UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

“Design and development of a service discovery community”

Diploma Thesis

Tsoukala Georgia

Supervisor: Tsalapata Harikleia

Volos 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

“Design and development of a service discovery community”

Diploma Thesis

Tsoukala Georgia

Supervisor: Tsalapata Harikleia

Volos 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

“Σχεδιασμός και ανάπτυξη κοινότητας αναζήτησης υπηρεσιών”

Διπλωματική Εργασία

Τσουκαλά Γεωργία

Επιβλέπων: Τσαλαπάτα Χαρίκλεια

Βόλος 2022

Approved by the Examination Committee:

Supervisor: **Tsalapata Harikleia**

L.T.S, Department of Electrical and Computer Engineering,
University of Thessaly

Member: **Stamoulis George**

Professor, Department of Electrical and Computer Engineering,
University of Thessaly

Member: **Daskalopulu Aspassia**

Assistant Professor, Department of Electrical and Computer Engineering,
University of Thessaly

Date of approval: 05/07/2022

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

The Declarant

[illegible]

Tsoukala Georgia

Acknowledgments

First and foremost, I would like to thank my supervisor, Tsalapata Harikleia, for her support and guidance throughout the development of this project and the writing process of this Thesis.

Also, I would like to give my gratitude to my friends and family for their continuous and unwavering support and understanding throughout the years of my studies. I would not have made it without their help.

Diploma Thesis

“Design and development of a service discovery community”

Tsoukala Georgia

Abstract

The global services market is one of the largest and fastest-growing markets and is expected to keep growing. Nowadays, only a very small percent of the home services market is online, in contrast with the digital disruption of countless other industries. The current way people consume services and professionals market their businesses, has many disadvantages and alternatives are needed. Some of the issues that stem from it can be solved with the use of online on-demand home services.

In this Thesis, a mobile application for home services, that matches service professionals with customers in their local area, is being studied and implemented. The application, called FixIt, is going to be an ecosystem where clients can connect with the most suitable tradesperson.

Keywords:

mobile application, community, home services, service professionals, customers, React Native, Expo, Firebase

Διπλωματική Εργασία

“Σχεδιασμός και ανάπτυξη κοινότητας αναζήτησης υπηρεσιών”

Τσουκαλά Γεωργία

Περίληψη

Η παγκόσμια αγορά υπηρεσιών είναι μία από τις μεγαλύτερες και ταχύτερα αναπτυσσόμενες αγορές και αυτό αναμένεται να συνεχιστεί. Σήμερα, μόνο ένα πολύ μικρό ποσοστό της αγοράς υπηρεσιών σπιτιού είναι online, σε αντίθεση με την εκτεταμένη διαδικτυακή παρουσία άλλων βιομηχανιών. Ο σημερινός τρόπος με τον οποίο οι άνθρωποι καταναλώνουν τις υπηρεσίες αυτές αλλά και οι επαγγελματίες προωθούν την επιχείρηση τους, παρουσιάζει αρκετά μειονεκτήματα και γι’ αυτό χρειάζονται εναλλακτικές λύσεις. Μερικά από τα ζητήματα που προκύπτουν από αυτόν τον τρόπο μπορούν να επιλυθούν με την βοήθεια των διαδικτυακών on-demand υπηρεσιών.

Σε αυτή τη διατριβή, μελετάται και υλοποιείται μια εφαρμογή κινητών τηλεφώνων για εύρεση επαγγελματιών για εργασίες σπιτιού, η οποία φέρνει σε επαφή τους πελάτες με τους επαγγελματίες της περιοχής τους. Η εφαρμογή, που ονομάζεται FixIt, πρόκειται για ένα οικοσύστημα όπου οι πελάτες μπορούν να συνδεθούν με τον πιο κατάλληλο μάστορα.

Λέξεις-κλειδιά:

εφαρμογή για κινητά, κοινότητα, εύρεση επαγγελματιών, εργασίες σπιτιού, επαγγελματίες υπηρεσιών, πελάτες, React Native, Expo, Firebase

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

Ad	Advertisement
API	Application Programming Interface
APNs	Apple Push Notification service
App	Application
CLI	Command-Line Interface
FCM	Firebase Cloud Messaging
GPS	Global Positioning System
GUI	Graphical User Interface
Info	Information
NoSQL	Non Structured Query Language
OS	Operating System
QR	Quick Response
SDK	Software Development Kit
SDLC	Software Development Life-Cycle
UI	User Interface
URI	Uniform Resource Identifier
UX	User Experience

1 Introduction

1.1 Project Overview

In this project, the goal is three-fold: first, to understand the way people consume services nowadays and identify the problems that stem from it for both customers and professionals, second, to review existing applications, and third, to produce a new mobile application, based on the findings of this study, which offers direct communication between clients and service professionals.

The application, called FixIt, consists of two parts, a User Interface, front-end part, written in React Native [1] and a database, back-end part, implemented in Firebase [2]. FixIt is designed to run on every device, regardless of the operating system (OS).

1.2 Stakeholders

Stakeholders [3] (Figure 1) are individuals or groups whose interests may be affected by the success or failure of a project. The people that have a direct or indirect interest in this mobile application are presented below:

1. Users: Ones that use the app either as customers or as professionals. Their needs and desires determine the main requirements of the end product.
2. Developers: Mobile application developers who work on similar projects and want to use it as an inspiration or a guide.

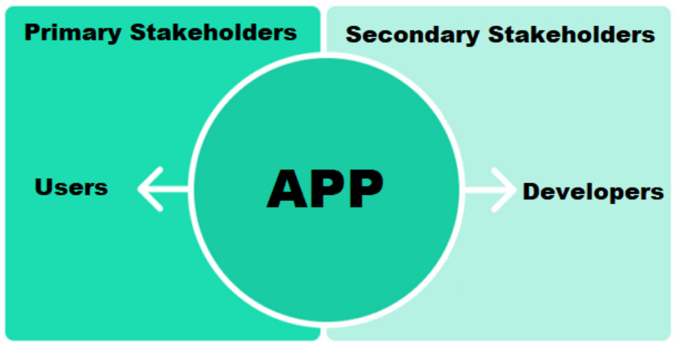


Figure 1 Types of stakeholders

1.3 Project Goals and Objectives

There are different objectives set amid the creation of this project. By accomplishing them, certain skills get also excelled. Some of these goals are denoted below:

a. Study of the way people consume services nowadays and identification of the problems that stem from it.

b. Review of existing apps.

c. Investigation of available tools for the design and the development of a mobile application and selection of the most suitable ones.

d. Design and implementation of a mobile app.

e. Assessment of the mobile application's practicality. The evaluation will be based on both customers' and professionals' experience from the real-time use of the app.

f. Proposal of future additions and optimizations.

1.4 Thesis Organization

This Thesis consists of six chapters, excluding the Introduction. Chapter 2 analyzes the current home services market and presents the challenges that both professionals and customers face. Chapter 3 contains reviews of existing applications for home services and showcases some key characteristics of FixIt. Chapter 4 analyzes the software development process, mentions some principles of mobile interface design and includes a requirement analysis. Chapter 5 provides a small presentation of the tools used to build the application as well as a detailed description of the design and the implementation of the app. Chapter 5 presents the evaluation and testing of the project. Finally, in Chapter 7 some conclusions are drawn and future work is proposed.

2 Current Situation and Problems

2.1 Customers

The home services [4] industry is a broad term that refers to businesses whose main purpose is working on residential homes. This includes plumbing, HVAC, cleaning services, flooring, landscaping, moving and storage services, and more. Finding the right professional or home service company for home repairs, maintenance or renovation can be a tedious and time-consuming process.

2.1.1 Ways to find a service professional

Hiring the right handyman is key considering that a good one can save the customer time and money. Finding a qualified service professional that is reliable and has all the required licenses provides peace of mind and consumer protection in the event of an accident or something going wrong.

The primary method people use to find a worker is word-of-mouth [5]. According to statistics, 83% of people trust recommendations from friends and family. By asking people for referrals, a customer can get a sense of what kind of work a specific professional does, how quickly they finish projects, and whether or not they're dependable. Word-of-mouth reviews generally rely on more than the review itself, many times witnessing the result of a service a trusted friend, neighbor or colleague had, tells a lot about the person they hired.

Another way customers find home service professionals is from advertisements [6]. It can be of any type: print advertisements like magazine or newspaper ads and flyers, television or radio advertisements, outdoor advertisements like billboards, ads on the sides of buses or inside subway cars, etc. Advertisements can reach customers in various places.

In today's digital world, an internet search is the easiest way to find a handyman in your area. A large portion of service professionals market their businesses online [7]. The customer may be able to see pricing information, ratings and reviews for their work on their website, social media accounts or targeted platforms.

