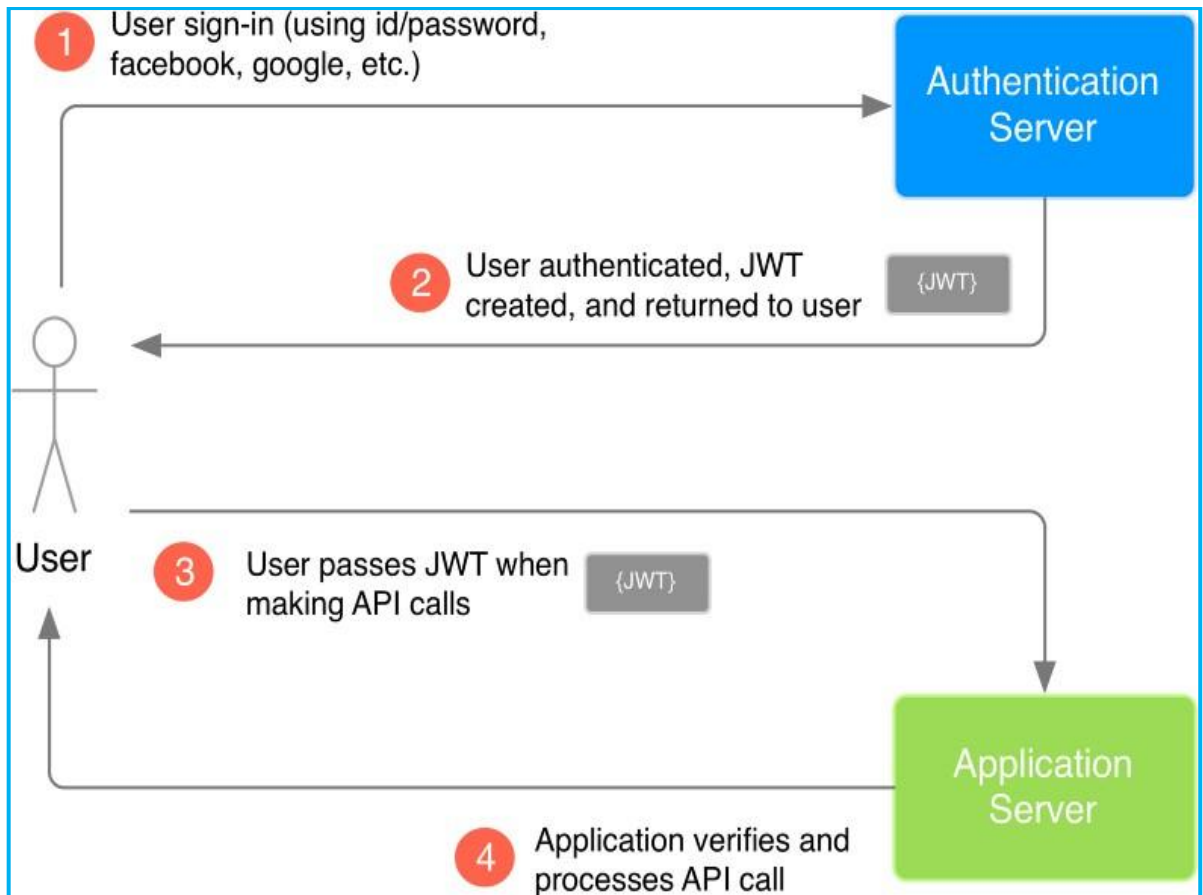


Figure 3.5: JWT to access the application server

### 3.4.4.2.3. Authentication second level DFD

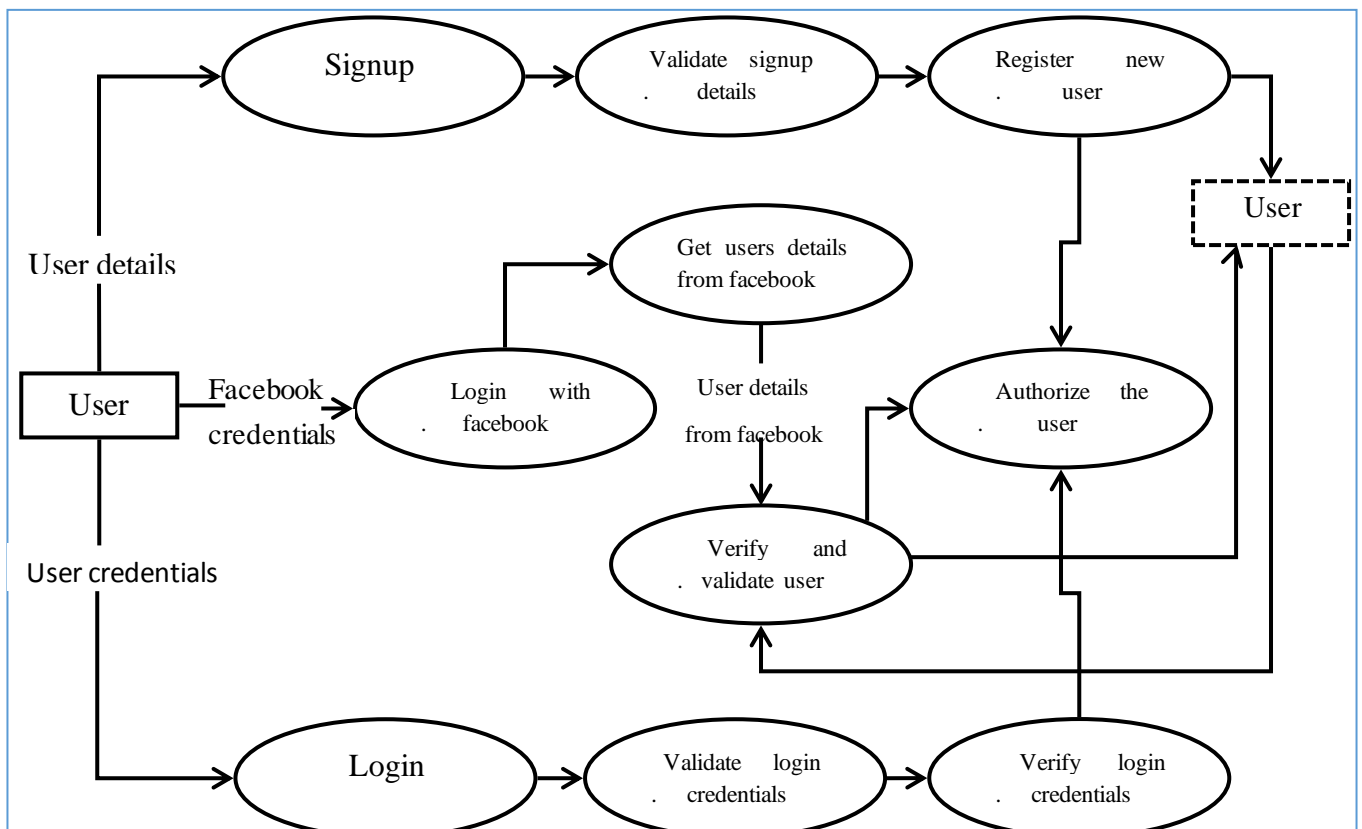


Figure 3.6: Authentication Second level DFD

Figure 06 illustrates our system authentication second level DFD.

### 3.4.4.3. Account Management

The account management system data flow diagram is often used as preliminary step to create an overview of the account without going into great details. It normally consists of overall application data-flow and processes of the account process.

Figure 07 illustrates our system account management second level DFD.

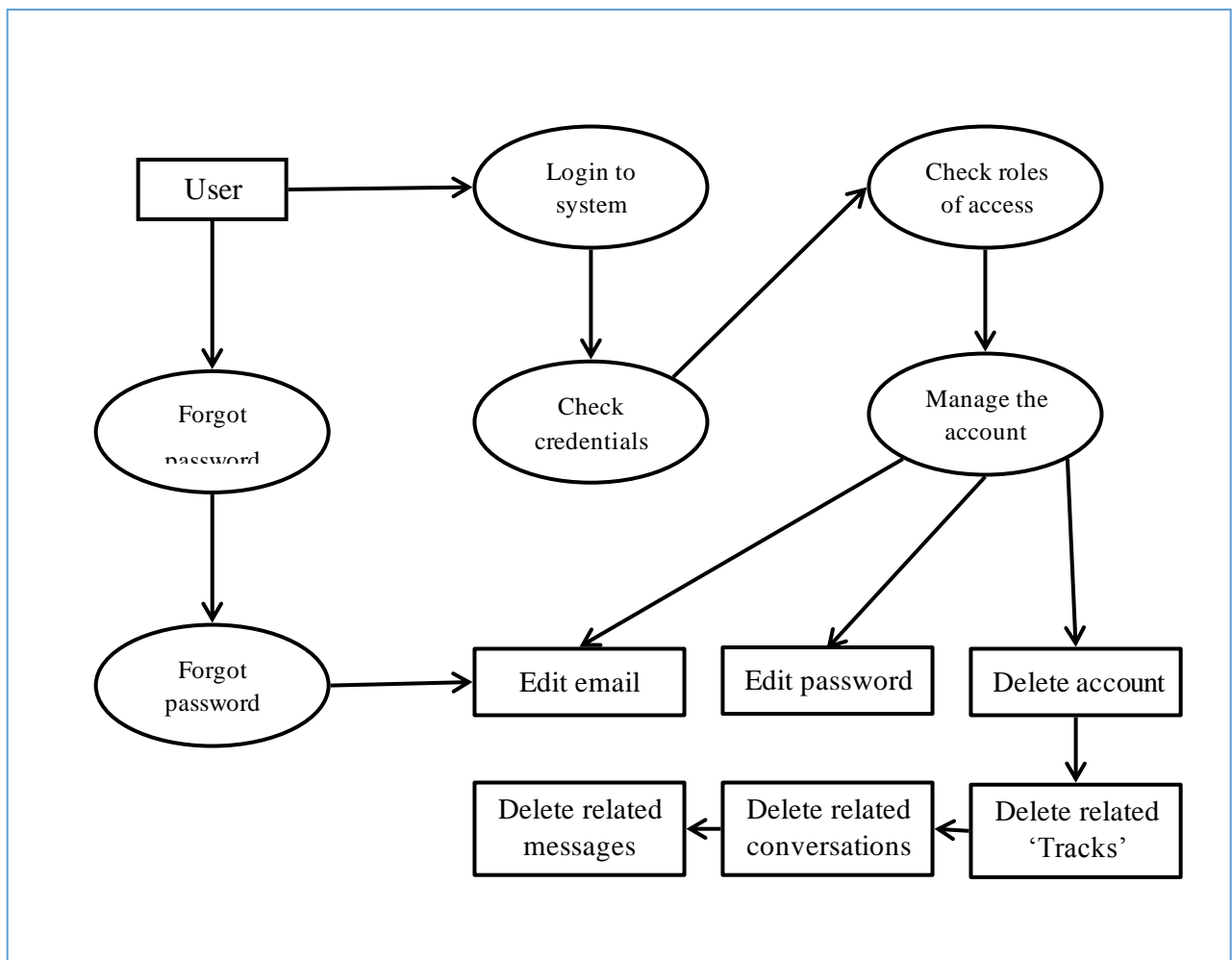


Figure 3.7: Account management Second level DFD

### 3.4.4.4. Track Management

Here we present our system 'track' manipulation .Every authenticated user has the ability to create,edit ,and delete his own track , and make it public to be seen by the other users, or private that only the owner can see it. Every track must have a phone number, the lost

person full name and one picture to make the communication and the search possible. The 'track' can have other optional information like the date of birth and the gender.

Figure 08 illustrates the 'Track' management second level DFD.

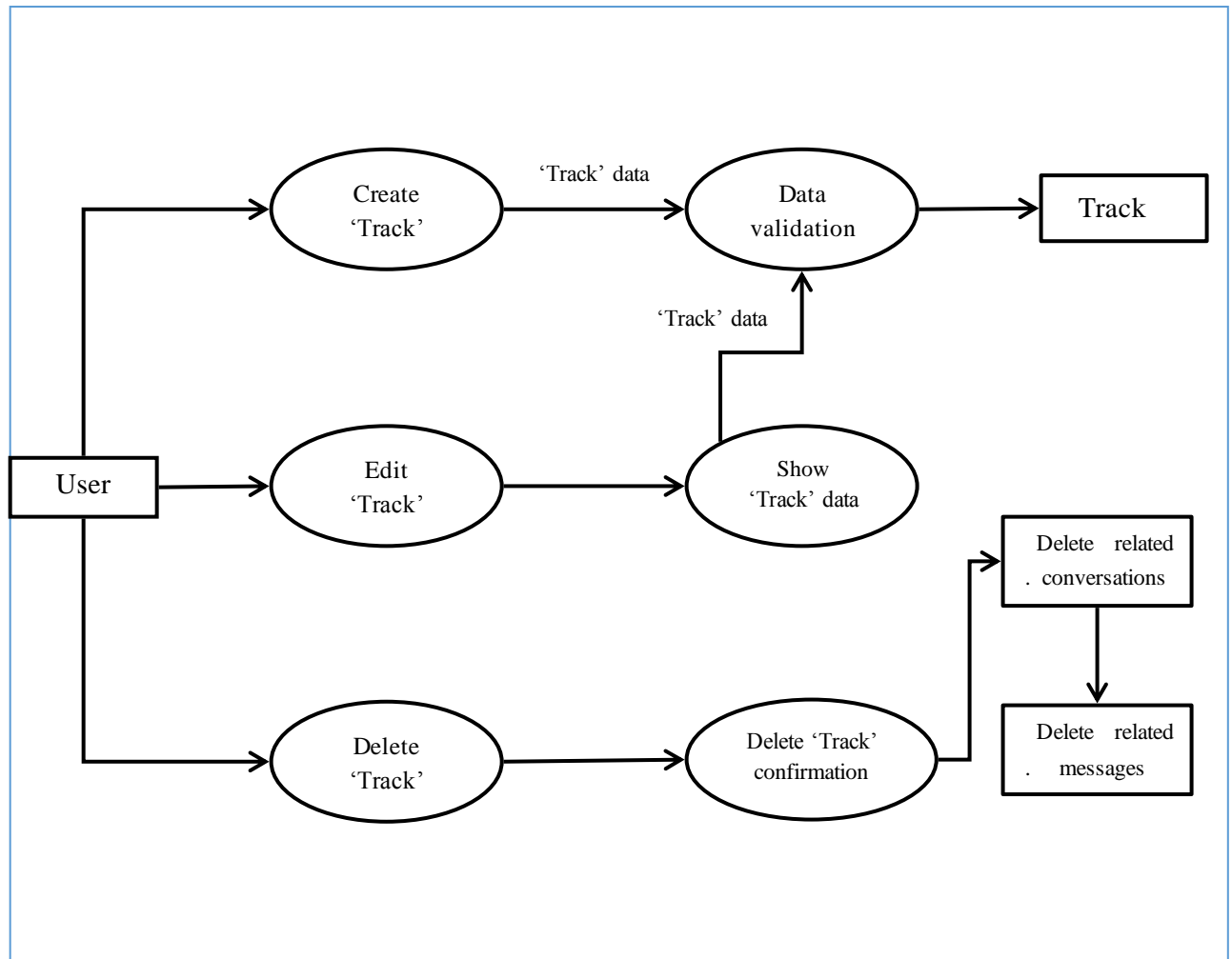


Figure 3.8: 'Track' management Second level DFD

#### 3.4.4.5. Communication System

FindMe has a communication system to make the user experience much better. The principle is simple, each public 'track' has a connect button where the user can communicate with the 'track' creator. When the user push the connect button and send the first message a new conversation will be created related to this 'track', if the track is deleted or the lost person is found this conversation will be blocked automatically.

Users can send, delete messages and block the conversation. FindMe has a real time communication System where the two members (sender/receiver) view all conversation updates at the same time.



Figure 09 illustrates our chat management second level DFD.