2.1.2 Problems and what to look for

Some of the most common methods homeowners use to find the most suitable professional are three: recommendations, advertisements and internet searches. But, there are some problems and risks when hiring someone with one of these methods.

The main issue of the word-of-mouth method [8] is that the number of friends and acquaintances is limited, and so is the information and consequently the options when asking for referrals. Another problem that arises comes from the different evaluation criteria each person has, both for the work provided and the cost price. These can influence the judgment and evaluation of a worker. Also, the personal relationships that a friend, neighbor or colleague may have with a professional may affect them and they may not be able to objectively judge the quality of that professional's work.

Print advertisements, television or radio advertisements, and outdoor advertisements usually misrepresent professionals or businesses and make tall claims to excite clients. Advertisements might be deceitful [9] as they often conceal facts, resulting in problems, unforeseen expenses and inconvenience. Also, the information provided by an ad is usually little due to the limited space (print advertisements, outdoor advertisements) or time (television advertisements, radio advertisements) so the customer can not easily form an opinion. It is also difficult to watch all ads in order to know all the available professionals in an area. Lastly, many times the cost of advertising is added to the total cost of service.

The internet is the medium that most people nowadays use when they need to find something. It is the primary mean of search. But many times it confuses the user with the amount of available information. Also, it is not easy to evaluate and decide on the reliability of that information. Another thing that needs to be pointed out and taken into account is that surfing the internet requires certain skills that some people do not have.

2.2 Professionals

Home services is a booming industry. The economy is growing, society is becoming less and less handy, houses are getting more complex and people are outsourcing more of their lives every day, creating a high demand for home services. But handyman businesses aren't just in competition with one another. They also have to contend with manufacturers and dealers that provide repair services, as well as clients' friends and family members who are handy with tools.

2.2.1 Ways to market a handyman's business

Being an expert in your field is not always enough. One of the challenges professionals face is maintaining the existing clientele while searching for new clients, especially when most people are wary of unfamiliar faces. That results in understanding how to advertise the business both offline and online being crucial.

Word-of-mouth is one of the most effective types of marketing [10]. Incentivizing client referrals can have a positive impact on the growth of the business. The first step toward obtaining referrals is to produce high-quality work. Also, a simple referral program that rewards customers for referring new customers can help in expanding the clientele.

Another effective way to promote a business’s services is advertising [11]. Advertisements can reach customers in various places, including while watching television, walking around town or browsing the internet. There are many types of ads a company can choose from based on its budget or the consumers it wants to target. Some of them are magazine or newspaper ads and flyers, TV ads, billboards, etc. Also, advertising directly on the company's vehicle can be a direct and affordable way of marketing.

Digital advertising serves as a more modern option. With technology, companies gain new ways of targeting consumers. Having an active presence online is crucial. For example, a website can display information like services, rates, contact information and a biography. Also, an active social media [12] presence can help in engaging with existing customers and followers.

2.2.2 Problems and challenges

Professionals usually rely on recommendations, advertisements and online means to find new clients and grow their businesses. But, there are some problems and risks when trying to expand the business's clientele with one of these methods.

A problem that arises with the word-of-mouth method is that friends, family or neighbors are limited so the number of potential new customers based on their referrals will be also limited. The same is true with the professional's previous clients, their clientele can not grow satisfactorily based only on their acquaintances and the acquaintances of their old clients. Also, among the previous customers, there will be dissatisfied ones and bad criticism [13] can spread very quickly and have lasting effects on a company. With word-of-mouth advertising, the customers control the marketing message.

Professionals also try to attract customers through print advertisements, television or radio advertisements, and outdoor advertisements. Ads can be very expensive, especially for small businesses and the results are never guaranteed. In addition, there is a great number of platforms that accept advertisements but the intended targets of an advertising campaign may never see it. Another problem that arises is many prospective clients believe that a lot of advertisements misrepresent professionals or businesses and try to deceive them. Finally, the message of an advertising campaign can get lost in the white noise of all the brand messages a person gets exposed to daily.

Marketing a handyman business with online means is another option. But, having a website to display a business's information can be costly not only to make but also to maintain. On top of that, for the website to show up toward the top of the search results on a search engine [14] is not easy. Also, maintaining an active social media presence requires a lot of time and effort in order to post regularly.

3 Solution and Existing Applications

3.1 Existing Applications

According to a New York Times poll, the market for on-demand home services [15] in the United States is presently worth $600 billion and exhibits consistent growth.

Nowadays, using mobile apps is getting popular among businesses and service providers [16]. Some of the problems mentioned in the previous chapter, that stem from the current way most people consume services, can be solved with the help of online platforms. On-demand handyman apps are online platforms that connect customers with professionals that provide household repairs and services.

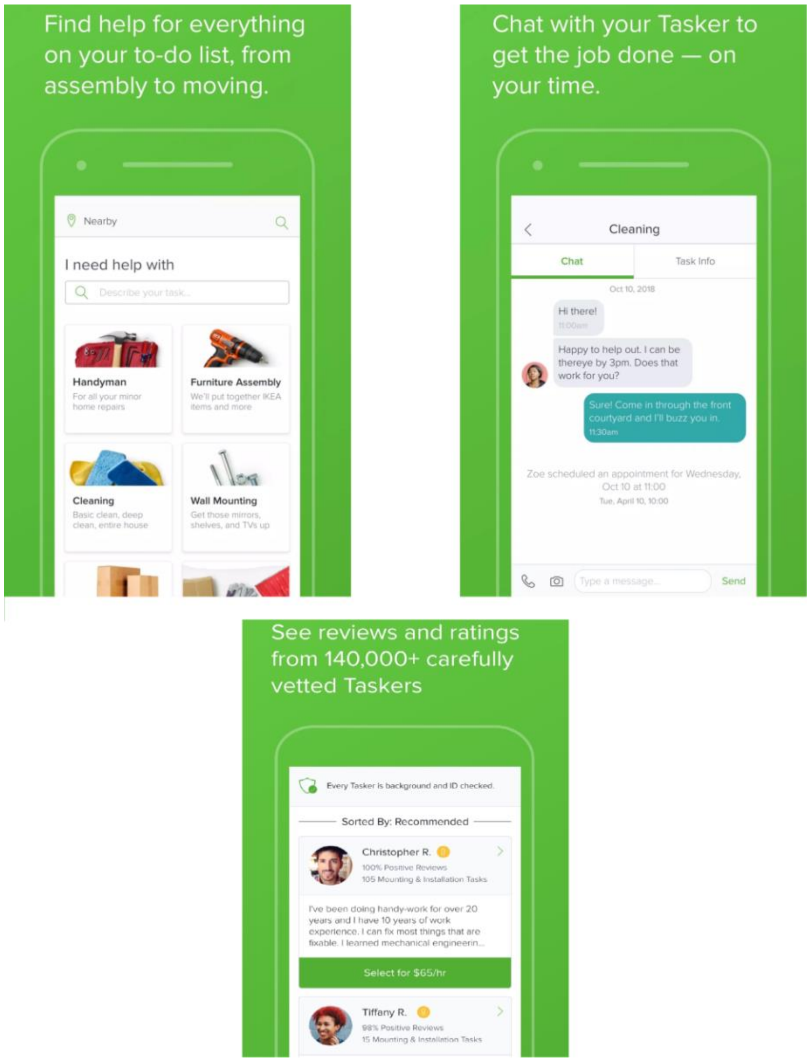
A mobile app-based handyman service, allows the customer to choose from an extensive network of experienced professionals and contractors. The mobile apps aim to simplify the hiring process. At the same time, professionals can find new clients and widen their clientele.

There is a vast availability of handyman service apps in the market with a similar workflow. Some key characteristics are listed below:

a. The user creates an account in the app (free or with a fee).
b. Customers provide personal information.
c. Professionals provide information about their business.
d. The customer selects the type of service needed.
e. The app displays available professionals that match the search criteria.
f. If a customer is satisfied with a professional they can proceed further by completing the hiring process.
g. The professional will show up at the client’s location.

Some of the most popular home service applications are presented below:

TaskRabbit [17] (Figure 2), which is owned by IKEA, is a platform where clients search for handymen (called Taskers) who can help with odd-jobs and errands. Taskers have to pay a non-refundable registration fee and meet a list of requirements including an age limit, a mandatory background check, and the right to work in the country of residence. Taskers will be notified of potential jobs around them via the app and if any of them suits them they can express their interest in it. The app runs both on Android and iOS devices and is available in the United States (36 states), Canada, United Kingdom, France, Spain, Portugal, Italy and Germany.


Find help for everything on your to-do list, from assembly to moving.

Chat with your Tasker to get the job done — on your time.

Hi there!
11:00am

Happy to help out. I can be thereye by 3pm. Does that work for you?

Sure! Come in through the front courtyard and I'll buzz you in.
11:30am

Zoe scheduled an appointment for Wednesday, Oct 10 at 11:00
Tue, April 10, 10:00

See reviews and ratings from 140,000+ carefully vetted Taskers

Every Tasker is background and ID checked.

I've been doing handy-work for over 20 years and I have 10 years of work experience. I can fix most things that are fixable. I learned mechanical engineerin...

Figure 2 TaskRabbit application

Urban Company (UrbanClap) [18] (Figure 3) is the leading home service provider app in India, UAE, Saudi Arabia, Sydney and Singapore. UrbanCompany connects customers to service professionals to meet their daily needs. It helps customers book reliable and high-quality services like beauty treatments, massages, haircuts, repairs, painting, cleaning, pest control and more and have them delivered by professionals conveniently at home. It is a subscription fee-based platform where workers need to pay a monthly subscription.

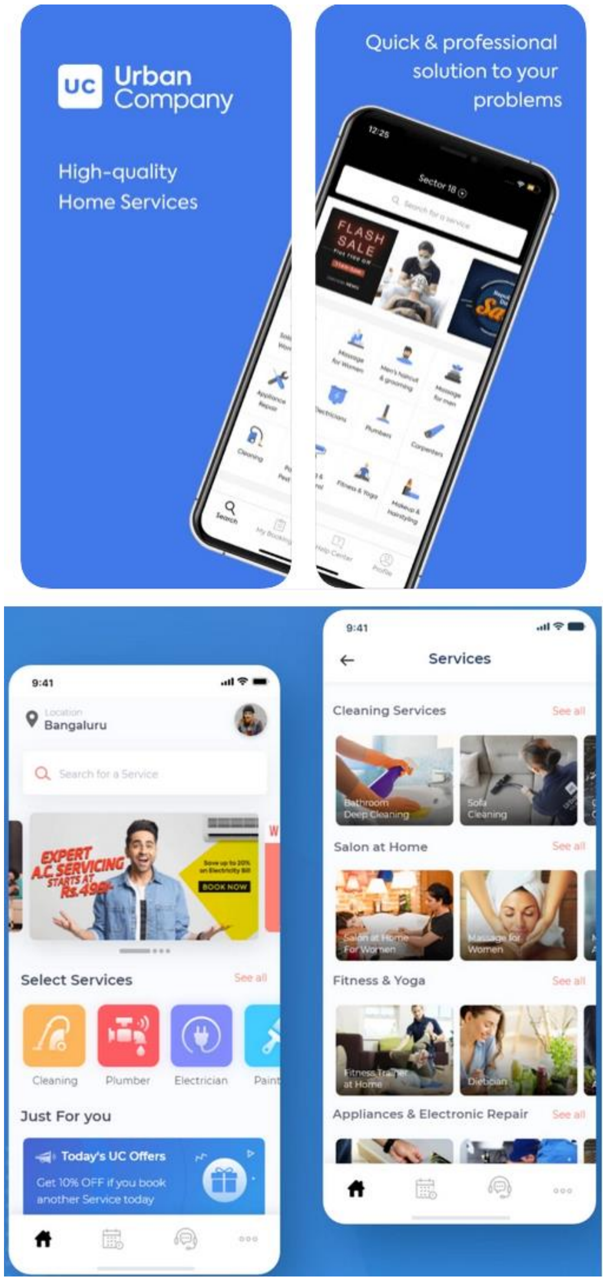
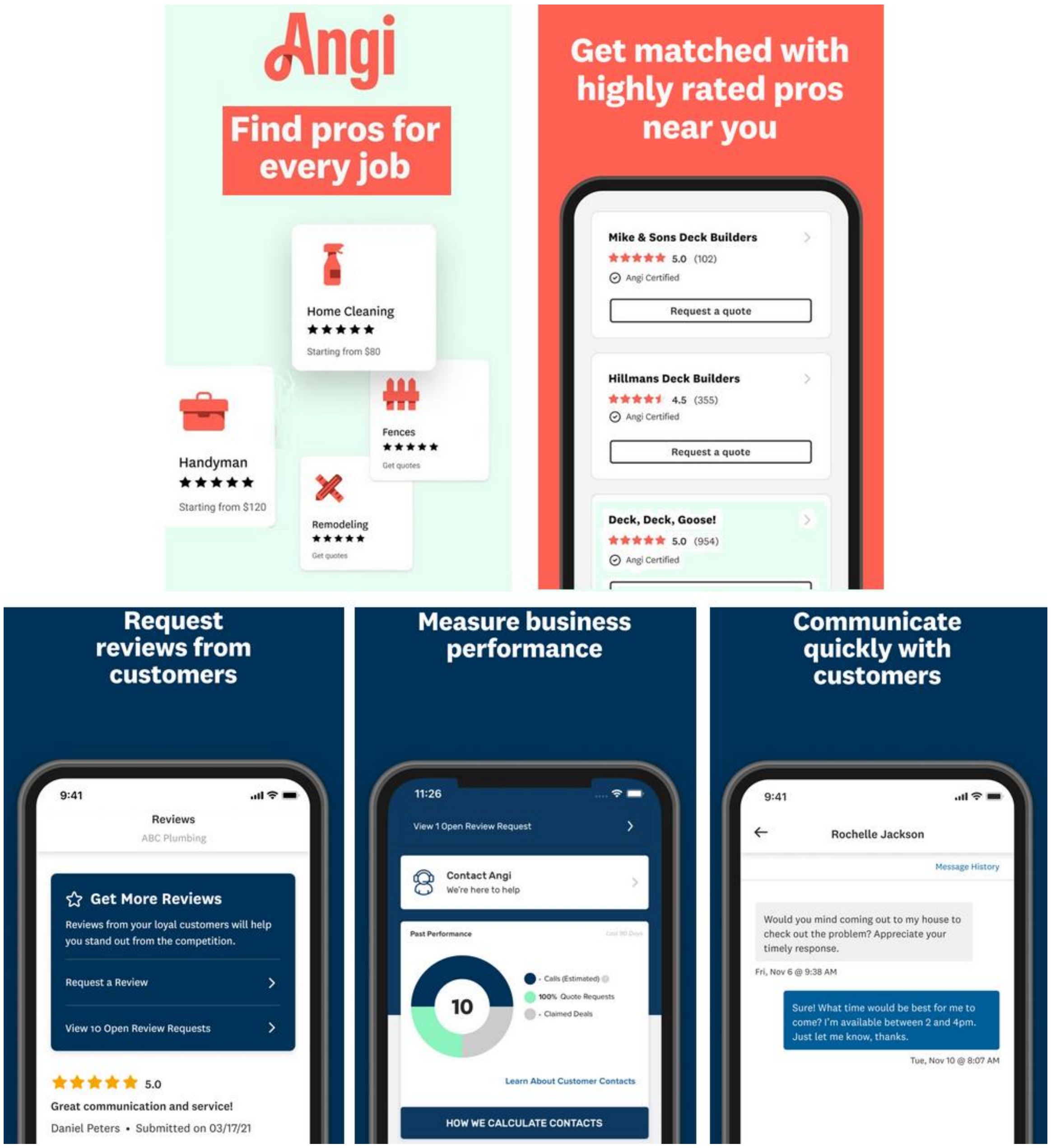


Figure 3 Urban Company application

Angi [19] (Figure 4) is an internet services company formed by the merger of Angie's List and HomeAdvisor. With a combination of cutting-edge technology and tried-and-true neighbor referrals, it makes the process of hiring a handyman and contractors simpler. Angi allows users to select from categories that include remodeling, cleaning, builders, etc. It also provides the professionals exclusive tools to manage and showcase their business and increases its exposure to millions of homeowners. It requires a fee and covers many United States cities.



Reviews from your loyal customers will help you stand out from the competition.

Great communication and service!

Daniel Peters • Submitted on 03/17/21

Would you mind coming out to my house to check out the problem? Appreciate your timely response.

Sure! What time would be best for me to come? I'm available between 2 and 4pm. Just let me know, thanks.

Figure 4 Angi application

Douleutaras (YourHero /YourPro) [20] (Figure 5) is a tech-enabled company for home services funded in Greece. Operating also in Cyprus, Ireland and Spain is one of the leading companies in the home services sector in these countries. No registration fee is required for either the customers or the professionals. When a job interests a professional, they can obtain the client's contact information for a small fee.