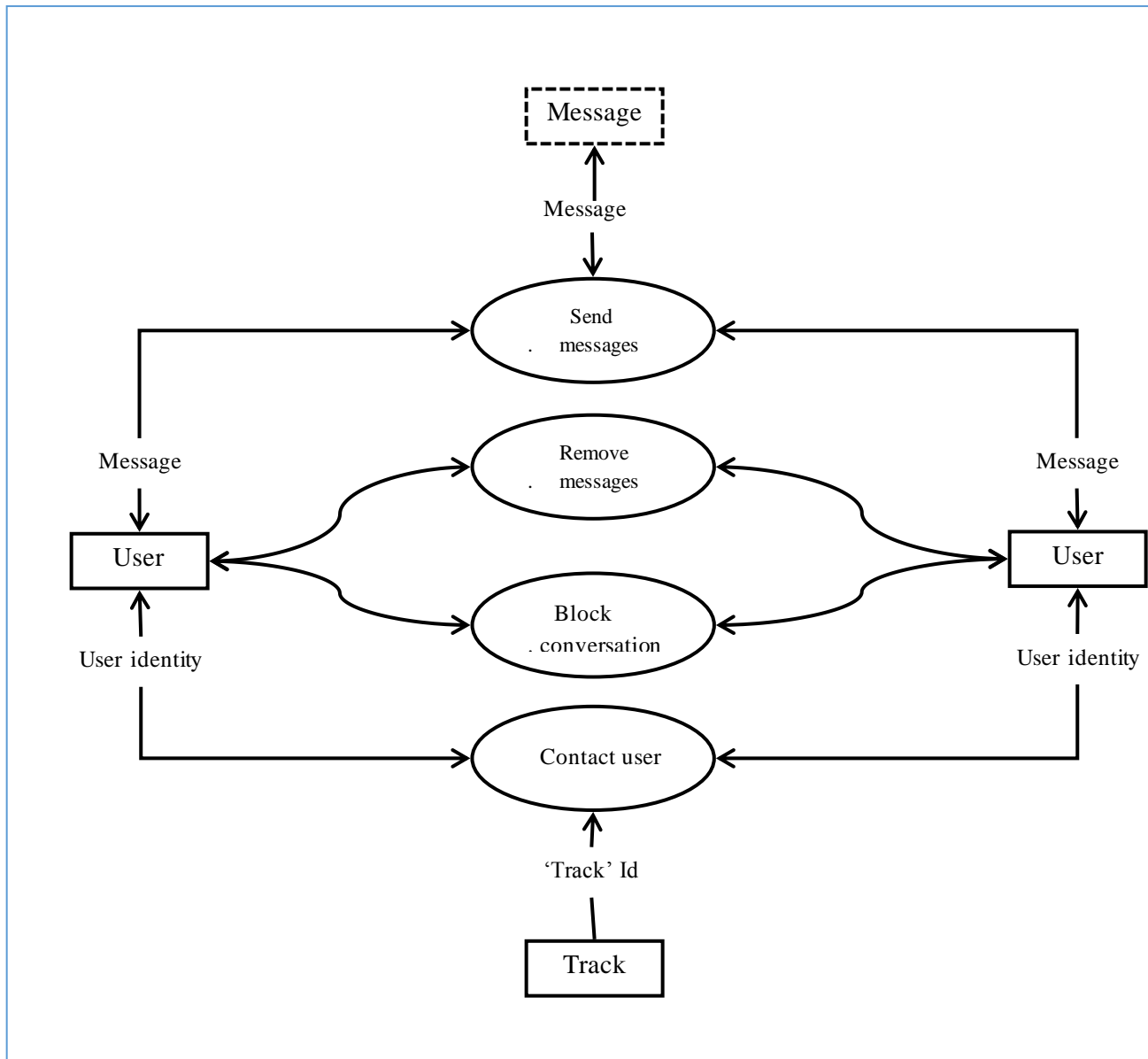


Figure 3.9: Second Level DFD of inter-user communication system

### 3.4.4.6. Face Recognition

#### 3.4.4.6.1. Machine Learning

Machine learning is the art of science of getting computers to act as per the algorithms designed and programmed. Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.[17]

Many researchers think machine learning is the best way to make progress towards human-level AI. Machine learning includes the following types of patterns:

- Supervised learning pattern
- Unsupervised learning pattern

#### **3.4.4.6.2. Deep Learning**

Deep learning is a subfield of machine learning where concerned algorithms are inspired by the structure and function of the brain called artificial neural networks.

All the value today of deep learning is through supervised learning or learning from labelled data and algorithms.[17]

Each algorithm in deep learning goes through the same process. It includes a hierarchy of nonlinear transformation of input that can be used to generate a statistical model as output.

Consider the following steps that define the Machine Learning process:

- Identifies relevant data sets and prepares them for analysis. Chooses the type of algorithm to use.
- Builds an analytical model based on the algorithm used.
- Trains the model on test data sets, revising it as needed. Runs the model to generate test scores.

#### **3.4.4.6.3. Difference between Machine Learning and Deep Learning**

##### **3.4.4.6.3.1. Hardware Dependencies**

Deep learning algorithms are designed to heavily depend on high-end machines unlike the traditional machine learning algorithms. Deep learning algorithms perform a number of matrix multiplication operations, which require a large amount of hardware support.[17]

##### **3.4.4.6.3.2. Amount of Data**

Machine learning works with large amounts of data. It is useful for small amounts of data too. Deep learning on the other hand works efficiently if the amount of data increases rapidly.

Figure 10 shows the working of machine learning and deep learning with the amount of data:

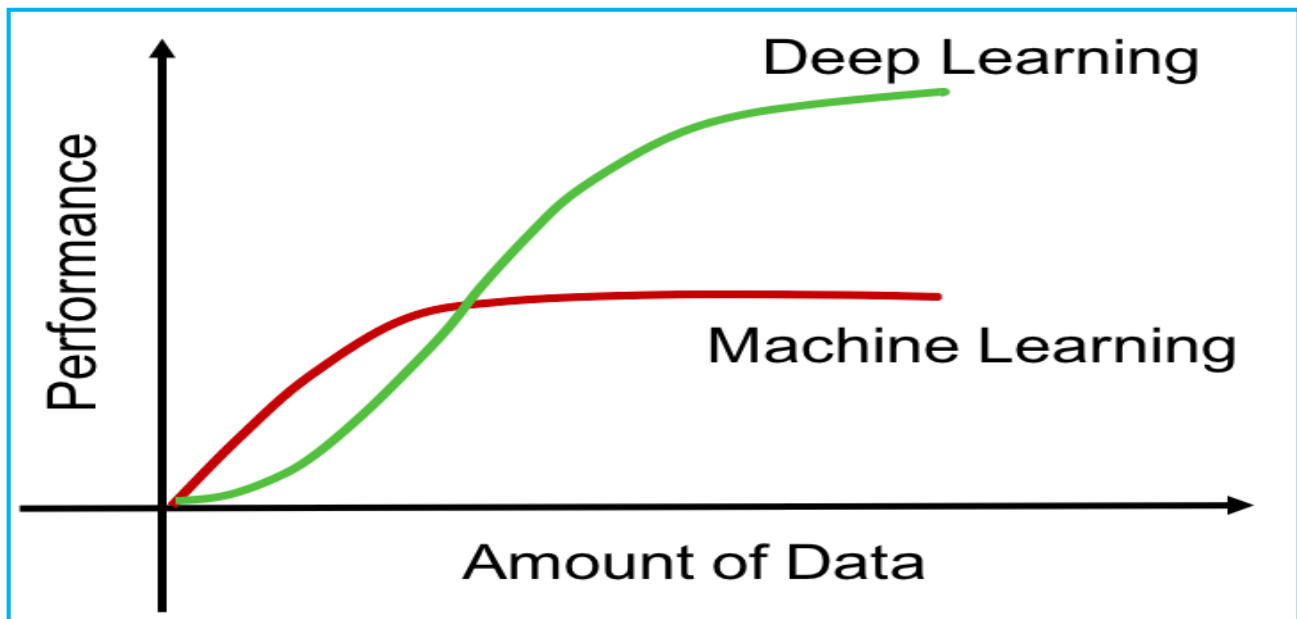


Figure 3.10: Machine Learning and Deep Learning Amount of Data

#### 3.4.4.6.4. Problem Solving Approach

The traditional machine learning algorithms follow a standard procedure to solve the problem. It breaks the problem into parts, solve each one of them and combine them to get the required result. Deep learning focuses in solving the problem from end to end instead of breaking them into divisions.[17]

##### 3.4.4.6.4.1. Execution Time

Execution time is the amount of time required to train an algorithm. Deep learning requires a lot of time to train as it includes many parameters, which takes a longer time than usual. Machine learning algorithm comparatively requires less execution time.

##### 3.4.4.6.4.2. Interpretability

Interpretability is the major factor for comparison of machine learning and deep learning algorithms. The main reason is that deep learning is still given a second thought before its usage in industry.

##### 3.4.4.6.4.3. Future Trends

With the increasing trend of using data science and machine learning in the industry, it will become important for each organization to inculcate machine learning in their businesses.

Deep learning is gaining more importance than machine learning. Deep learning is proving to be one of the best techniques in state-of-art performance.

Machine learning and deep learning prove beneficial in research and academics field.

#### **3.4.4.6.5. Machine Learning Model**

A machine learning model is made of up of nodes which are similar to Neurons in our human brains. These neurons are structured as layers. There is an Input Layer, Hidden Layer, and Output Layer.

The Input layer takes the input, pre-processes it for the next layers and sends it to the hidden layer.

The hidden layer itself can have multiple layers within itself, which do the inferencing/processing of the input to get to output. There is some 'weight' associated with each node of the model (just like Neurons in our brain). These weights are tuned while the model is being trained until we get the desired accuracy in the output.[17]

The output layer gets the inferred output from the Hidden layer and gives the output in the desired format.

#### **3.4.4.6.6. TensorFlow**

##### **3.4.4.6.6.1.Overview**

TensorFlow is an open source framework developed by Google researchers to run machine learning, deep learning and other statistical and predictive analytics workloads. Like similar platforms, it's designed to streamline the process of developing and executing advanced analytics applications for users such as data scientists, statisticians and predictive modelers.

The TensorFlow software handles data sets that are arrayed as computational nodes in graph form. The edges that connect the nodes in a graph can represent multidimensional vectors or matrices, creating what are known as tensors. Because TensorFlow programs use a data flow architecture that works with generalized intermediate results of the computations, they are especially open to very large-scale parallel processing applications, with neural networks being a common example.

TensorFlow applications can run on either conventional CPUs or GPUs, as well as Google's own TPUs, which are custom devices expressly designed to speed up TensorFlow

jobs. Google's first TPUs, detailed publicly in 2016, were used internally in conjunction with TensorFlow to power some of the company's applications and online services, including its RankBrain search algorithm and Street View mapping technology.[17]

#### 3.4.4.6.2.Tensor Data Structure

Tensors are used as the basic data structures in TensorFlow language. Tensors represent the connecting edges in any flow diagram called the Data Flow Graph. Tensors are defined as multidimensional array or list.[17]

Tensors are identified by the following three parameters:

- **Rank:** unit of dimensionality described within tensor is called rank. It identifies the number of dimensions of the tensor. A rank of a tensor can be described as the order or n-dimensions of a tensor defined.
- **Shape:** the number of rows and columns together define the shape of Tensor.
- **Type:** type describes the data type assigned to Tensor's elements.

A user needs to consider the following activities for building a Tensor:

- Build an n-dimensional array
- Convert the n-dimensional array.

Figure 11 illustrates the various dimensions of TensorFlow.

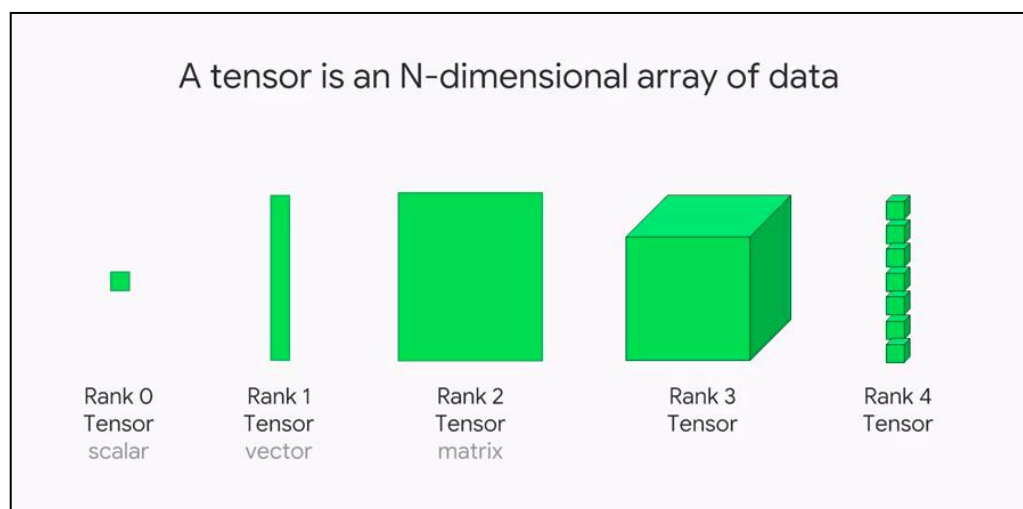


Figure 3.11: Various Dimensions of TensorFlow

### **3.4.4.6.6.3.Convolutional Neural networks**

Convolutional Neural networks are designed to process data through multiple layers of arrays. This type of neural networks is used in applications like image recognition or face recognition. The primary difference between CNN and any other ordinary neural network is that CNN takes input as a two-dimensional array and operates directly on the images rather than focusing on feature extraction, which other neural networks focus on.

The dominant approach of CNN includes solutions for problems of recognition. Top companies like Google and Facebook have invested in research and development towards recognition projects to get activities done with greater speed.[17]

### **3.4.4.6.7. Face Recognition with Javascript**

#### **3.4.4.6.7.1.Face-api.js**

To make a facial recognition app, we can use a library like face-api.js. Face-api.js is a JavaScript module, built on top of tensorflow.js core, which implements several CNNs to solve face detection, face recognition and face landmark detection, optimized for the web and for mobile devices. [23]

#### **3.4.4.6.7.2.Solving Face Recognition with Deep Learning**

To keep it simple, what we actually want to achieve, is to identify a person given an image of his face, e.g. the input image. The way we do that, is to provide one (or more) image(s) for each person we want to recognize, labeled with the person's name, e.g. the reference data. Now we compare the input image to the reference data and find the most similar reference image. If both images are similar enough we output the person's name, otherwise we output 'unknown'. [23]

However, two problems remain. Firstly, what if we have an image showing multiple persons and we want to recognize all of them?

Secondly, we need to be able to obtain such kind of a similarity metric for two face images in order to compare them.

#### **3.4.4.6.7.3.Face Detection**

The most accurate face detector is SSD (Single Shot Detector) , which is a CNN based on MobileNet V1, with some additional box prediction layers stacked on top of the network.

Furthermore, face-api.js implements an optimized Tiny Face Detector, basically an even tinier version of Tiny Yolo v2 utilizing depthwise separable convolutions instead of regular convolutions, which is a much faster, but slightly less accurate face detector compared to SSD MobileNet V1.

Lastly, there is also a MTCNN implementation, which is mostly around nowadays for experimental purposes however.

The networks return the bounding boxes of each face, with their corresponding *scores*, e.g. the probability of each bounding box showing a face. The scores are used to filter the bounding boxes, as it might be that an image does not contain any face at all. Note, that face detection should also be performed even if there is only one person in order to retrieve the bounding box.

#### **3.4.4.6.7.4.Face Landmark Detection and Face Alignment**

First problem solved! However, we want to point out that we want to align the bounding boxes, such that we can extract the images centered at the face for each box before passing them to the face recognition network, as this will make face recognition much more accurate.

For that purpose face-api.js implements a simple CNN, which returns the 68 point face landmarks of the image. From the landmark positions, the bounding box can be centered on the face.[23]

#### **3.4.4.6.7.5.Face Recognition**

Now we can feed the extracted and aligned face images into the face recognition network, which is based on a ResNet-34 like architecture and basically corresponds to the architecture implemented in dlib The network has been trained to learn to map the characteristics of a human face to a face descriptor (a feature vector with 128 values), which is also oftentimes referred to as face embedding.[23]

Now to come back to our original problem of comparing two faces: We will use the face descriptor of each extracted face image and compare them with the face descriptors of the reference data. More precisely, we can compute the Euclidian distance between two face descriptors and judge whether two faces are similar based on a threshold value (for 150 x 150 sized face images 0.6 is a good threshold value). Using Euclidian distance works surprisingly well, but of course you can use any kind of classifier of your choice.[23]

### 3.4.4.7. Search Mechanism

#### 3.4.4.7.1. Manual Search

FindMe has the flexibility that allows the users to use both manual and automatic search.

The manual search principle is simple, after signing in the users are routed to 'home-page' where the authenticated user sees other users public 'tracks', he can also filter them as he wishes and contact the 'track' creator for more details about the missing person.

Figure12 illustrates the manual search second level DFD.