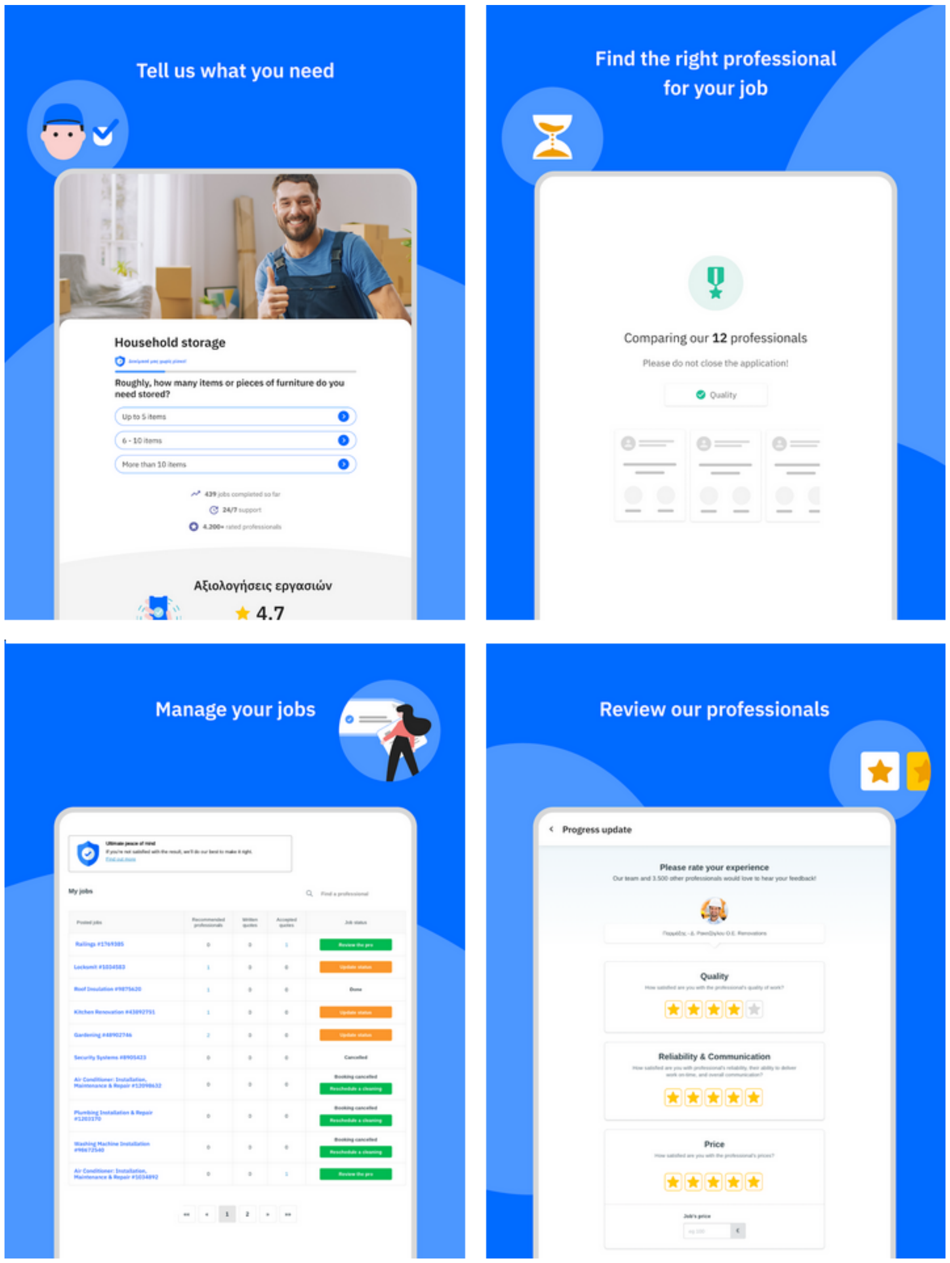

Tell us what you need
Household storage
Roughly, how many items or pieces of furniture do you need stored?
Up to 5 items
6 - 10 items
More than 10 items
Αξιολογήσεις εργασιών
4.7
Find the right professional for your job
Comparing our 12 professionals
Please do not close the application!
Quality
Manage your jobs
My jobs
Review our professionals
Progress update
Please rate your experience
Quality
Reliability & Communication
Price

Figure 5 Douleutaras application

3.2 FixIt Application

The application called FixIt (Figure 6), which is designed and developed in this project is going to be an ecosystem where clients can connect with the most suitable professionals. FixIt will provide a solution to the problem of finding reliable home services professionals while at the same time enable professionals to experience unlimited business opportunities.

Figure 6 FixIt logo

Houses, commercial buildings and offices need timely maintenance and repairs with plumbing, painting, cleaning, AC repairs, and other services. Several professionals are available for such work, but people only want the best workers to do the finest job on their property. Consumers, when searching to hire a professional, are looking for reliable options, speed of service and safety and FixIt provides those.

FixIt provides all the home-related service options on the same platform, making it convenient for users to look into different services in the same app. Creating an account on FixIt does not require a subscription fee for either the customers or the professionals. Also, searching on the platform is a completely free process.

Finding the right people to do the required work usually involves a lot of time searching or asking friends and family for referrals. With Fixit, the customer does a thorough market research easily and quickly. A list of professionals that meet the user's search criteria is provided by the app. For each professional other than their personal info, a rating and reviews from previous clients are provided, helping users to select service providers with higher ratings.

Feedback from former clients helps the user to get an idea of the quality of work each service provider offers. The more satisfied customers the bigger the confidence in the quality of work of a specific professional. Hiring decisions of users primarily rely on ratings and preceding reviews of other customers. Once the customer has completed their service, they can also leave a review about the professional.

FixIt allows service professionals to connect with a greater audience and transform their service offerings into a wholesome business. Also, reviews for the quality of services posted by previous customers can be used as references. In a way, positive reviews can spread a good word for a professional. A BrightLocal study [21] found that 86% of consumers read reviews for local businesses and these reviews have been shown to influence up to 67% of purchasing decisions.

Registration and Login
Advanced Search
Rating and Review

Figure 7 Key features of FixIt

4 Analysis

4.1 Software Development Process

The Software Development Process [22], also known as the software development life cycle (SDLC), is a set of steps a system or project goes through when developed. Several approaches bring a product's concept from the ideation stage to implementation and eventually to the market. However, all these different approaches cover the following same six stages (Figure 8):

1. Requirement analysis
2. Software design
3. Software development
4. Testing
5. Deployment
6. Maintenance

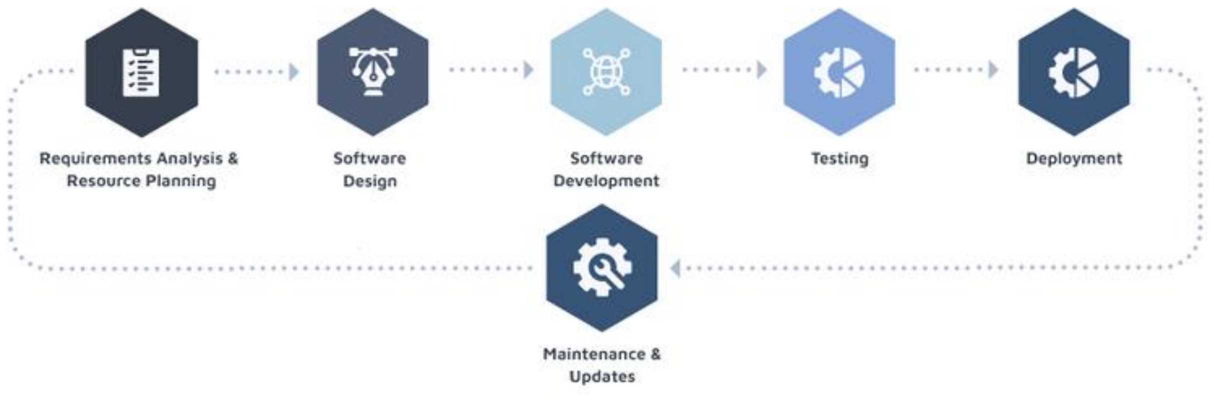


Figure 8 Stages of Software Development Process

Some widely used SDLC models [23] are detailed below followed by the one that is adapted in this project.

4.1.1 Software Development Process Models

The Waterfall model [24] (Figure 9) is one of the oldest and best-known processes where the developers follow all the steps mentioned previously, in perfect order. It is a linear model that is easy to use and understand because each phase is processed and completed before the next one begins. The Waterfall Process is suitable for small projects where milestones are well understood, as it is great for planning and management. The disadvantage of waterfall development is that it does not allow much revision because working software is produced late during the life cycle.