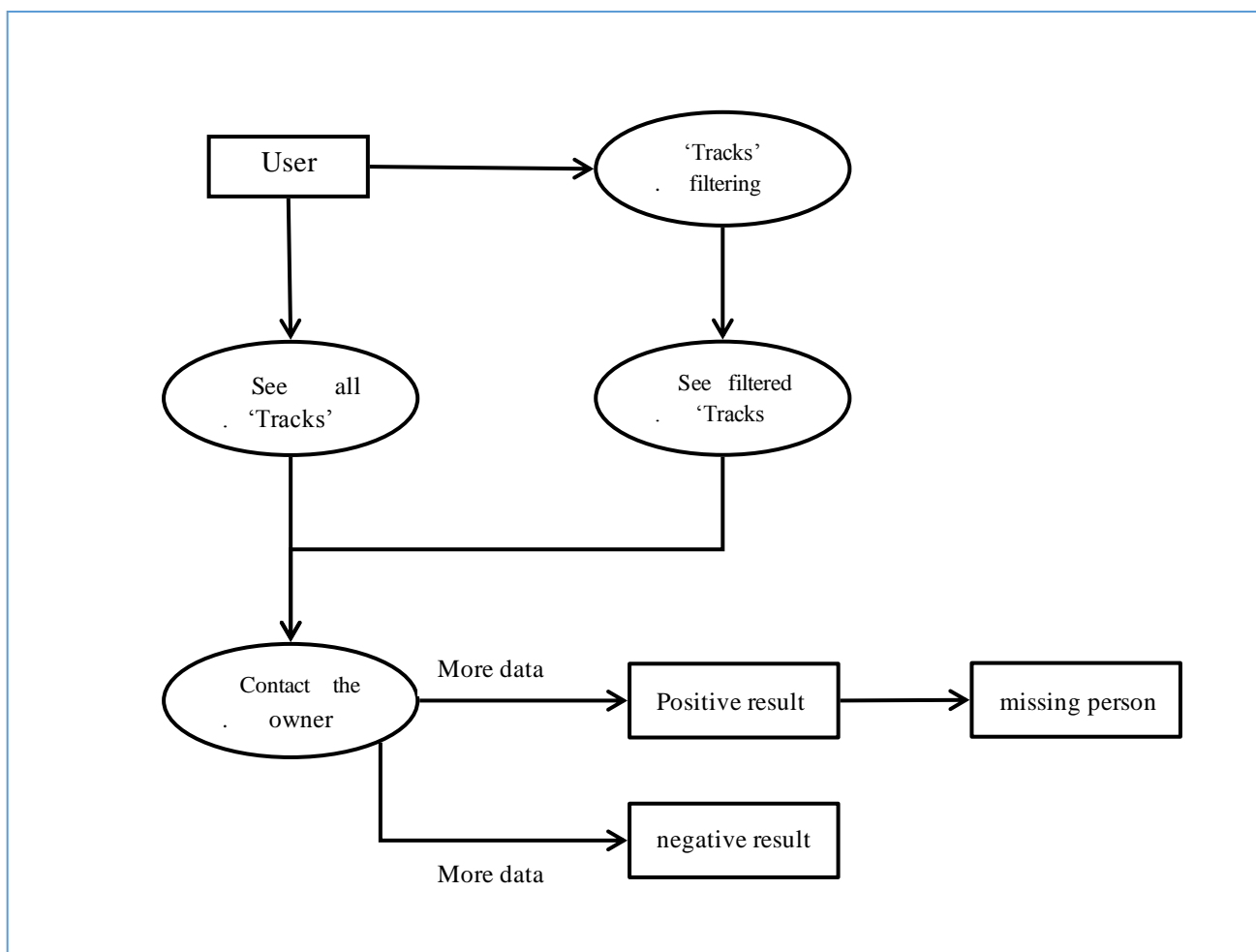


Figure 3.12: Manual Search Second Level DFD

#### 3.4.4.7.2. Automatic Search Mechanism

Both users and visitors (not authenticated users) can use the search feature. First, the user takes a picture using FindMe mobile application or sends a picture using the web application; this picture will be stored temporary on our server.



FindMe search system extracts the faces from this picture and compares it with all the missing people faces that are already stored when the users create their ‘tracks’. If any matches are detected, the server will send the phone numbers, the emails, and the full name to picture sender (positive result). This picture will be deleted when the process ends.

Face-api.js is responsible for detecting the picture faces and all the heavy things. The face recognition process may take some times so we need a high performance server. We advise users to store and send pictures containing one face to accelerate this process.

Figure 13 illustrates the automatic search mechanism second level DFD.

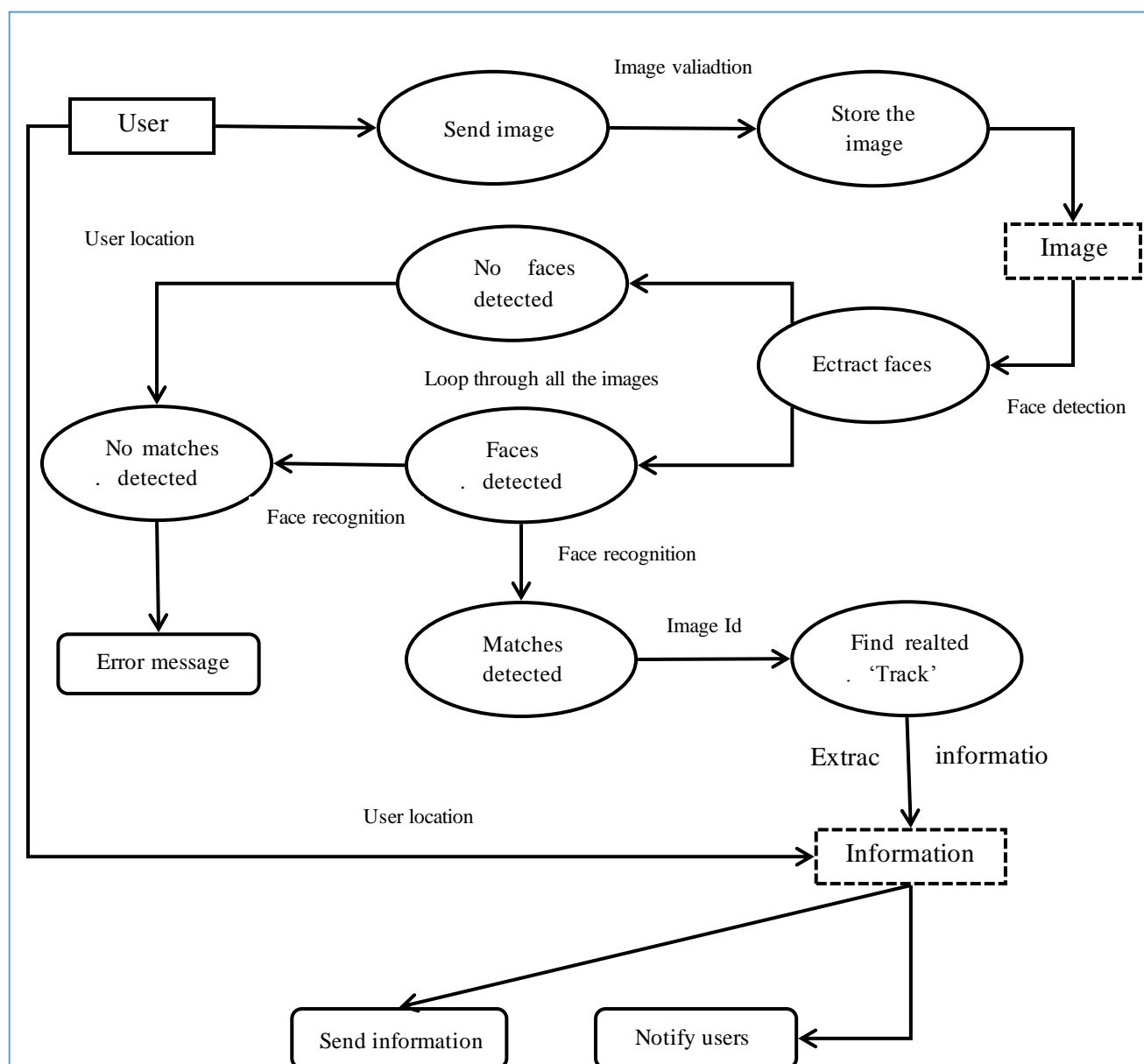


Figure 3.13: Automatic Search Mechanism Second Level DFD

### 3.4.5. DataBase Architecture

#### 3.4.5.1. Mongoose

Mongoose is a JavaScript framework that is commonly used in a Node.js application with a MongoDB database. It is an Object Document Mapper. This means that Mongoose allows you to define objects with a strongly-typed schema that is mapped to a MongoDB document.

Mongoose provides an incredible amount of functionality around creating and working with schemas. It enabled our project to connect MongoDB with Node.js easily.

#### 3.4.5.2. Models And Database Schema

Figure 14 illustrates these Mongoose schemas associations (relations).

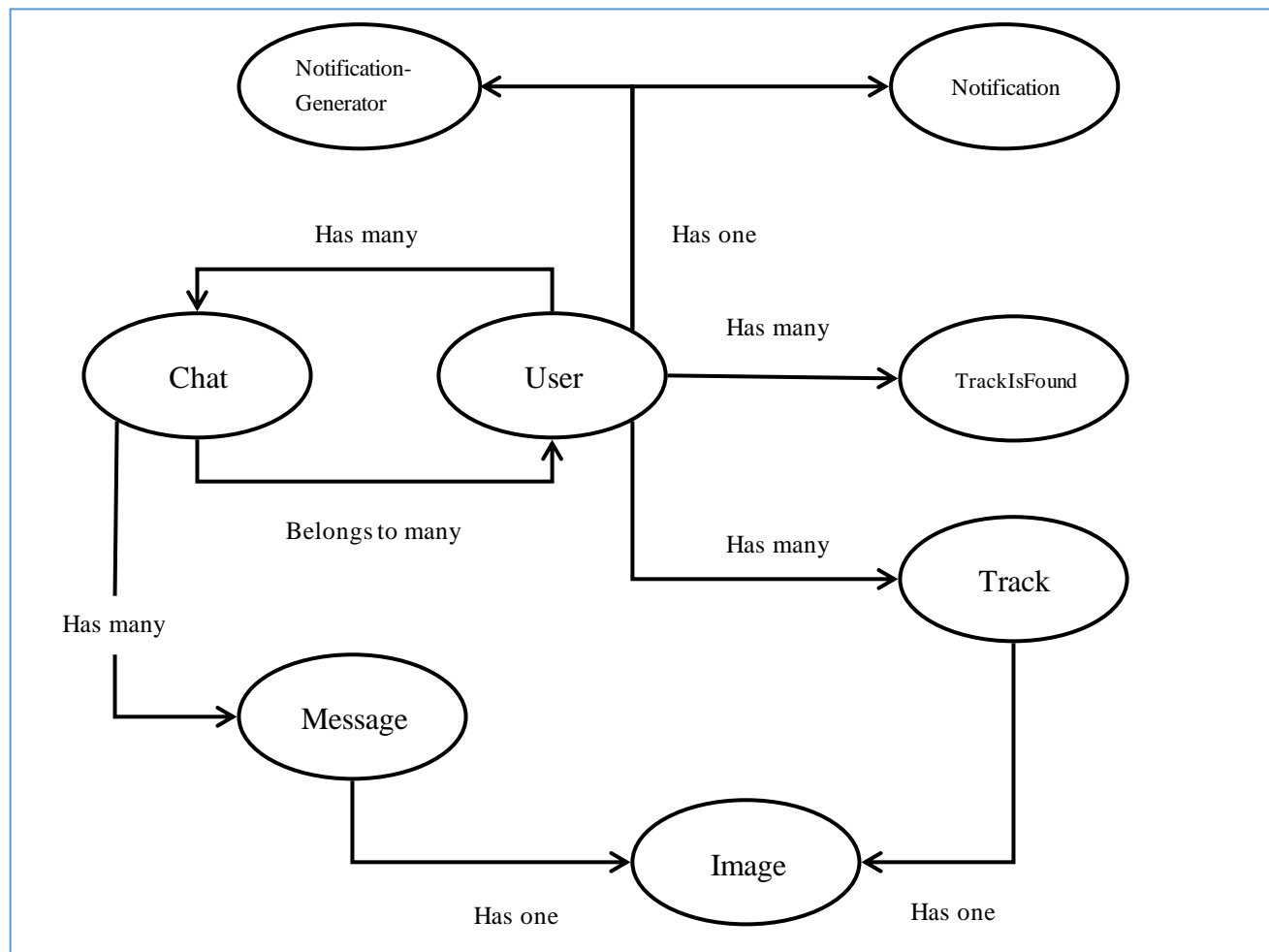


Figure 3.14: Data Base Schema Associations Diagram

These files represent the data structure of the application. It implements data logic and handles the storage to MongoDB. Everything in Mongoose starts with a Schema. Each

schema maps to a MongoDB collection and defines the shape of the documents within that collection

The user belongs to many conversations, each conversation has many messages and belongs to one 'track', The user has many 'tracks' and temporary 'tracks' (TheyFoundMe) that's related to the found person 'track' after the positive result of the search.

### 3.4.5.3. Data Model Schema

#### 3.4.5.3.1. User

User model schema represents the account data structure.

*Table 1: User Data Model Schema*

Key Name	Data Type
_id	ObjectId
email	String
userName	String
password	String
facebookId	String
resetExpiration	Date
resetToken	String
editExpiration	Date
editToken	String
createdAt	Date

#### 3.4.5.3.2. Track

Track model schema represents the 'tracks' data structure.

Table 2: Track Data Model Schema

Key Name	Data Type
_id	ObjectId
Name	String
Surname	String
dateOfBirth	Date
Gender	String
lostAt	Date
lostIn	String
isFound	Boolean
foundAt	Date
foundIn	Date
imageName	String
Emails	[String]
phoneNumbers	[String]
userId	ObjectId
Access	String
createdAt	Date

**3.4.5.3.3. TrackIsFound**

It represents the data structure of the 'tracks' that are found.

*Table 3: TrackIsFound Data Mode Schema*

Key Name	Data Type
_id	ObjectId
Name	String
Surname	String
imageName	String
Location	String
foundAt	Date
userId	ObjectId
trackId	ObjectId

**3.4.5.3.4. Temp User**

*Table 4: TempUser Data Model Schema*

Key Name	Data Type
_id	ObjectId
Email	String
Username	String
Password	String
Token	String
Expiration	Date
createdAt	Date

When the user creates the account for the first time, the tempUser will be created to store the user data.

After the email confirmation, this tempUser will be deleted.

### 3.4.5.3.5. Image

It represents the data structure of the images in our application.

*Table 5: Image Data Model Schema*

Key Name	Data Type
_id	ObjectId
imageName	String
Access	String
trackId	ObjectId
chatId	ObjectId
messageId	ObjectId
userId	ObjectId
createdAt	Date

### 3.4.5.3.6. Chat

It represents the data structure of the users conversations.

*Table 6: Chat Data Model Schema*

Key Name	Data Type
_id	ObjectId
Between	[ObjectId]
trackId	ObjectId
isBlocked	Boolean
lastMessageCreatedAt	Date
createdAt	Date

### 3.4.5.3.7. Message

It represents the data structure of the users messages.

Table 7: Message Data Model Schema

Key Name	Data Type
_id	ObjectId
Text	String
File	String
fileType	String
filename	String
userId	ObjectId
trackId	ObjectId
chatId	ObjectId
Seen	Boolean
Removed	Boolean
createdAt	Date

**3.4.5.3.8. Notification**

It represents the data structure of the users notifications.

Table 8: Notification Data Model Schema

Key Name	Data Type
_id	ObjectId
firstName	String
lastName	String
userId	ObjectId
trackIsFoundId	ObjectId
createdAt	Date

**3.4.5.3.9. NotificationGenerator**

It represents the data structure of the users notifications counter.

Table 9: NotificationGenerator Data Model Schema

Key Name	Data Type
_id	ObjectId
messagesCounter	Number
notificationsCounter	Number
userId	ObjectID
createdAt	Date

### 3.5. Conclusion

In this chapter we described the functionality of the FindMe application and our project requirements. We also described the project model using the data flow diagrams.

In the next chapter we will start with the implementation and we will present the result that we have obtained.



## Chapter 4 Implementation and results

### 4.1. Introduction

This chapter describes the implementation of the FindMe application, which is represented on setting up the project. In this chapter, we will present the important part of the code, and we will display the user interfaces.

This chapter also provides a simple discussion about our project limitations.

### 4.2. Setting Up The Project

#### 4.2.1. Setting Up The Back-end

#### 4.2.2. Initiating The Back-end Project

To initiate the back-end project let us create a new empty project folder using the terminal:

```
$ mkdir find_me
```

Then we change into that newly created folder by using:

```
$ cd find_me
```

Let's create a 'package.json' file inside that folder by using the following command:

```
$ npm init -y
```

All npm packages contain a file, usually in the project root; called 'package.json'. This file holds various metadata relevant to the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies.

It can also contain other metadata such as a project description, the version of the project in a particular distribution, license information, even configuration data - all of which can be vital to both npm and to the end users of the package. The 'package.json' file is normally located at the root directory of a Node.js project.

With the 'package.json' file available in the project folder we're ready to add some dependencies to the project.

Figure 01 shows our back-end project 'package.json' file.

```

{
  "name": "find_me",
  "version": "1.0.0",
  "description": "last year project",
  "main": "app.js",
  "scripts": {
    "start": "node app.js",
    "server": "nodemon app.js"
  },
  "author": "houssam eddine",
  "license": "MIT",
  "dependencies": {
    "@tensorflow/tfjs-node": "^1.7.0",
    "bcryptjs": "^2.4.3",
    "canvas": "^2.6.1",
    "config": "^3.3.0",
    "express": "^4.17.1",
    "express-validator": "^6.4.0",
    "face-api.js": "^0.22.1",
    "jsonwebtoken": "^8.5.1",
    "mongoose": "^5.9.2",
    "multer": "^1.4.2",
    "nodemailer": "^6.4.4",
    "nodemailer-sendgrid-transport": "^0.2.0",
    "passport": "^0.4.1",
    "passport-facebook-token": "^3.3.0",
    "socket.io": "^2.3.0"
  },
  "devDependencies": {
    "concurrently": "^5.1.0",
    "nodemon": "^2.0.2"
  }
}

```

Figure 4.1: Back-End package.json File

To install the dependencies we just setup, we just go into our console and type 'npm install', we will see our application working to bring in those modules into the 'node\_modules' directory that it creates.

#### 4.2.2.1. Setting Up The app.js File

This is the back-end project entry point. We have now pulled in our modules, configured our application for things like database, some express settings, routes, and then started our server. The entire code for the file is here and it is commented for help understanding.