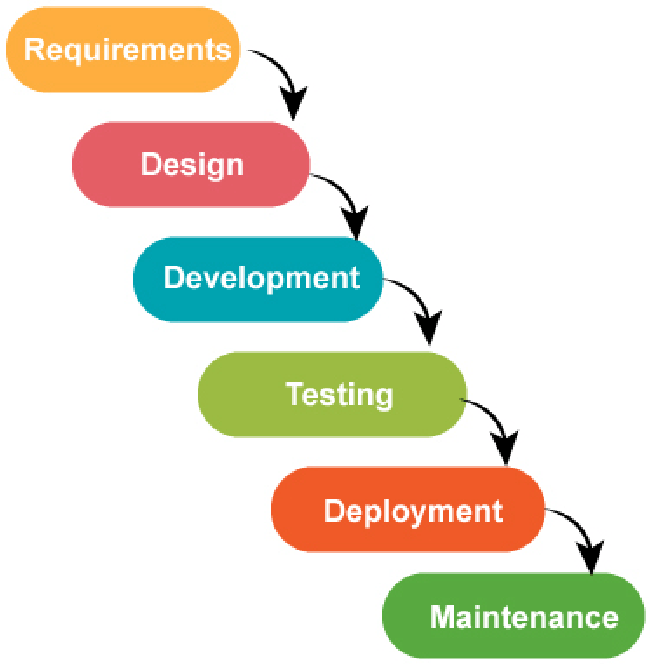


Figure 9 Waterfall model

The Incremental model [25] (Figure 10) applies the waterfall model incrementally. It breaks the software development process down into a series of releases known as increments. Each one builds on the previous version so that improvements are made step by step. This method allows easier testing and debugging because relatively smaller changes are made during each iteration. It also avoids a long development time as the initial product delivery is faster. As additional functionality is added at every step, problems that were not evident in earlier stages may arise.

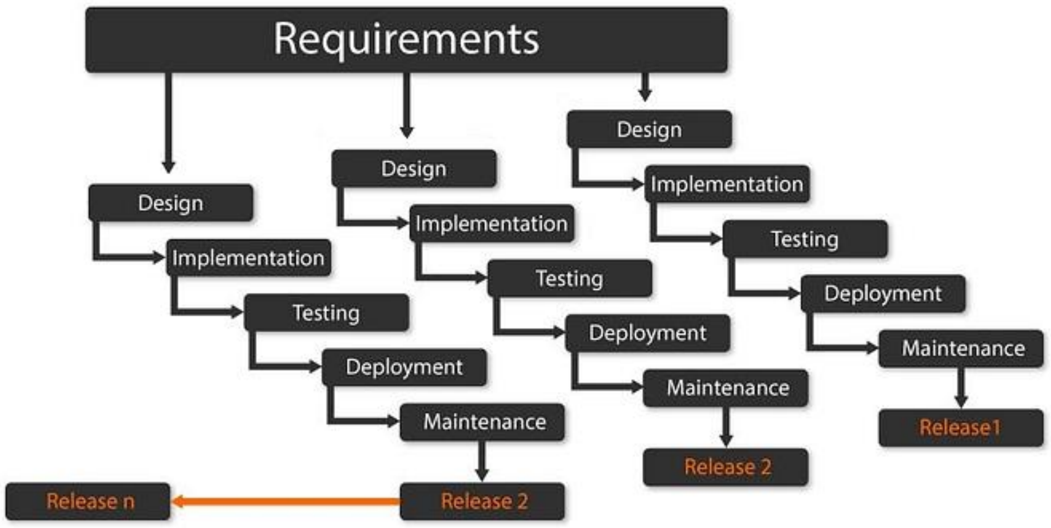


Figure 10 Incremental model

The Spiral model [26] (Figure 11) is similar to the Incremental model but with a high emphasis on risk analysis. It has four phases: Planning phase, Risk analysis phase, Engineering phase and Evaluation phase. In each iteration, called spiral, the project passes through all these phases. This method is favored for large and complicated projects which require continuous enhancements.

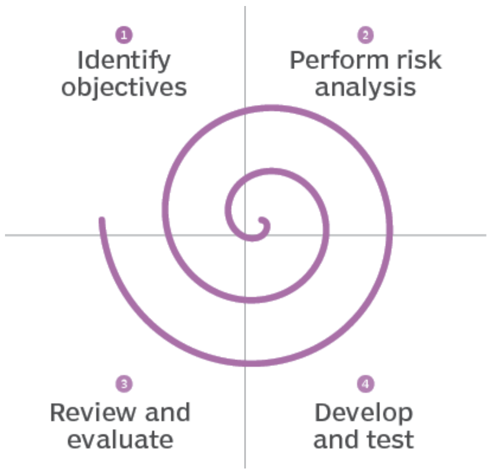


Figure 11 Spiral model

4.1.2 Approach

For this project, the Agile model [27] (Figure 12) was chosen as the most suitable development process. The Agile method uses an adaptive approach whereas the models mentioned previously are based on a predictive approach. It does not require a complete forecast of the exact features to be developed and adapts to the changing requirements dynamically. The partial working solution released in each iteration is thoroughly tested, reducing the likelihood of problems arising in the future. This model is a very realistic approach, easy to manage and offers flexibility to the developer because it allows to easily change the requirements when needed and also thanks to the small incremental releases, the application's design can be quickly improved, and new features can be added and bugs can be fixed.

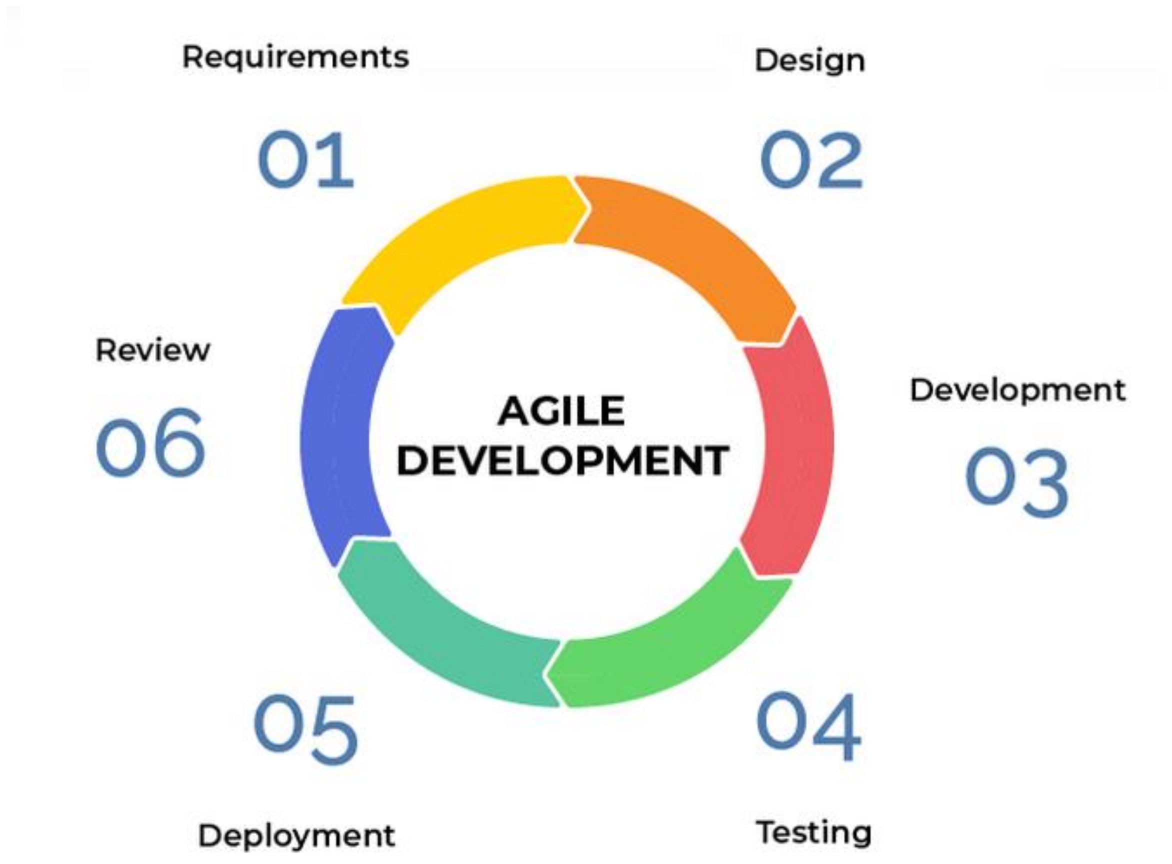


Figure 12 Agile model

The project implemented in this Thesis is a mobile application, which requires good design and technical excellence. Many increments will be made to produce a successful application, each of which will be an improved version of the prior release. Changes in requirements and design enhancements will occur with each increment, resulting in new software development cycles.