Figure 02 shows the 'app.js' file.

```

// declare variables
const PORT = process.env.PORT || 5000;

app.use((err, req, res, next) => {
  const status = err.status || 500;
  //if (status === 500) err.message = 'server problem';
  console.log(err);
  const { message, data } = err;
  res.status(status).json({
    message,
    data,
  });
});
//connect to databse and start server
mongoose
  .connect('mongodb://localhost/findMe', {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true,
  })
  .then(() => {
    console.log('dbconnected');
    const httpServer = app.listen(PORT);
    socketIo.init(httpServer);
    socketIo.getIo().on('connection', socketServer);
  })
  .catch((err) => {
    console.log(err);
  });

```

Figure 4.2: Back-End Project app.js File

#### 4.2.2.2. Routes

##### 4.2.2.2.1. Auth.js (Route)

This file contains the routes, which are responsible for the account manipulation and the user authentication endpoints. Therefore, this is the link between the account management back-end logic and the front-end.

Figure 03 illustrates the register route source code.

```

//@access public
router.put(
  '/register',
  [
    body('email', 'the email must be exist and correct')
      .isEmail()
      .custom(async (value, { req }) => {
        try {
          const user = await User.findOne({ email: value });
          if (user) throw new Error('this user is already exist');
          return true;
        } catch (error) {
          throw error;
        }
      })
      .normalizeEmail(),
    body('userName')
      .not()
      .isEmpty()
      .withMessage('the username must be exist')
      .isLength({ min: 3 })
      .withMessage('the username must be more than 3 characters')
      .custom(async (value, { req }) => {
        try {
          const user = await User.findOne({ userName: value });
          if (user) throw new Error('this username is already exist');
          return true;
        } catch (error) {
          throw error;
        }
      })
  ]
);

```

Figure 4.3: Register Route Source Code

#### 4.2.2.2.2. Track.js (Route)

This file contains the routes, which are responsible for the ‘track’ manipulation endpoints. Therefore, this is the link between the ‘track’ management back-end logic and the front-end.

Figure 04 illustrates the ‘create track’ route source code.

```

router.post(
  '/create-track',
  isAuth,
  multerConfig('trackImages'),
  [
    body('name', 'first name is required').not().isEmpty(),
    body('surname', 'last name is required').not().isEmpty(),
    body('dateOfBirth', 'you must enter a correct date of birth').custom(
      (value, { req }) => {
        if (value) {
          if (new Date(value) == 'Invalid Date') {
            return false;
          }
          return true;
        }
        return true;
      }
    ),
    body('gender', 'the gender must be male or female').custom(
      (value, { req }) => {
        if (value !== 'male' && value !== 'female') return false;
        return true;
      }
    ),
    body('lostAt', 'the date of loss must be correct').custom(
      (value, { req }) => {
        if (value) {
          if (new Date(value) == 'Invalid Date') {
            return false;
          }
          return true;
        }
      }
    )
  ]
);

```

Figure 4.4: ‘Create-Track’ Route Source Code

#### 4.2.2.2.3. Chat.js (Route)

This file contains the routes, which are responsible for the chat endpoints. So this is the link between the chat back-end logic and the front-end.

Figure 05 illustrates all the chat routes source code.

```
const router = express.Router();

router.post(
  '/message',
  isAuth,
  multerConfig('messageImages'),
  conversationController.sendMessage
);

router.patch('/remove-message', isAuth, conversationController.removeMessage);

router.patch(
  '/block-conversation',
  isAuth,
  body('chatId', 'chatId is required').not().isEmpty(),
  conversationController.blockConversation
);

router.get(
  '/chat-messages/:chatId',
  isAuth,
  conversationController.getChatMessages
);

router.get('/last-messages', isAuth, conversationController.getLastMessages);

module.exports = router;
```

Figure 4.5: Chat Routes Source Code

### 4.2.2.3. Controllers

#### 4.2.2.3.1. Auth.js (Controller)

This file contains the middlewares, which are responsible for the account manipulation and the user authentication logic.

Figure 06 illustrates the register middleware source code.

```

//register
exports.register = async (req, res, next) => {
  try {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      const err = new Error('validation failed');
      err.status = 422;
      err.data = errors.array();
      throw err;
    }
    const { password, email, userName } = req.body;
    const hashedPassword = await bcrypt.hash(password, 12);
    const token = await crypto.randomBytes(6).toString('hex');
    let tempUser = new TempUser({
      email,
      userName,
      password: hashedPassword,
      expiration: Date.now() + 3600000,
      token,
    });
    tempUser = await tempUser.save();
    await sendMail(
      email,
      'account register',
      `

${userName} this is your confirmation code: <b>${token}</b></p>`
    );
    res.status(201).json({
      message: 'tempUser was created',
      tempUserId: tempUser._id,
    });
  } catch (err) {
    next(err);
  }
}


```

Figure 4.6: Register Middleware Source Code

#### 4.2.2.3.2. Track.js (Controller)

This file contains the middlewares, which are responsible for the ‘track’ manipulation and the track management logic.

Figure 07 illustrates the ‘create-track’ middleware source code.

```
exports.createTrack = async (req, res, next) => {
  try {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      const err = new Error('validation failed');
      err.status = 422;
      err.data = errors.array();
      throw err;
    }
    if (!req.file) {
      const err = new Error('you must send an image');
      err.status = 422;
      throw err;
    }
    let { phoneNumbers, emails, socketId } = req.body;
    phoneNumbers = phoneNumbers ? phoneNumbers.split('-') : phoneNumbers;
    emails = emails ? emails.split('-') : emails;
    let track = new Track({
      ...req.body,
      imageName: req.file.filename,
      userId: req.userId,
      phoneNumbers,
      emails,
    });

    await image.save();
    res.json({
      message: 'track was created',
      track,
    });
  }
};
```

Figure 4.7: 'Create-Track' Middleware Source Code

#### 4.2.2.3.3. Conversation.js (Controller)

This file contains the middlewares which are responsible for the chat and the chat management logic.

Figure 08 illustrates the 'send-message' middleware source code.

```

//send message
exports.sendMessage = async (req, res, next) => {
  try {
    const { text, trackId, userId, socketId } = req.body;
    const file = req.file;
    let fileType = '';
    let fileName = '';
    const sender = await User.findById(req.userId);
    if (!(text || file)) {
      const err = new Error('you can not send an empty message!');
      err.status = 422;
      throw err;
    }
    if (text && file) {
      const err = new Error(
        'you can not send a text and a file at the same time!'
      );
      err.status = 422;
      throw err;
    }
    if (req.userId === sender.userId) {
      const err = new Error('the sender and the receiver must be different!');
      err.status = 422;
      throw err;
    }

    const track = await Track.findById(trackId);
    if (!track) {
      const err = new Error('track not found!');
      err.status = 404;
      throw err;
    }
  }
}

```

Figure 4.8: 'Send-Message' Middleware Source Code

### 4.2.3. Setting Up The React Application

#### 4.2.3.1. Initiating The Front-end Project

We create the initial React project by using the 'create-react-app' script. What's great about 'create-react-app' is that this script can be executed by using the npx command without the need to install it first on our system. Just execute the following command :

```
$ npx create-react-app client
```

Executing this command creates a new project directory 'client'. Inside this folder we will find the default React project template with all dependencies installed.

#### 4.2.3.2. Setting Up React Router

The next thing we need to be added to the project is the React Router package (react-router-dom) :



```
$ npm install react-router-dom
```

With this package installed we're ready to add the routing configuration in `app.js`.

#### 4.2.3.3. Setting Up The `index.js` File

This file is the entry point for our app which contains a `root` `div` element, then we add the `render` method, the only required method in a class component, which is used to render DOM nodes.

Inside the `render` method, we're going to put what looks like a simple HTML element. This is called `JSX`, and we'll talk about it in the next section.

Figure 09 illustrates the `index.js` file source code.

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router } from 'react-router-dom';

import './util/socketConnection'; //socket.io initialization
import { AuthProvider } from './hooks/context/authContext';
import { TrackProvider } from './hooks/context/trackContext';
import { ChatProvider } from './hooks/context/chatContext';
import { NotificationProvider } from './hooks/context/notificationContext';

import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(
  <Router>
    <AuthProvider>
      <TrackProvider>
        <ChatProvider>
          <NotificationProvider>
            <App />
          </NotificationProvider>
        </ChatProvider>
      </TrackProvider>
    </AuthProvider>
  </Router>,
  document.getElementById('root')
);
```

Figure 4.9: `index.js` Source Code

#### 4.2.3.4. JSX

As you've seen, we've been using what looks like HTML in our `index.js` code, but it's not quite HTML. This is `JSX`, which stands for JavaScript XML.

With `JSX`, we can write what looks like HTML, and also we can create and use our own XML-like tags.

Using JSX is not mandatory for writing React. Under the hood, it's running 'createElement', which takes the tag, object containing the properties, and children of the component and renders the same information.

JavaScript expressions can also be embedded inside JSX using curly braces, including variables, functions, and properties.

JSX is easier to write and understand than creating and appending many elements in vanilla JavaScript, and is one of the reasons people love React so much.

#### 4.2.3.5. Setting Up The App.js File

We've created one component (the App component). Almost everything in React consists of components, which can be class components or simple components. Most React apps have many small components, and everything loads into the main App component.

Figure 10 illustrates the app.js file source code.

```
const { getUser, isAuth, loading } = useContext(authcontext);
const { showSearch } = useContext(trackContext);
const getUserRef = useRef(getUser);
useEffect(() => {
  getUserRef.current();
}, []);
return (
  <div className={showSearch ? 'app show' : 'app'}>
    {isAuth ? <Route component={Navbar} /> : null}
    {loading ? (
      <Spinner />
    ) : (
      <div>
        {showSearch ? <Route path="/" component={Search} /> : null}
        <Switch>
          <Route exact path="/" component={isAuth ? Home : Landing} />
          <Route exact path="/register" component={Register} />
          <Route exact path="/login" component={Login} />
          <Route exact path="/reset-password" component={ForgotPassword} />
          <PrivateRoute
            exact
            path="/found-track/:trackId"
            component={FoundTrack}
          />
          <PrivateRoute exact path="/my-tracks/:state" component={MyTracks} />
          <PrivateRoute exact path="/my-tracks" component={MyTracks} />
          <PrivateRoute
            exact
            path="/user-settings/:state"
            component={UserSettings}
          />
        </Switch>
      </div>
    )}
  </div>
);
```

Figure 4.10: Front-End Project app.js File

### 4.2.3.6. State Management

#### 4.2.3.6.1. AuthContext.js

This file contains all the functions and the logic that is responsible for front-end authentication manipulation.

Figure 11 illustrates the register method source code.

```

//register
const register = async (formData) => {
  try {
    const response = await axios.put('/auth/register', formData);
    dispatch({
      type: TEMP_USER_IS_CREATED,
      value: { tempUserId: response.data.tempUserId },
    });
    Swal.fire({
      icon: 'success',
      title: `your verification code was sent to your email!`,
      showConfirmButton: false,
      timer: 1500,
    });
  } catch (error) {
    dispatch({ type: CREAT_TEMP_USER_FAILED });
    if (error.response.status === 500) {
      return Swal.fire({
        icon: 'error',
        title: 'Server Error',
        text: 'please try again',
      });
    }
    Swal.fire({
      icon: 'error',
      title: 'Validation Faild',
      text: error.response.data.data[0].msg,
    });
  }
};

```

Figure 4.11: Rgister Method Source Code

#### 4.2.3.6.2. TrackContext.js

This file contains all the functions and the logic that is responsible for front-end track manipulation.

Figure 12 illustrates the ‘create-track’ method source code.

```
const createTrack = async (form, socketId) => {
  const formData = new FormData();
  formData.append('socketId', socketId);
  for (let key in form) {
    formData.append(key, form[key]);
  }

  try {
    const response = await axios.post('/tracking/create-track', formData, {
      headers: { 'Content-Type': 'multipart/form-data' },
    });
    Swal.fire({
      icon: 'success',
      title: 'this track was created successfully',
      showConfirmButton: false,
      timer: 1500,
    });
    const tracks = [...state.tracks];
    tracks.unshift(response.data.track);
    dispatch({
      type: CREATE_TRACK_SUCCEED,
      value: { tracks },
    });
    history.goBack();
  } catch (error) {
    if (error.response.status === 500) {
      return Swal.fire({
        icon: 'error',
        title: 'Server Error',
        text: 'please try again',
      });
    }
  }
}
```

Figure 4.12: 'Create-track' Method Source Code

#### 4.2.3.6.3. ChatContext.js

This file contains all the functions and the logic that is responsible for front-end chat management.

Figure 13 illustrates the 'load-messages' method source code.