In this project, all the phases of the agile model are followed. More specifically, the following steps will take place:

a. Requirements specification: Identify the requirements for the project development by establishing a complete description of the behavior of the system to be developed.
b. Design: Identify the important components of the system and define the system's architecture.
c. Implementation: Implement the system's design into code.
d. Evaluation and Testing: Test that the system code has met its specification, evaluate the success of the previous phases and make the necessary modifications to them.

An important note is that the use of the agile model also means that the order of the phases in the following chapters is not absolute (Figure 13). This means, for example, that if new requirements are needed during the design process, then going back to the requirements chapter and defining them is necessary.

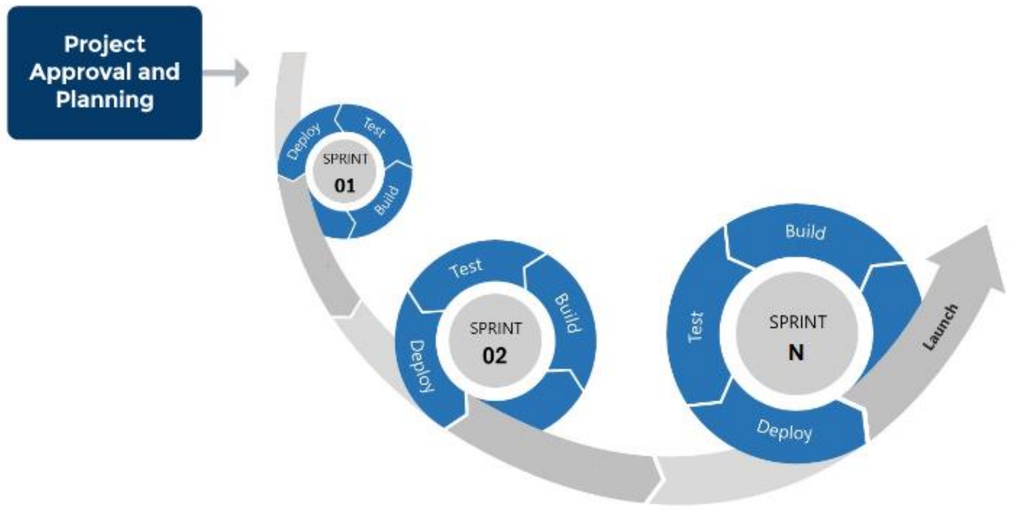


Figure 13 Generic agile project process

4.2 Mobile Devices

Mobile devices and computers have a lot of crossover but, they also have different strengths and weaknesses. Some of the differences mobile devices have are:

a. Small screen
b. Battery-powered
c. Spotty connection
d. Expensive data
e. Limited storage
f. Distracted user
g. Touch vs. mouse
h. Personal
i. Always on
j. Always with
k. Usually connected
l. GPS, Gyroscope, Camera, etc

These differences mean that a developer needs to consider a few important things when designing a new mobile application because a mobile environment is quite different from a computer one.

<u>Small Device:</u> Mobile devices have relatively small screens and they cannot support multiple windows. As a solution, smaller sizes of layered information are usually used by developers, such as drop-down menus, pop-up windows, or small dialog boxes. Another problem small screens have is that it is difficult for users to read large texts because the context can be easily lost while scrolling up and down. Thus, the content and the layout of a mobile application should be carefully designed and tailored for a small screen in a way that will help users focus while reading or interacting with it. Also, users can interact with only one application at a time on mobile devices. That means that every app must be characterized by responsiveness and provide its functionalities very fast. Developers could pre-load data whenever possible, to decrease the delay time.

Personal Device: Mobile devices are personal devices. An application should be designed to hide any password entry or secure input data, as it is easier for a user to hide the screen of their device rather than hide the device’s keyboard. Also, an application could provide options to store some private information like login details, to allow faster usage without the need of re-typing the entry data.

Customizable Device: An application should be attractive to all users and provide the option to use different themes (dark or light) based on their needs. Also, some users may have difficulties in reading some text sizes and thus options to customize the user interface are a good feature. Finally, the application’s design could provide silent or vibrate modes, to allow users to use them without being noticed by others.

Battery-Powered: Mobile devices use batteries as their power source and need to be recharged. Their processor power, screen display, Wi-Fi or network connectivity increase the power usage. An application can be designed to be battery friendly and use less power. This can be accomplished by pre-fetching data while the screen is off when possible. An app could also allow users to choose how often its services will run in the background and notify them about the power demand of each choice.

Internet Connectivity: Mobile devices have the advantage of being able to tap into a cellular network to stay connected to the web if a WiFi connection is not available. Mobile data however can be more expensive than a standard broadband connection. That means that an application needs to be designed to handle inconsistent connectivity gracefully. But, their ability to execute some tasks offline should be taken into account when they are being designed.

Difficult Text Entry: Text entry is more difficult on mobile devices than on full-sized computers due to the size of the screen and because they feature an on-screen touch keyboard that's smaller than that of a standard computer. The smaller keyboard can make typing more difficult for users, especially those who have big fingers. These are the reasons users prefer to type short texts while they are using their phones or tablets. An application could provide other input sources, such as cameras, address books, calendars, auto-completion, or GPS.

4.3 Principles of Mobile Interface Design

App design is a combination of UI and UX design [28]. The goal is to build an app that is not confusing to use. Ideal mobile app design is good-looking, functional, and straightforward all at the same time. Some mobile interface design principles [29] (Figure 14) a developer needs to follow are presented below:

a. Intuitive Navigation: Users should be able to easily understand how to get around the app. This can be achieved by providing menus, showing the name of each page, using recognizable icons ("home" icon for the home screen, etc.), accessing any part of the app in a few clicks, etc.

b. Informative Alerts: Alert dialogs should inform the user whenever an error occurs or an operation is requested, for the application to retain its responsiveness.

c. Finger Friendly: Interactive elements should be easy to press. All the components of the application's screen must be easily clickable and have the proper size, color and be spaced out.

d. Minimal Commitment: Scrolling and typing should be avoided when possible. Reduce the need for typing by providing drop-down menus or autocomplete when possible and removing unnecessary fields.

e. Provide Feedback: Instant feedback should be provided for every interaction to inform the user know if the action is completed or not. The feedback could be tactile or visual and be provided by modal alerts, vibration, led notification or a combination of them.

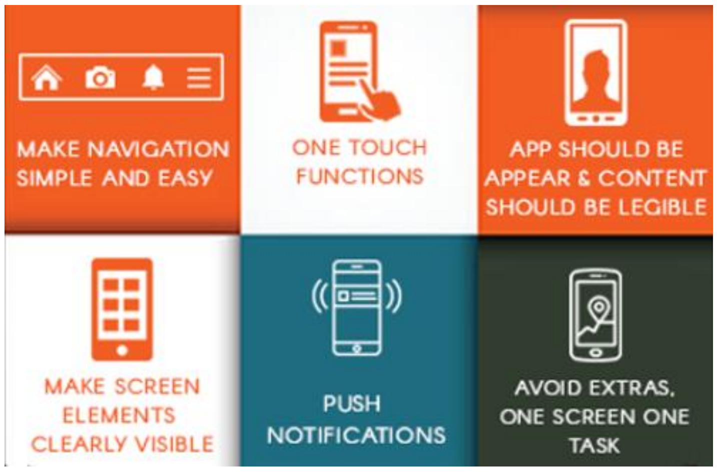

MAKE NAVIGATION SIMPLE AND EASY
ONE TOUCH FUNCTIONS
APP SHOULD BE APPEAR & CONTENT SHOULD BE LEGIBLE
MAKE SCREEN ELEMENTS CLEARLY VISIBLE
PUSH NOTIFICATIONS
AVOID EXTRAS, ONE SCREEN ONE TASK

Figure 14 Principles of Mobile App Design

4.4 System Requirements

Requirements analysis is a process that enables the success of a software project to be assessed. They can be divided into two main categories, Functional Requirements and Non-Functional Requirements.

4.4.1 Functional Requirements

Functional requirements (Table 1) are the basic functionalities the end-user expects the system to offer. They can be simply defined as something the system must do to operate properly.

Table 1 Functional Requirements

Functional Requirements	
ID	**Description**
FR01	Users must be able to register with an email and a password
FR02	Users must be able to log in with their email and password
FR03	Users must be able to logout at any time
FR04	The system must show an error message in case of error during login/registration
FR05	Users must be able to select a photo from their phone as a profile picture
FR06	Users must be able to select if the account is personal or professional
FR07	Users must be able to edit their profile information
FR08	Users must be able to select the category of the worker they want to search for
FR09	Users must be able to search for a worker in a specific area
FR10	The system must show on a map all the workers that match the search criteria
FR11	The system must provide all the necessary information about a professional
FR12	Users must be able to see the reviews each professional has received
FR13	Users must be able to leave a review of the worker they hired
FR14	Users must be able to view all past reviews they made
FR15	Users (professionals) must be able to view all the reviews they have received

4.4.2 Non-Functional Requirements

Non Functional requirements [30] (Table 2) are the quality constraints that the system must satisfy. They can be simply defined as the way the system works to perform well.