```

const loadLastMessages = async () => {
  try {
    const lastItem =
      state.lastMessages.messages[state.lastMessages.messages.length - 1];
    const currentPage = lastItem ? lastItem.createdAt : '';
    const response = await axios.get(
      `/conversation/last-messages?page=${currentPage}&&perPage=10`
    );
    const messages = [
      ...state.lastMessages.messages,
      ...response.data.messages,
    ];
    const loading = response.data.totalItems > messages.length;
    dispatch({
      type: LOAD_LAST_MESSAGES_SUCCEED,
      value: {
        messages: messages,
        totalItems: response.data.totalItems,
        loading,
      },
    });
  } catch (error) {
    dispatch({ type: LOAD_LAST_MESSAGES_FAILED });
    history.push('/');
    if (error.response.status === 500) {
      return Swal.fire({
        icon: 'error',
        title: 'Server Error',
        text: 'please try again',
      });
    }
  }
}

```

Figure 4.13: 'Load-Messages' Method Source Code

## 4.3. User Interfaces

### 4.3.1. Public Pages

#### 4.3.1.1. The FindMe Landing Page

A landing page is the first page you land on after clicking the website link. In this sense, a landing page could be almost anything.

FindMe landing page contains login, signin buttons, and a button to start the automatic search.

Figure 14 and figure 15 show the landing page user interface displayed on all screens.

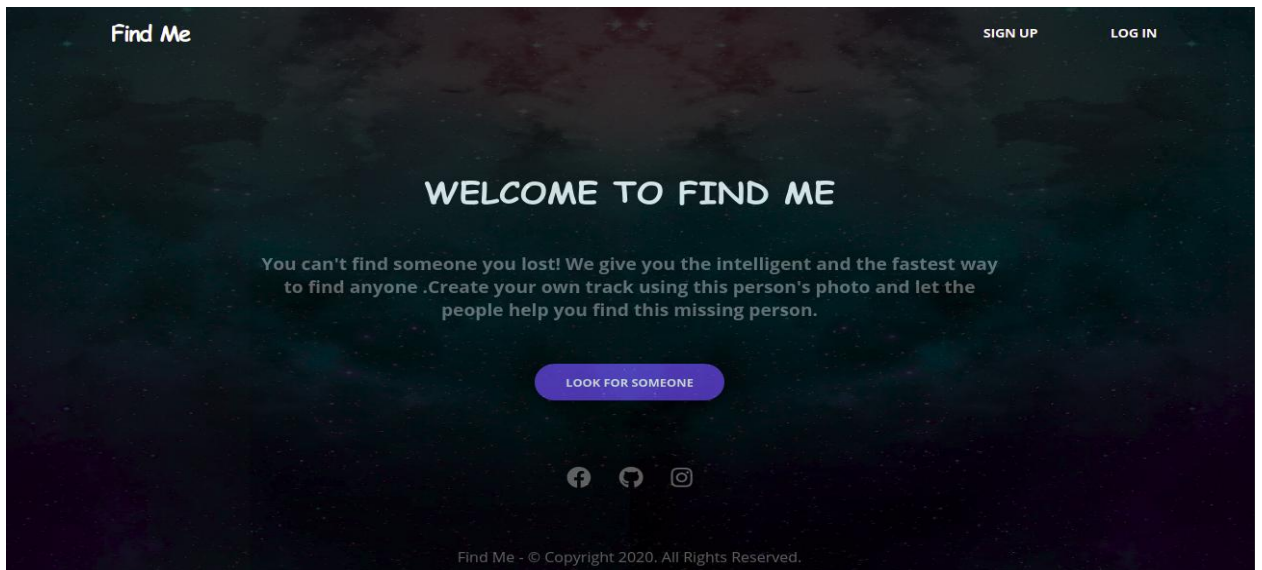


Figure 4.14: Large Screen Landing Page User Interface

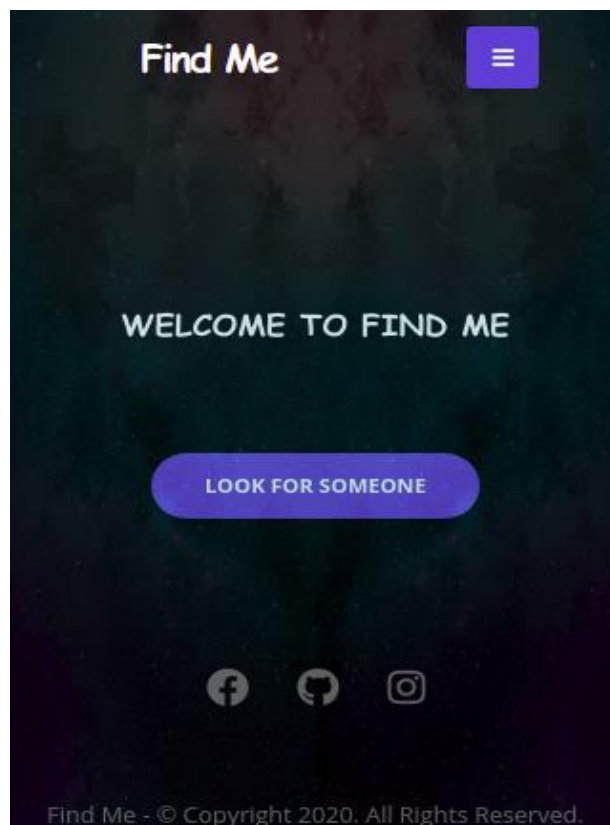
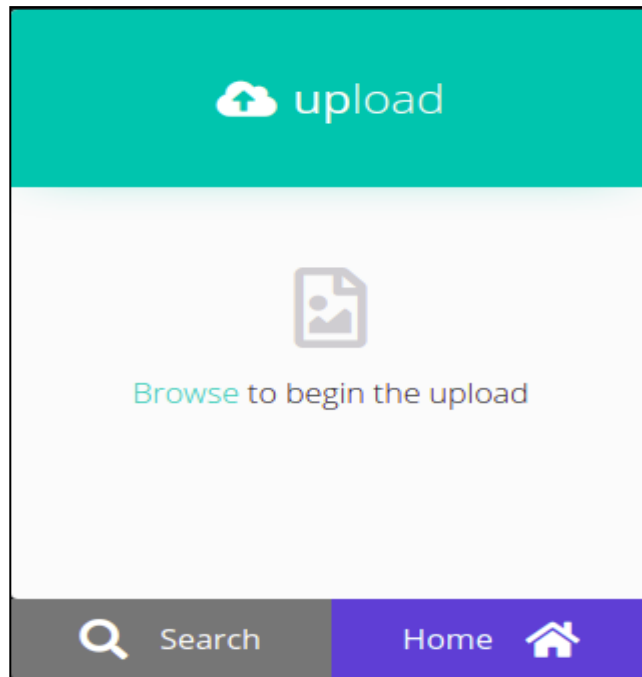


Figure 4.15: Small Screen Landing Page User Interface

#### 4.3.1.2. Automatic Search UI

Here the user can browse an image and search for anyone by clicking on the ‘look for someone’ button. The search result will appear here on the same page. This page will appear to all users (authenticated and not authenticated users).

Figure 16 bellow shows the automatic search page user interface (same page displayed on all screens).



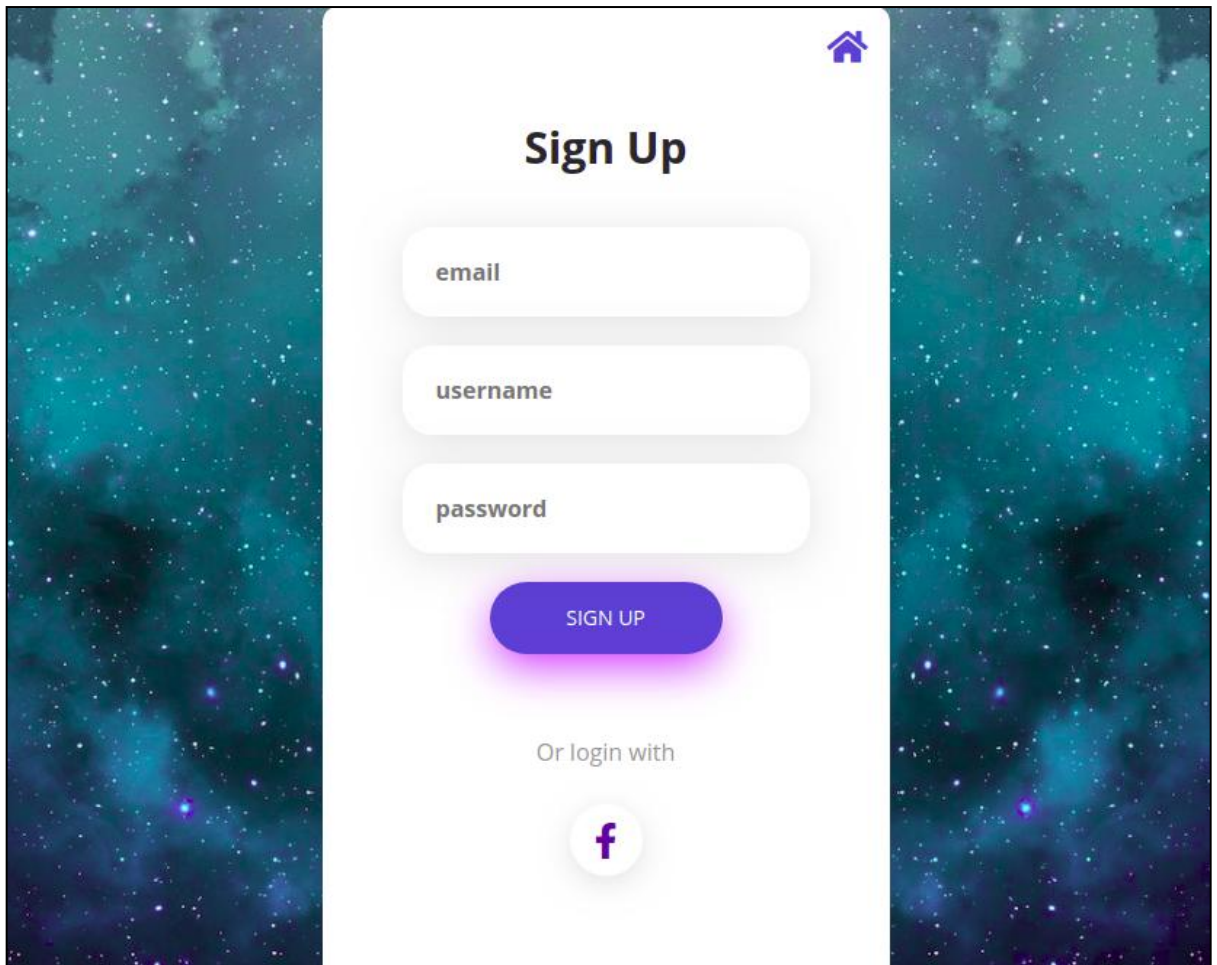
*Figure 4.16: Automatic Search User Interface*

### **4.3.1.3. Authentication Pages**

#### **4.3.1.3.1. Signup Ui (Registration)**

The Signup page (also known as a registration page) enables other users to independently register and gain access to FindMe system. The user needs to enter his email, username and password to create an account .In this page the user has an option to create an account using his Facebook profile (the username will be generated automatically).

Figure 17 shows the signup user interface (same page displayed on all screens)



*Figure 4.17: Signup User Interface*

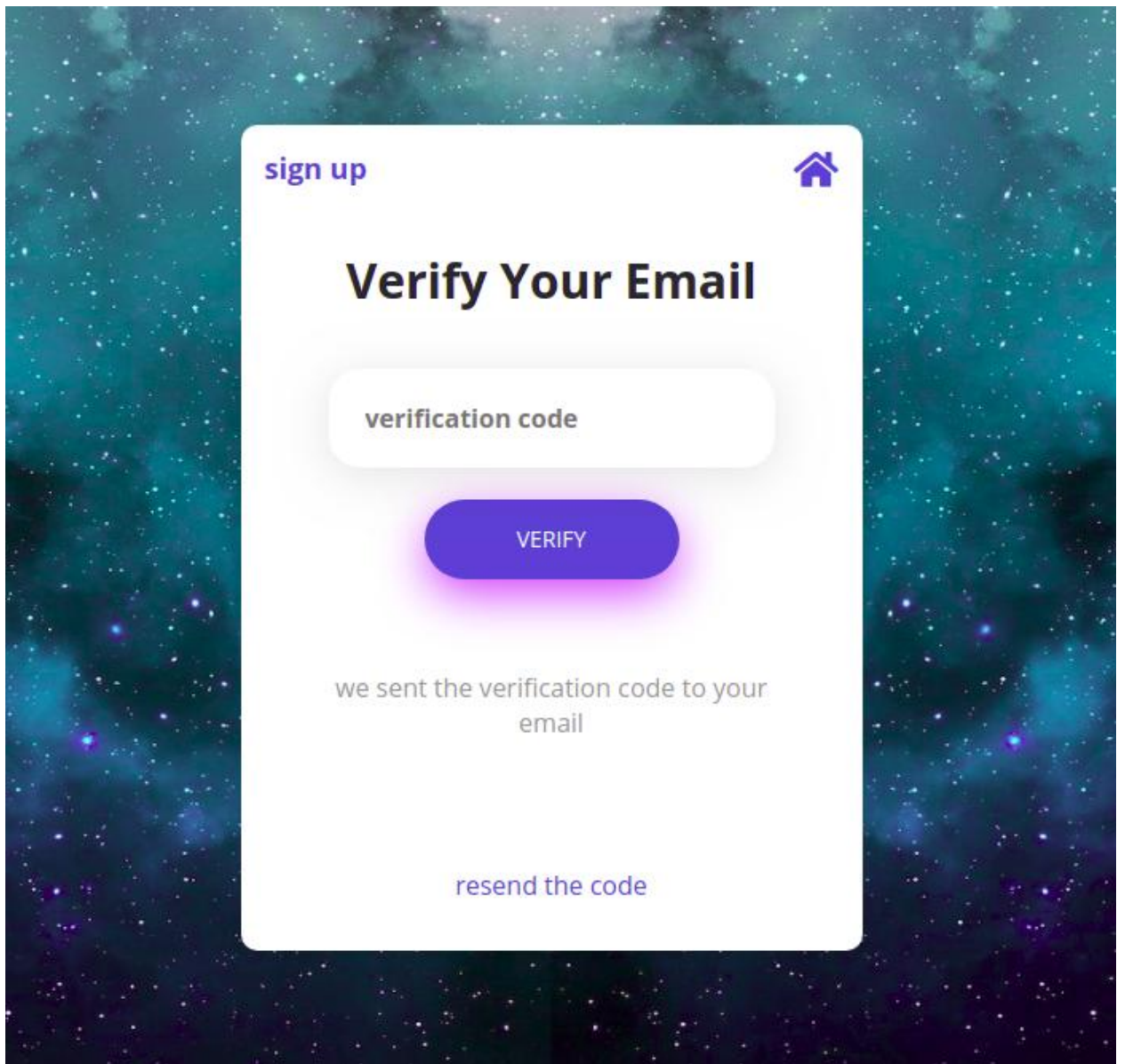
#### **4.3.1.3.2. Email Confirmation UI**

The confirmation email is a kind of a transactional email sent to a customer after a certain condition is triggered.

To complete the registration, the user must confirm his email .An automatic response email will be send to the user it contains a confirmation code.

Figure 18 shows the email confirmation user interface (same page dispayed on all screens).



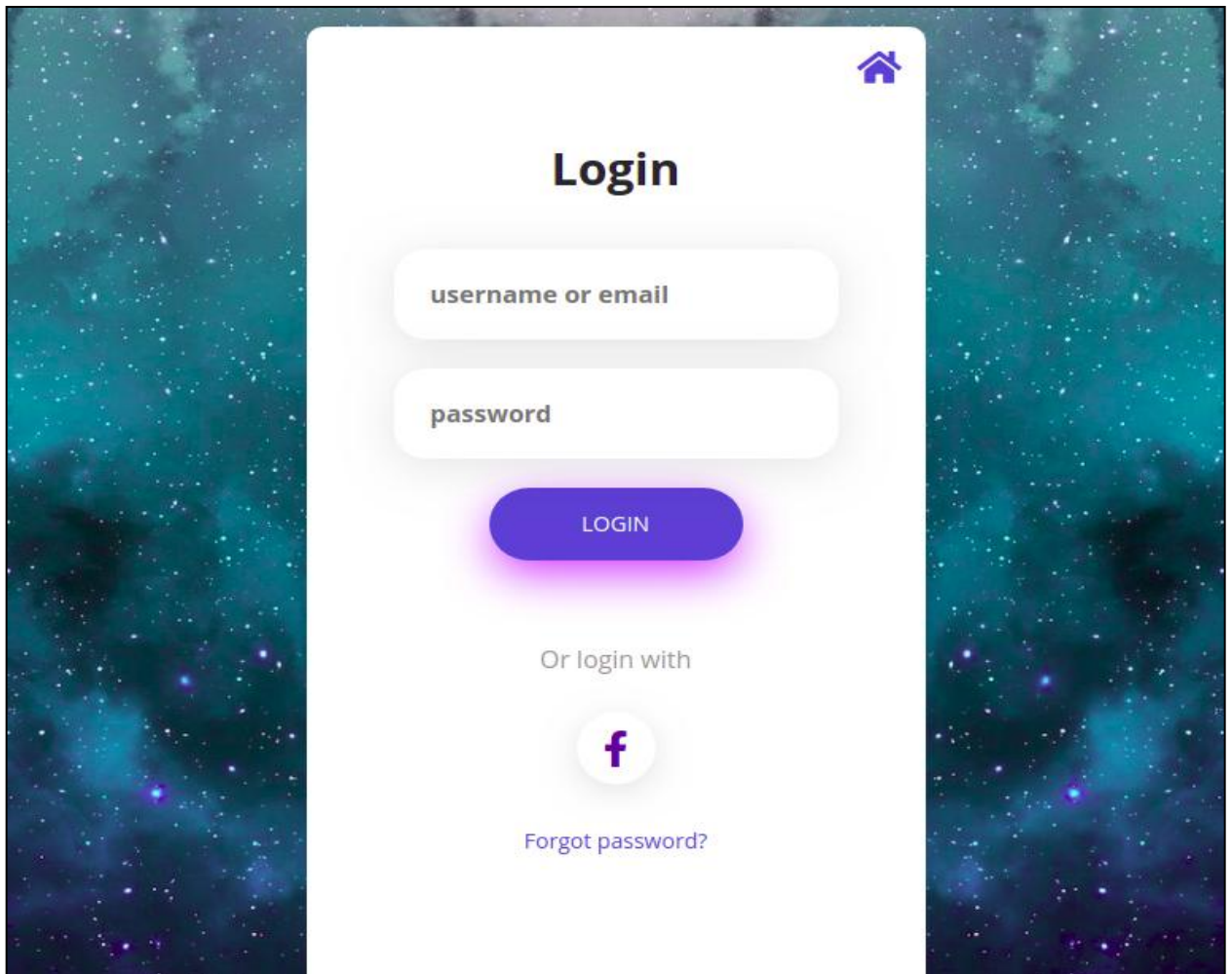


*Figure 4.18: Email Confirmation User Interface*

#### **4.3.1.3.3. Login UI**

The login page allows a user to gain access to an application by entering their username (or email) and password or by authenticating using Facebook.

Figure 19 shows the login user interface (same page displayed on all screens).



*Figure 4.19: Login User Interface*

The user navigates to the application and is presented with a login page as a way to gain access to the application. There are two possible results:

- Authentication is successful and the user is directed to the application home page.
- Authentication fails and the user remains on the login page. If authentication fails, the screen shows an informational or error message about the failure.

The user is automatically logged out and redirected to the login page, which will display an informational message explaining what happened.

#### **4.3.1.3.4. Password Recovery**

If the user has forgotten his password. A link is available in the login page to begin the process to reset this password. Once the user clicks on this link, an automatic response email will be sent to the user. It contains a confirmation code and the contents of the login page are replaced with fields specific to recovery the password.

Figure 20 and 21 illustrate the UI password recovery steps.

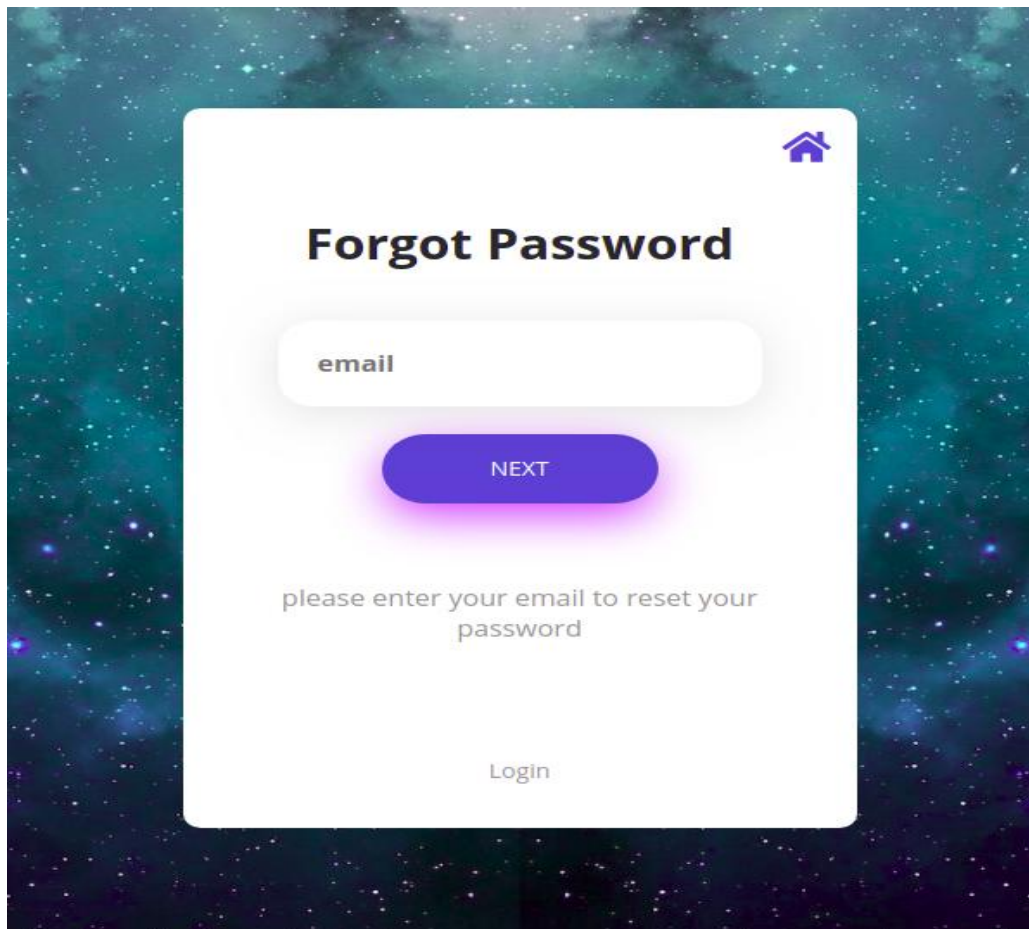