They are also called non-behavioral requirements and some of their most important classes are:

a. Usability
b. Security
c. Performance/Speed
d. Reliability
e. Maintainability
f. Modifiability/Extensibility
g. Scalability/Capacity

Table 2 Non-Functional Requirements

Non-Functional Requirements	
ID	**Description**
NFR01	The GUI must be easy to use
NFR02	All screens should have a similar style
NFR03	The system should display detailed notification/error messages to the user
NFR04	The system should require a password from each user
NFR05	The system must have a fast response time
NFR06	The system must be checked for any possible bugs in its operations
NFR07	The system will be able to run on all devices regardless of their OS.
NFR08	The system must be capable of being maintained/repaired
NFR09	The system should be easily changed to meet new requirements
NFR10	The system should be able to handle a considerable number of users

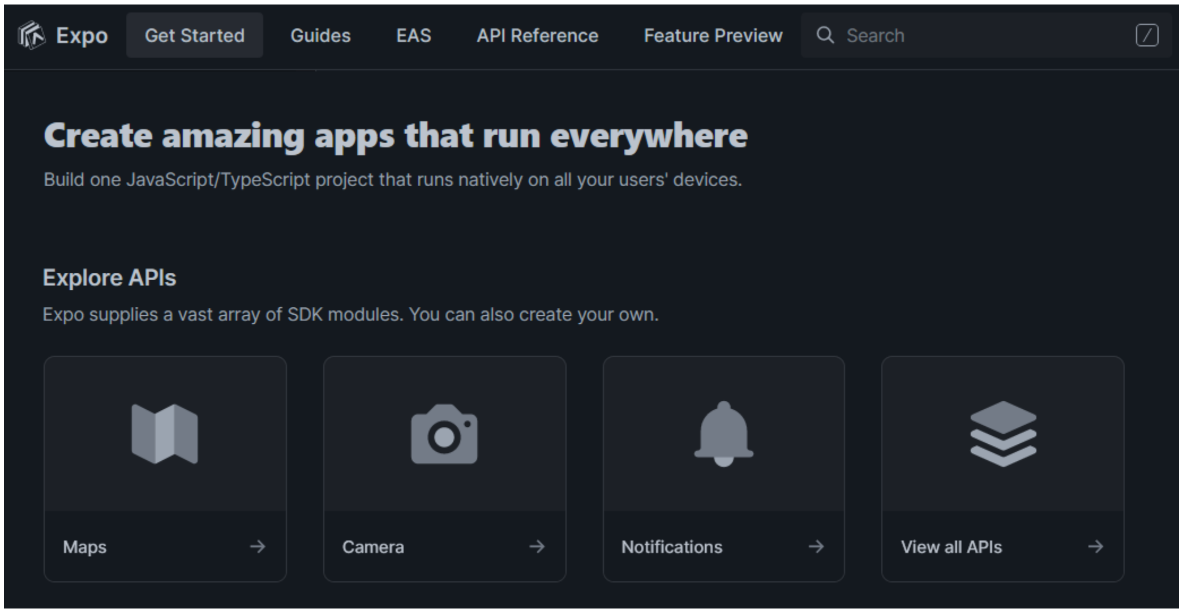
5 Application Development

5.1 Tools Used

Visual Studio Code [31] is the code editor selected for developing FixIt. The application's UI is built with the use of React Native [1] and Expo [32]. Firebase [2] is used for the backend development of the app.

5.1.1 Expo

Expo [32] is a toolchain designed for React Native [1] that helps developers create and test their applications. Expo SDK (Figure 15) provides a collection of ready solutions that use the phone's camera, GPS and more, it offers functions like MapView, ImagePicker, etc that simplify the development experience. Expo CLI (Figure 16) opens in a browser and allows checking the app's status, scanning a QR code to deploy the app in Expo Go (Figure 16), publishing on the Expo server and more. Expo Go can be downloaded from Google Play or Apple Store, allowing developers to open projects from a mobile device during development, which is helpful when testing as they can see the differences between code changes.



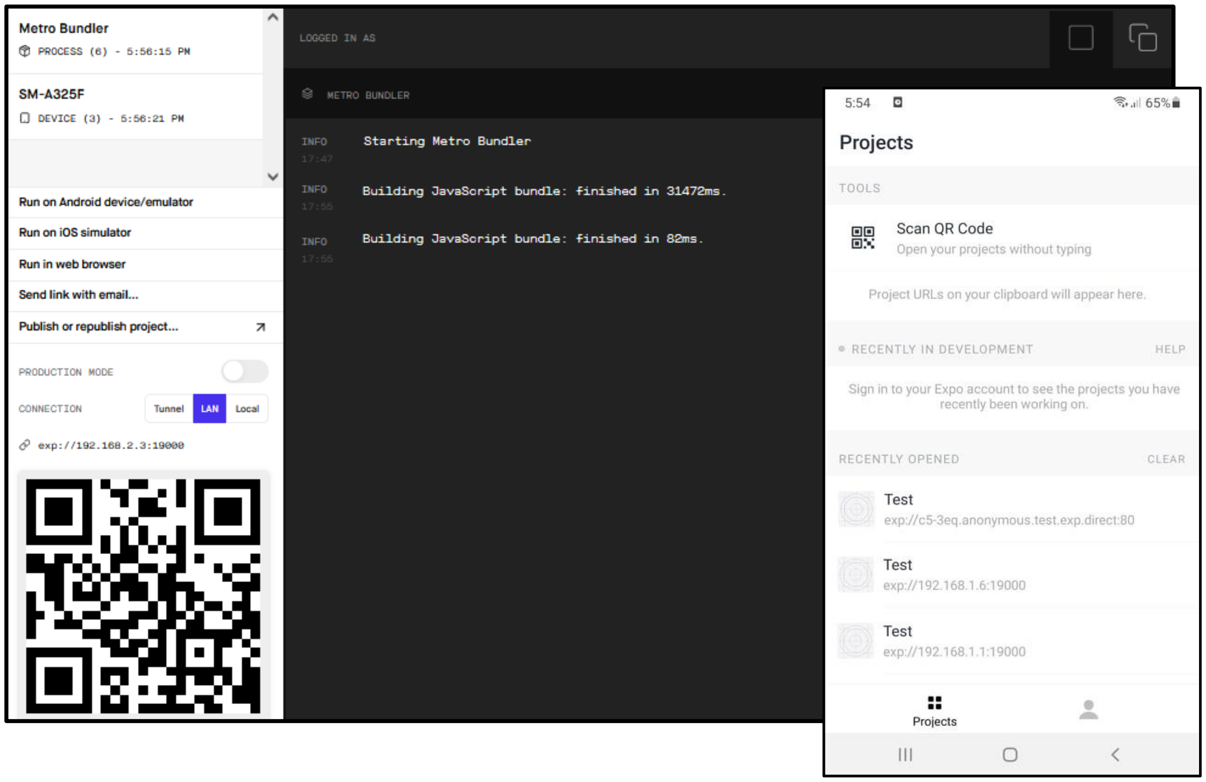
Create amazing apps that run everywhere

Build one JavaScript/TypeScript project that runs natively on all your users' devices.

Explore APIs

Expo supplies a vast array of SDK modules. You can also create your own.

Figure 15 Expo SDK



```
INFO    Starting Metro Bundler
17:47
INFO    Building JavaScript bundle: finished in 31472ms.
17:55
INFO    Building JavaScript bundle: finished in 82ms.
17:55
```

Open your projects without typing

Project URLs on your clipboard will appear here.

Sign in to your Expo account to see the projects you have recently been working on.

Figure 16 Expo CLI and Expo Go

The fastest way for a developer to share their Expo project is to publish it and open it in a development client app. This can be done by installing expo-updates in the project and running expo publish. This process gives the app a URL that can be shared with anybody who has the Expo Go app or the Development Build of the app for iOS or Android devices, and they can open the app immediately.

When the application is ready, a standalone app **(**.ipa and .aab) can be created for submission to Apple and Google's app stores. This is easy to do as it requires only one command because Expo will build the binary. You can also use internal distribution to share your app with ad-hoc or enterprise provisioning on iOS and an APK on Android.

The versions of Android and iOS that are supported by Expo apps are:

- Android 5+
- iOS 11+

5.1.2 React Native

React Native [1] is an open-source User Interface framework that allows developers to create native apps, using JavaScript, that run both on Android and iOS devices. React Native combines the best parts of native development with React, a best-in-class JavaScript library for building user interfaces.