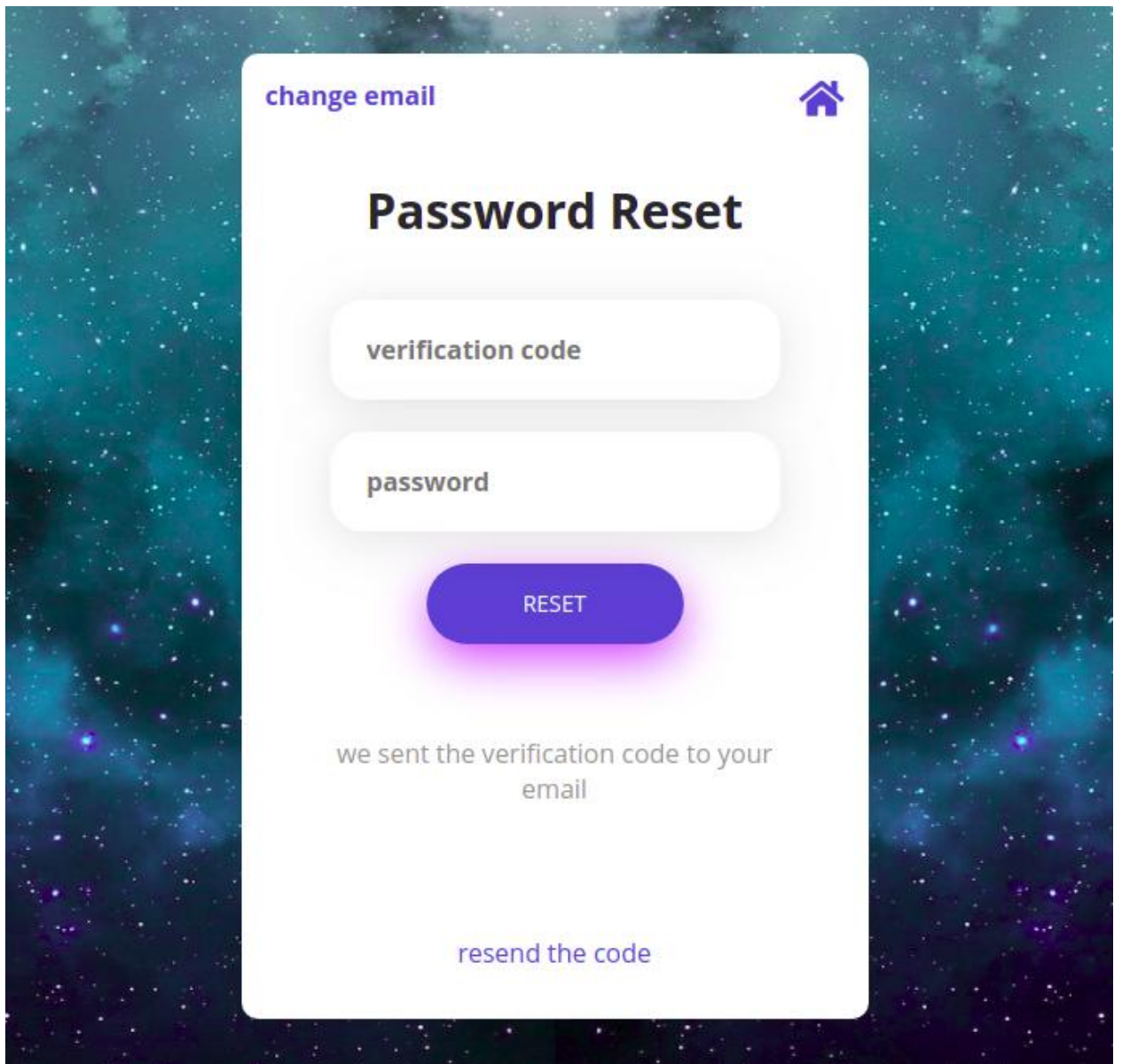


Figure 4.20: Forgot Password User Interface



*Figure 4.21: Reset Password User Interface*

#### **4.3.1.4. Error Page UI (Not Found Page)**

When a visitor clicks on a link to a deleted or moved page, he will see the 404 error page. He will also see it if he mistypes a URL, or clicks on a broken or truncated link

Figure 22 shows the error page user interface (same page displayed on all screens).





*Figure 4.22: Error Page User Interface*

## **4.3.2. Private Pages**

### **4.3.2.1. Home page**

#### **4.3.2.1.1. Components**

##### **4.3.2.1.1.1. Components Distribution**

The Home UI pops in after successful registration of user details (or login). The home page is the biggest part in FindMe Web client that uses React.js components principle, so we decided to divide the UI explanation into 3 main parts (components) :

Filter component.

Right sidebar component.

Track component.

##### **4.3.2.1.1.2. Track Component**

We can imagine the track as a post on the social media apps. The track contains the missing person data and the image that FindMe automatic search based in.

The user can see only the public tracks and his own tracks; the home page displays all the public tracks that the users have.

We have 4 types of tracks:

The other users track that is created by a different user, this track provides a communication feature with the owner represented by the ‘connect’ button. Figure 23 illustrates the users track UI.

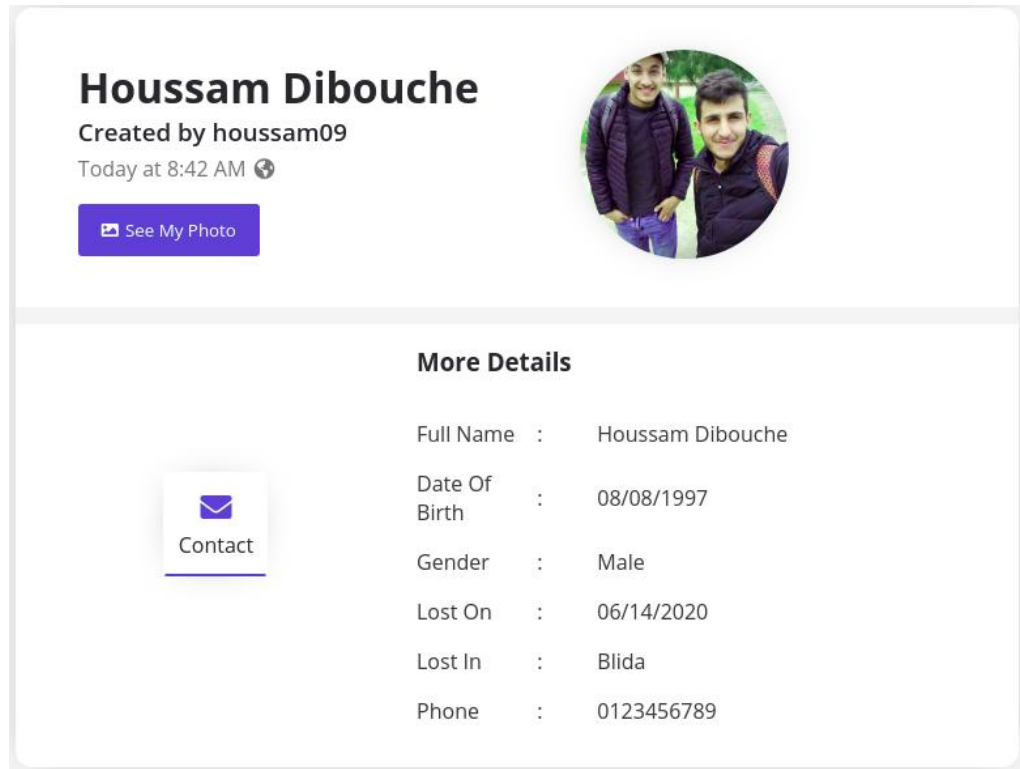


Figure 4.23: Other Users Track User Interface

The owner track that belongs to the same user, this track provides a delete and edit button where the creator has the freedom to do anything with this track.

Figure 24 illustrates the owner track UI.

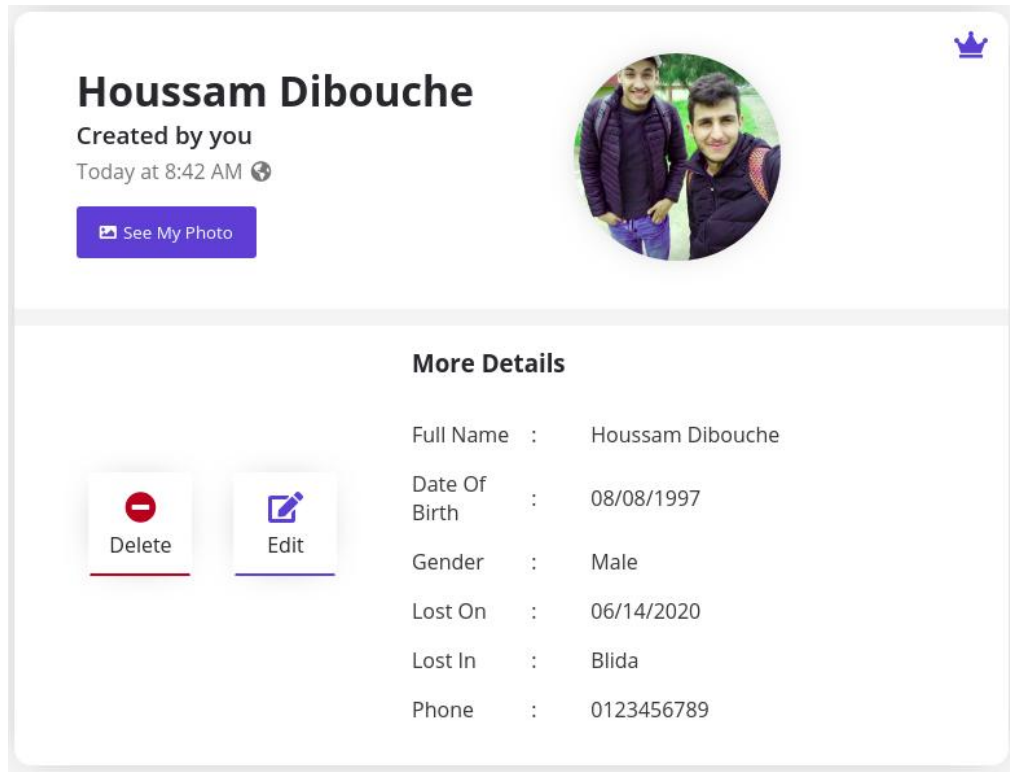


Figure 4.24: Owner Track User Interface

The temporary track, this track will be created when the user-missing person is found. The track provides a delete and confirm button. Figure 25 illustrates the temporary track UI.

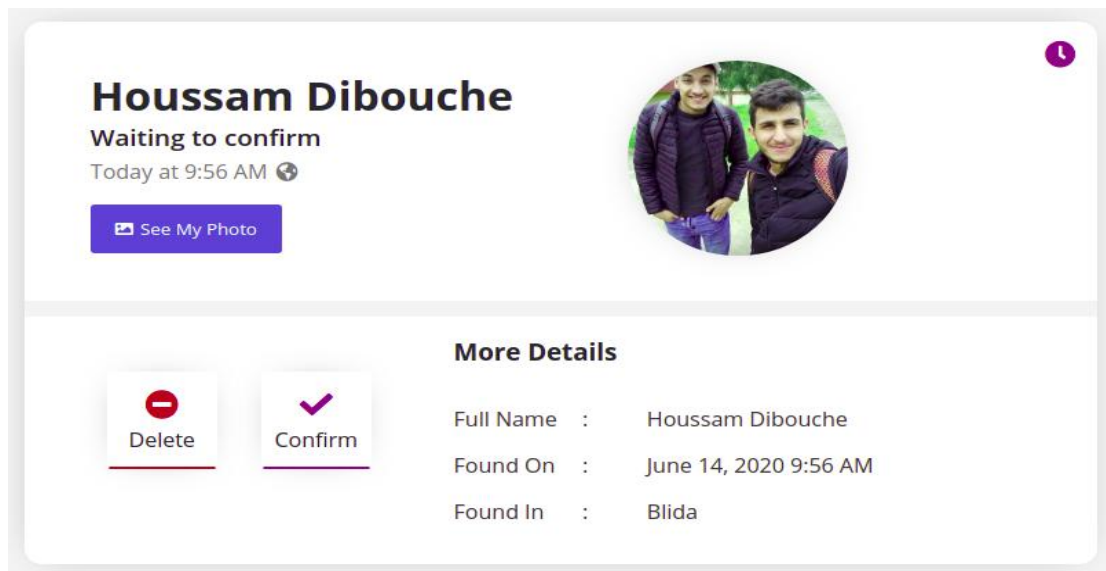


Figure 4.25: Temporary Track User Interface

The found person track, it's created when the user confirm the temporary track, this track provides only a delete option and contains this person data as will the information that are related to the person finding . Figure 26 illustrates the found person track user interface.

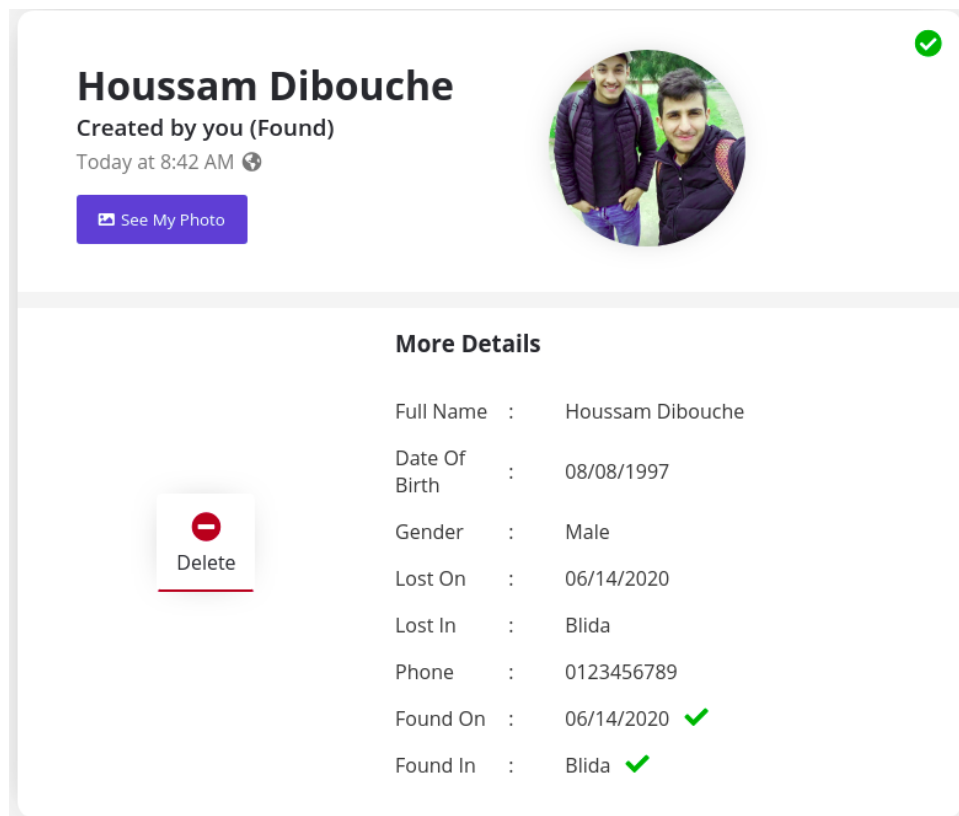


Figure 4.26: Found Person Track User Interface

#### 4.3.2.1.1.3.Filter Component

This component gives the user the ability to manage the home page tracks sorting. The user has the option of filtering the 'tracks' to see them based on the location and if the 'track' is found or not. Figure 27 illustrates the filter UI.

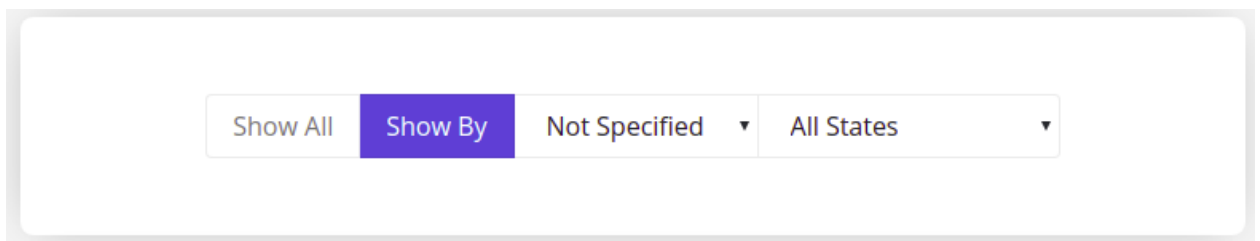
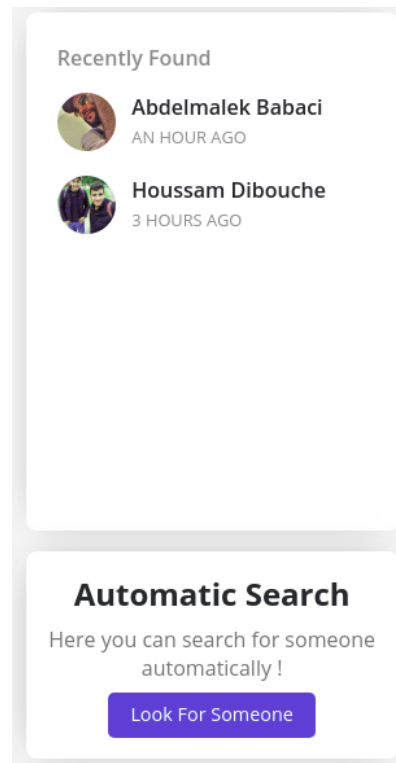


Figure 4.27: Filter User Interface

#### 4.3.2.1.2. Right Sidebar Component

This component provides a search button and displays the recent found people for the last 24 hours. Figure 28 illustrates the right sidebar component UI.





*Figure 4.28: Right SideBar Component User Interface*

#### **4.3.2.1.3. Home UI**

Now we present the full page with all the components together.

On Small screens the 'recent found' part (component) disappears and the automatic search part is displayed above the filter component.

Figure 29 and figure 30 show the home page user interface displayed on all screens.

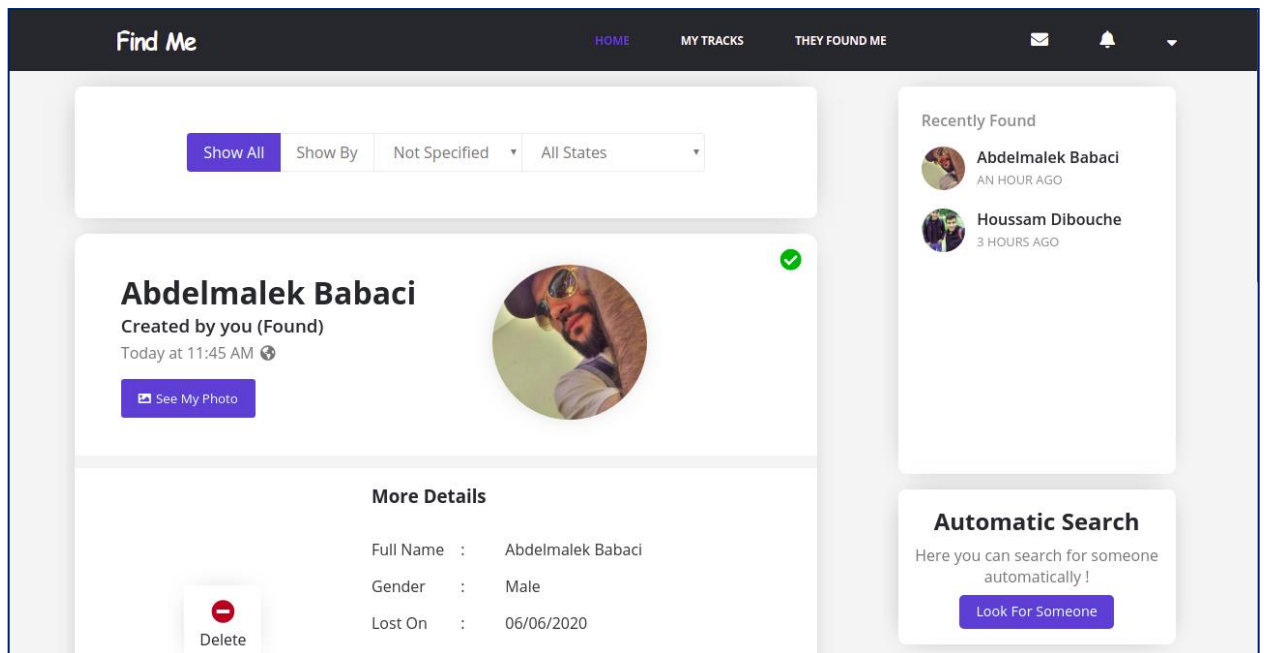


Figure 4.29: Home User Interface Displayed On Large Screen

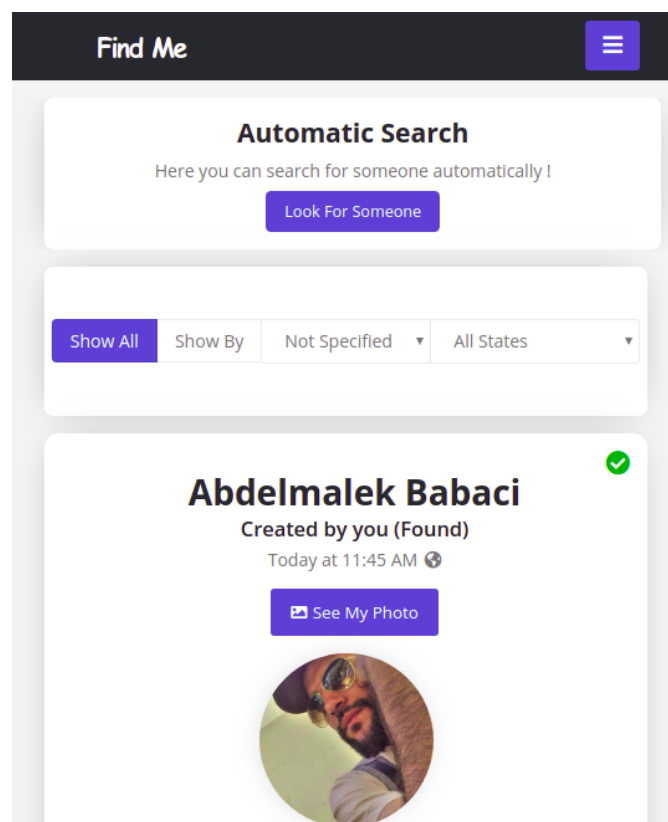


Figure 4.30: Home User Interface Displayed On Small Screen

#### 4.3.2.2. My Tracks UI

Here the user can create a new track. This page also allows users to reach their tracks, edit and delete them.

Figure 31 shows my tracks user interface (same page displayed on all screens).

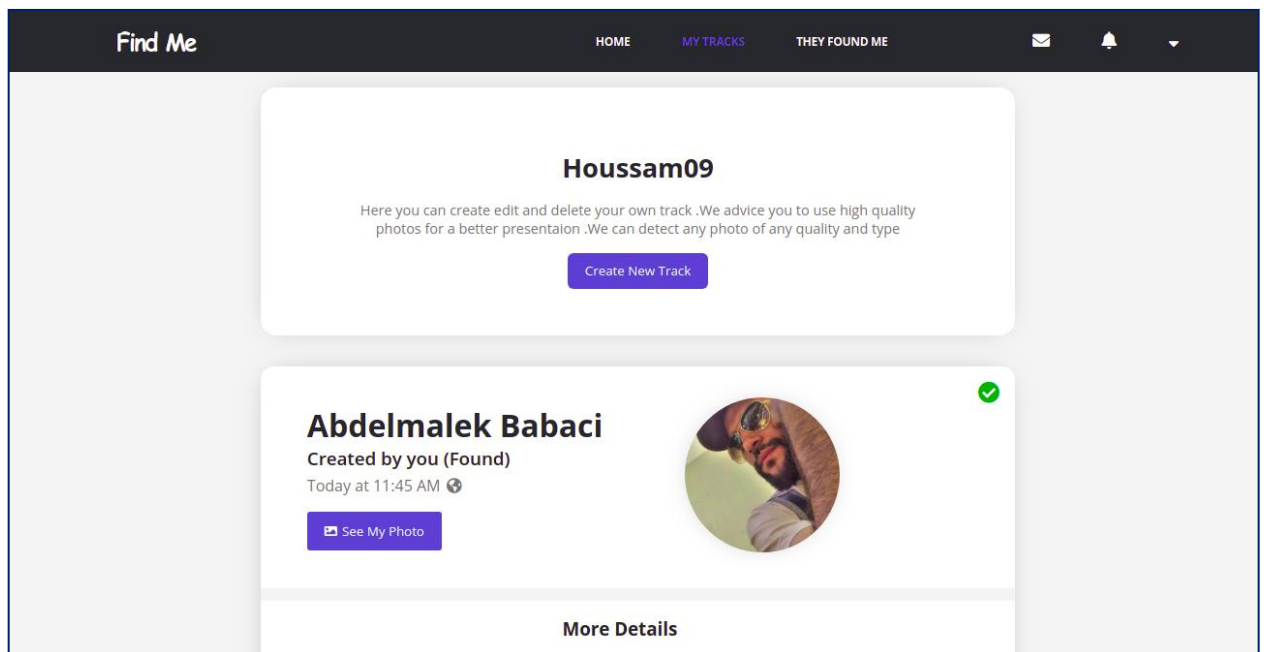


Figure 4.31: My Tracks User Interface

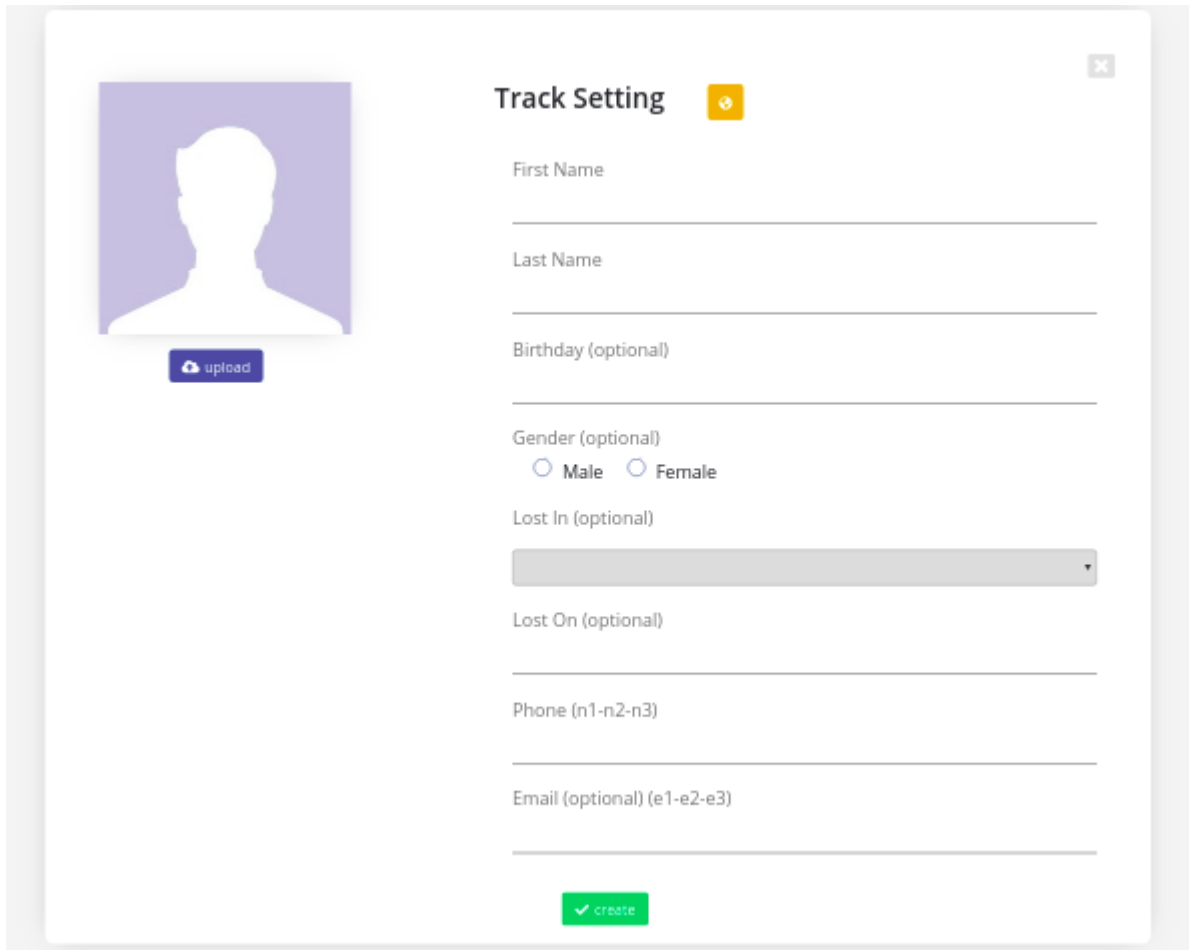
#### 4.3.2.3. Create Track UI

Create track and edit track have the same UI (it is the same component). Here the user enters the correct missing person information.

There is a required information that must be included like:

- First name.
- Last name.
- Phone number.
- The missing person picture.

Figure 32 shows create track user interface (same page displayed on all screens).



The screenshot displays a 'Track Setting' form. On the left, there is a purple square placeholder for a profile picture with a white silhouette of a person and a blue 'upload' button below it. To the right, the form is titled 'Track Setting' with a yellow close button (X) in the top right corner. The form contains the following fields: 'First Name' (text input), 'Last Name' (text input), 'Birthday (optional)' (text input), 'Gender (optional)' (radio buttons for 'Male' and 'Female'), 'Lost In (optional)' (dropdown menu), 'Lost On (optional)' (text input), 'Phone (n1-n2-n3)' (text input), and 'Email (optional) (e1-e2-e3)' (text input). A green 'create' button with a checkmark is positioned at the bottom center of the form.

Figure 4.32: Create Track User Interface

#### 4.3.2.4. They Found Me UI

Here the user can find the tracks that have a match with some people photos (temporary tracks). If he confirms any track, other users will be notified and this track will be marked so others can notice it.

This track confirmation leads to delete all the temporary tracks that are related to the same missing person.

Figure 33 shows they found me user interface (same page displayed on all screens).

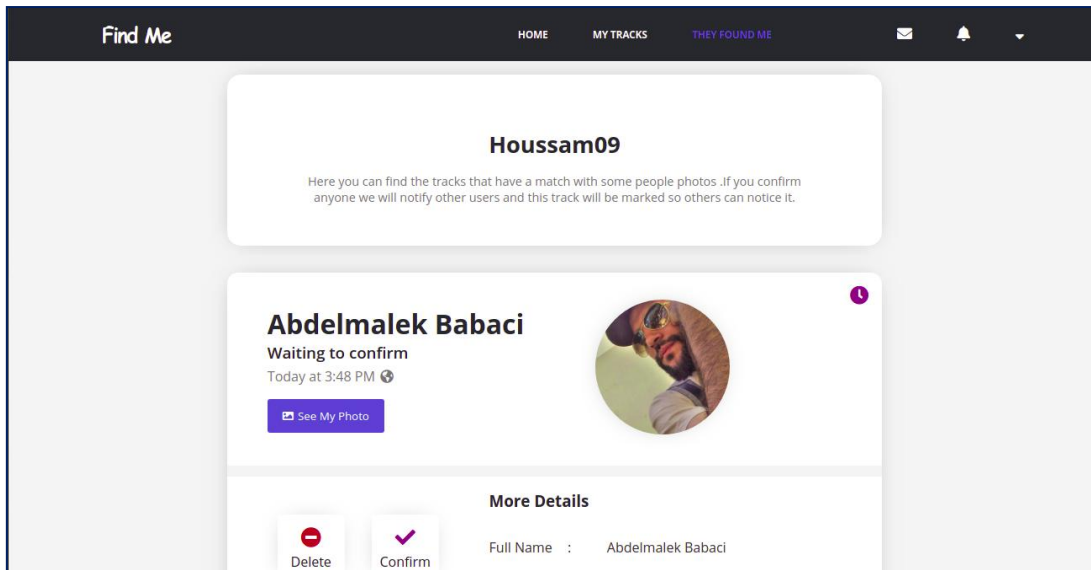


Figure 4.33: They Found Me User Interface

#### 4.3.2.5. Notifications UI

The notification is a message that is automatically sent to the user to tell him there has been an activity on the application, in our case this message is just an information about finding a missing person .

Figure 34 shows the notifications user interface.

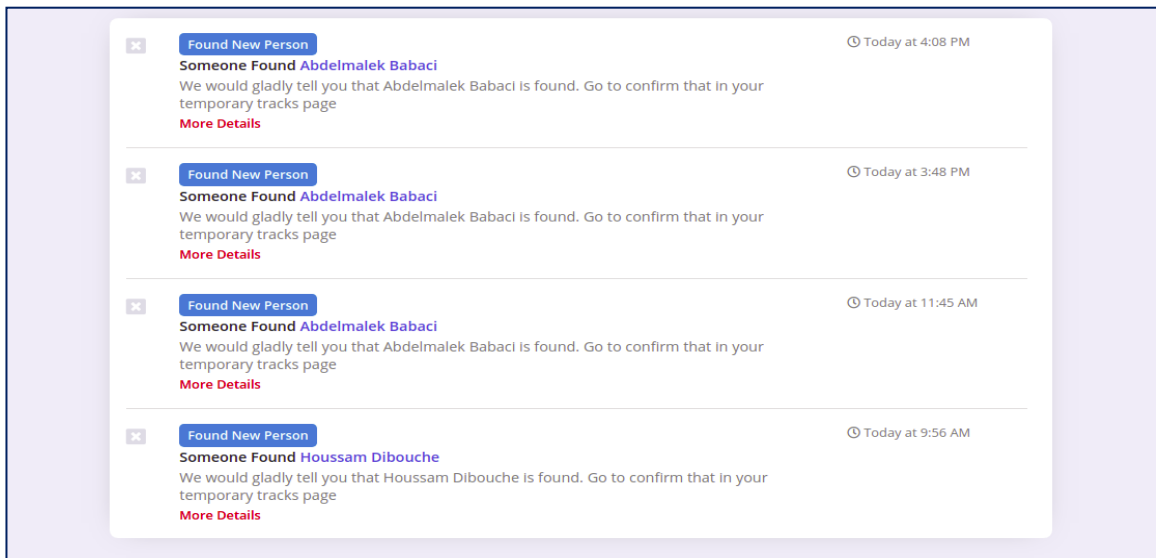


Figure 4.34: Notifications User Interface

### 4.3.2.6. Chat UI

Figure 35 and figure 36 show the chat user interface displayed on both large and small screens.

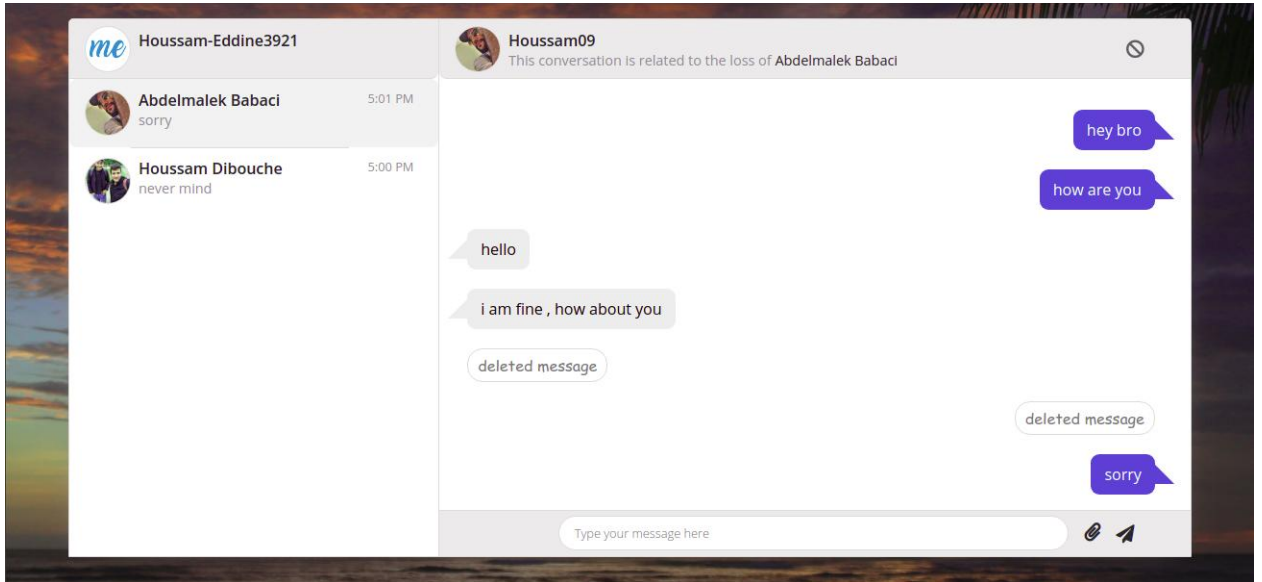


Figure 4.35: Chat User Interface On Large Screen

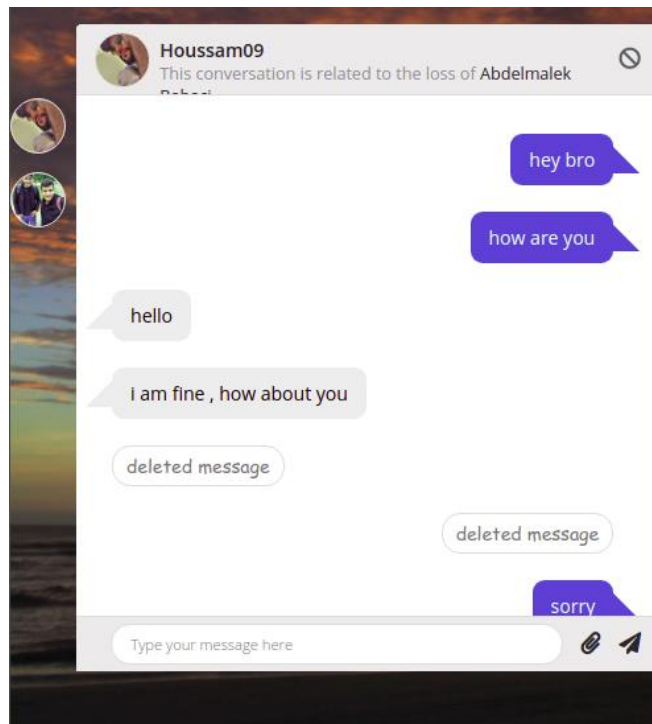


Figure 4.36: Chat User Interface On Small Screen

The chat UI consists of several parts (components) :

A chat message lists for displaying attachments, messages, and channels.

A chat message component that is simply a what a user shares.

A chat message input component, because Composing the perfect message takes a robust and flexible message input Ui , attaching images and files.

#### 4.3.2.7. Account Settings Page

##### 4.3.2.7.1. Profile UI

The profile contains the user personal information. Here the user can edit his email and password.

Email editing passes through a validation process (validation email). Figure 37 shows the profile user interface

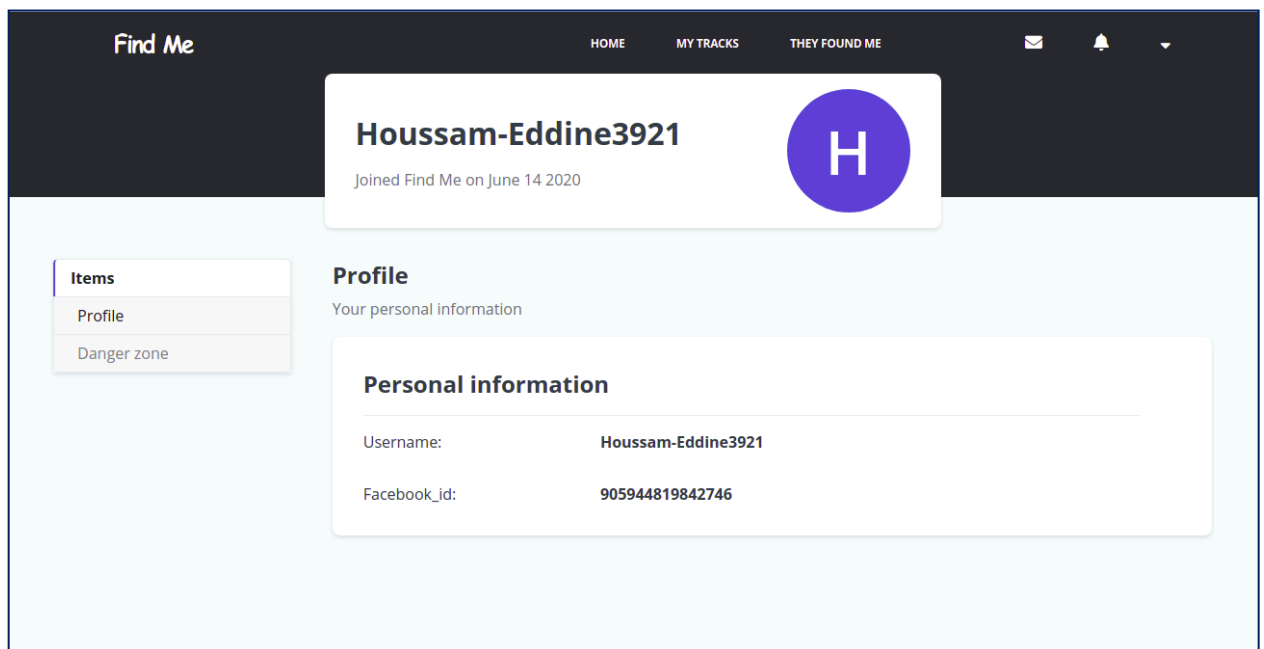


Figure 4.37: Profile User Interface

##### 4.3.2.7.2. Danger Zone UI

Here the user can delete his account using the password confirmation for the accounts that are created using the email. Figure 38 shows the danger zone user interface.

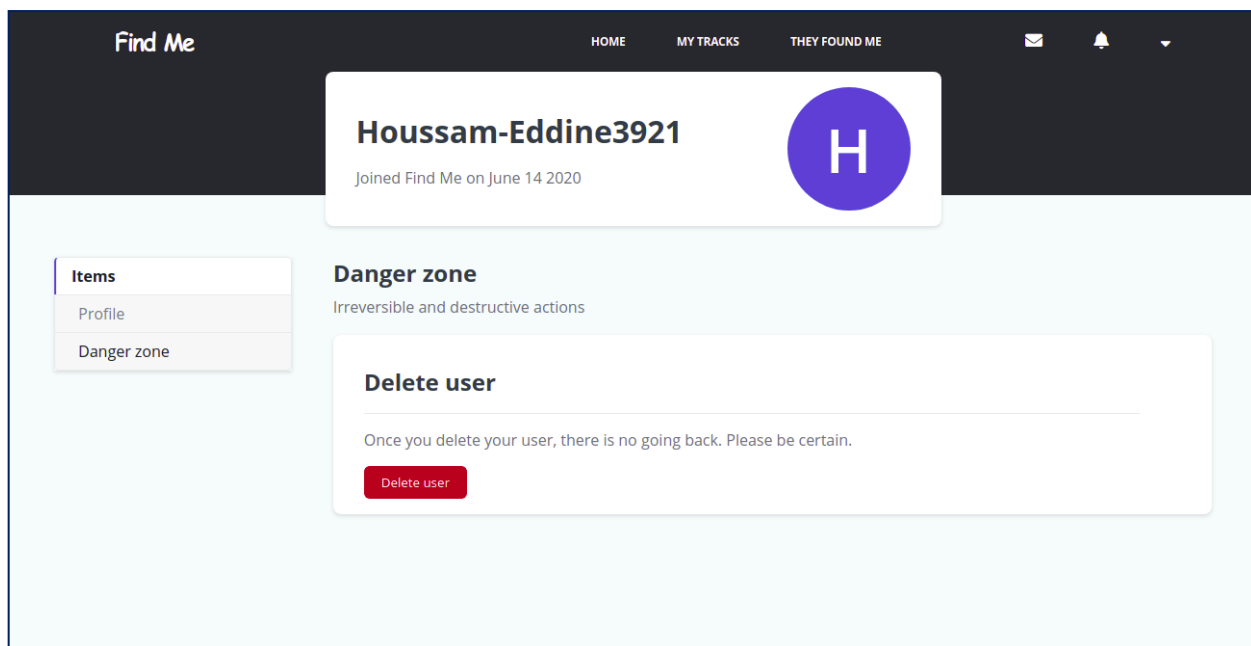


Figure 4.38: Danger Zone User Interface

#### 4.4. Limitations

We have tried to implement all the major functions in our applications but it still has several limitations:

- FindMe supports only English.
- FindMe is available only in Algeria.
- The mobile app is limited and supports only the automatic search feature.
- The face recognition process takes times and needs a high performance server.

#### 4.5. Conclusion

In this chapter we described the implementation of the FindMe application, we explain how to initialize the front-end and the back-end. We presented the important part of the code then we displayed the UI of the main pages of the web and the mobile application.

Finally, we talked about our project limitations and the main parts that could be improved on the future.

## Conclusion

Despite the limited time, we have completed our project in the given time period successfully. This project was an opportunity for us to learn many things including coding,



working in a team. We were also able to put our knowledge and our skills that we got in these last five university years into this project. This project also helped us getting familiar with platforms like Android, Ios, and Web.

We have created a functional, interactive, modular, evolvable, Web application called FindMe to help finding the missing people using their faces photos. FindMe provides a high quality system and a new searching mechanism (both manual and automatic) using the face recognition and the Google cloud Deep Learning that will play an important role in our lives. We have implemented the web client side with ReactJS, which is responsive for all the screens, and the server side with node.js and mongoDB. We have implemented the mobile application with flutter (framework of dart) which works on both IOS and Andriod. We have obtained a code of more than sixty thousand lines, resulting from generated codes, and our own source codes.

Today, the world is suffering from the corona virus that made creating FindMe difficult, we were not able to meet. We have tried to implement all the major functions in our application but it still has several limitations that we will improve in the future.

Future improvements include the design and implementation of a wearable IOT device, with appropriate embedded software and electronics, which complements the FindMe application. Persons with risk of getting lost (mentally disabled, Alzheimer sick people, children) could wear such a device, and in case of a missing person event, the application should register the photo and the name, and trigger the IOT to trace the missing person.

We can add more features to the mobile application. We can also work on making FindMe available in other languages than English, and making it available in other world regions, by giving it access to world missing person databases. The most important improvement should be the development of our own deep learning algorithm to process people's searches faster.

## Biographies

- 1- **Berners-Lee, T. (2000)** Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. New York: HarperBusiness.
- 2- Build a Login/Auth App with the MERN stack [Internet] Bits and Pieces **November 2018.**
- 3- **Comer, D. (1991)** Internetworking with TCP/IP (Volume 1: Principles, Protocols, and Architecture). Englewood Cliffs, NJ: Prentice-Hall.
- 4- **Dana Nourie.** Java technologies for Web Applications - Oracle official website **Accessed 2 June 2016.**
- 5- **Daniel Deutsch ,04 February 2017** - Understanding MVC Architecture with React
- 6- **Davidson, J. (1988)** An Introduction to TCP/IP. New York: Springer-Verlag.
- 7- **Jacksi, Karwan & Abass, Shakir. (2019).** Development History Of The World Wide Web. International Journal of Scientific & Technology Research.
- 8- **Hafner, K. and Lyon, M. (1996)** Where Wizards Stay Up Late: The Origins of the Internet. New York: Simon & Schuster.
- 9- **Hanin M. Abdullah, Ahmed M. Zeki,2014,p.85-89.** Frontend and Backend Web technologies in Social Networking Sites: Facebook as an Example. Advanced Computer Science Applications and Technologies (ACSAT), IEEE 3<sup>rd</sup> International Conference .
- 10- **Hernandez A,Accessed10 July 2016.** Init.js: A Guide to the Why and How of Full-Stack JavaScript Toptal LLC developers' official website.
- 11- **JQuery API-JQuery officialwebsite accessed 10july 2016.**
- 12- **Krol, E. (1994)** The Whole Internet User's Guide and Catalog, Second Edition. Sebastopol, California: O'Reilly.
- 13- **Mimi Gentz,24 June 2016 Accessed 3june 2016.** NoSQL vs SQL - Microsoft Azure official website.
- 14- **Mozilla developers, 7July 2016'** official website.
- 15- **Nelson, T. H. (1982)** Literary Machines 931. Sausalito, California: Mindful Press.

- 16- React.js, 3 September 2018:** a better introduction to the most powerful UI library ever created [hackernoon.com](http://hackernoon.com).
- 17- R. Max Wideman (2004),** A Management Framework: For Project, Program and Portfolio Integration. P. 30 Banker, Kyle (March 28, 2011), MongoDB in Action (1st ed.), Manning, p. 375, ISBN 978-1-935182-87-0.
- 18- Rodriguez A, 9 February 2015 - Accessed 5 June 2016.** RESTful Web services: The basics - IBM official website.
- 19- Rosenfeld, L. and Morville, P. (1998)** Information Architecture for the World Wide Web. Sebastopol, California: O'Reilly & Associates.
- 20- Stephenson, N. (1999)** In the Beginning Was the Command Line. New York, NY: Avon Books.
- 21- The modern application stack MongoDB Official Website - 26 January 2017**
- 22- Vincent Mühler, 25 June 2018 -** [face-api.js](#) "JavaScript API for Face Recognition in the Browser with tensorflow.js" .
- 23- Web Applications: What are They? What of Them -** Acunetix official website **Accessed 3 June 2016.**
- 24- What exactly is Node.js? freeCodeCamp - 18 April 2018.**
- 25- What is ReactJS and Why should we use it? .c-sharpcorner.com - 14 November 2018.**
- 26- What's AJAX? - Mozilla developers' official website. Accessed 10 July 2016**
- 27- Williams, R. and Tollett, J. (2000)** The Non-Designer's Web Book. Berkeley, California: Peachpit Press.
- 28- Wood, D. (1999)** Programming Internet Email. Sebastopol, California: O'Reilly