React native allows cross-platform development. This saves developers a lot of time because one source code is needed, regardless of the operating system the end user's device has. Specifically, this ability increases efficiency, brings quick development results and reduces cost. Also, the development process speeds up considerably due to open-source libraries, including UI ones. Another advantage is that it makes updates easier, the developer has to maintain only one version of the software and all users get to update their apps at the same time. Another strong point of React Native is that it gives the developers the ability to link any plugin with a native or third-party module, like Google Maps, into their application. Finally, React Native has the biggest community among all cross-platforms. The greatest benefit of strong community support is the help available to developers.

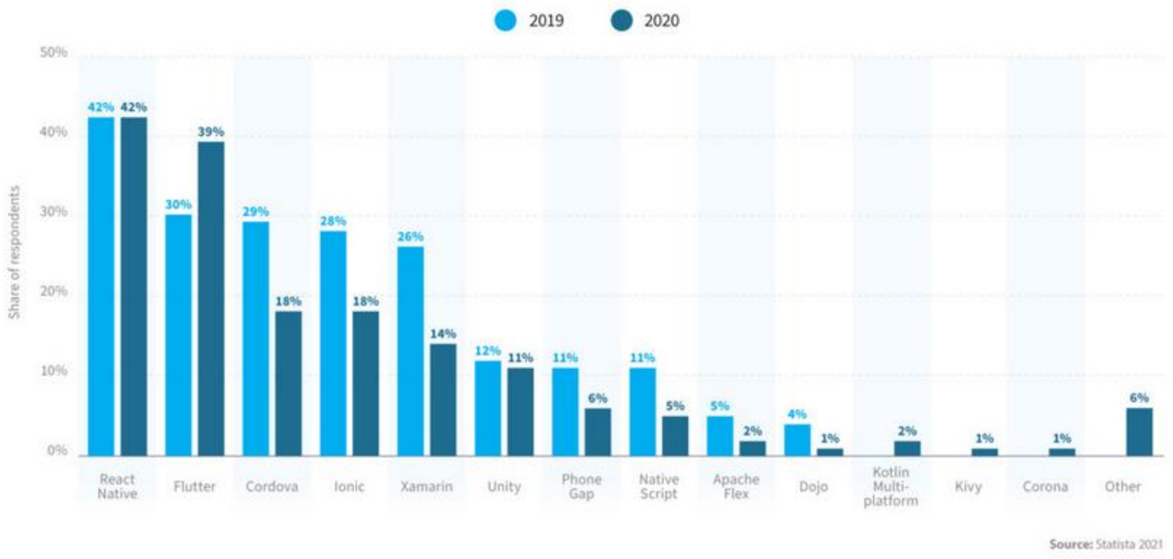


Figure 17 Cross-platform mobile frameworks used by software developers worldwide

According to statistics (Figure 17), React Native is a rising solution and the market leader in cross-platform mobile development.

The advantages mentioned above have played a major role in leading many companies (Figure 18) to select to develop their apps using React Native. They prefer to build their apps using React Native because their developers can write all of the code in JavaScript and share it across iOS and Android, and don’t need to pay for two separate teams of developers maintaining two different code bases.

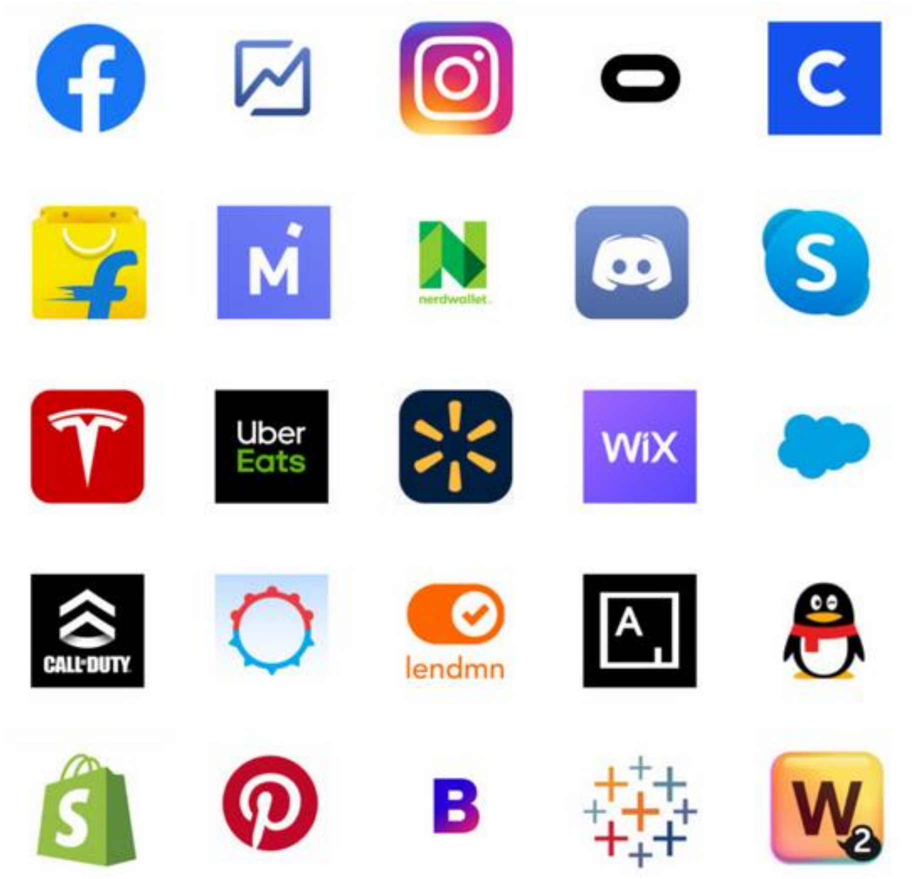


Figure 18 Apps using React Native

Due to React Native’s popularity, unique benefits and convenience, thousands of apps are using it as a solution. Some of them are Facebook, Instagram, Pinterest, Skype, Bloomberg, Uber Eats, etc. All these brands and top-notch applications use the React Native framework to gain a competitive advantage in the marketplace.

5.1.3 Firebase

Firebase [2] (Figure 19) is a Backend-as-a-Service (Baas) launched by James Tamplin and Andrew Lee in 2011. It is a toolset that was acquired by Google in 2014 and expanded to offer more services. It provides developers with services like authentication, databases, file storage, push messaging, etc. It is also used to develop features for applications to enhance the user experience and user engagement. Firebase stores data in JSON-like documents and is categorized as a NoSQL database program. As it is built on Google's infrastructure, the services are hosted in the cloud meaning that they are fully maintained and operated by Google. Being stored in the cloud also means that the data can be synced across all your users' devices or shared among multiple users. Also, as it is a real-time database, the data remain synchronized across all of the app's users because changes are automatically fetched from the database as they happen. Finally, some developers also employ it to increase revenues through the Admob Firebase feature.

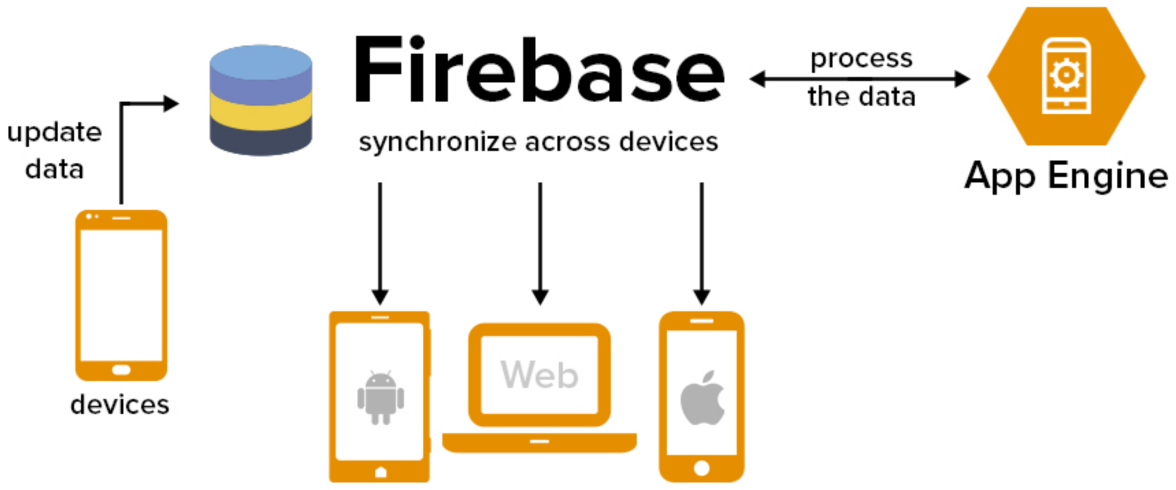


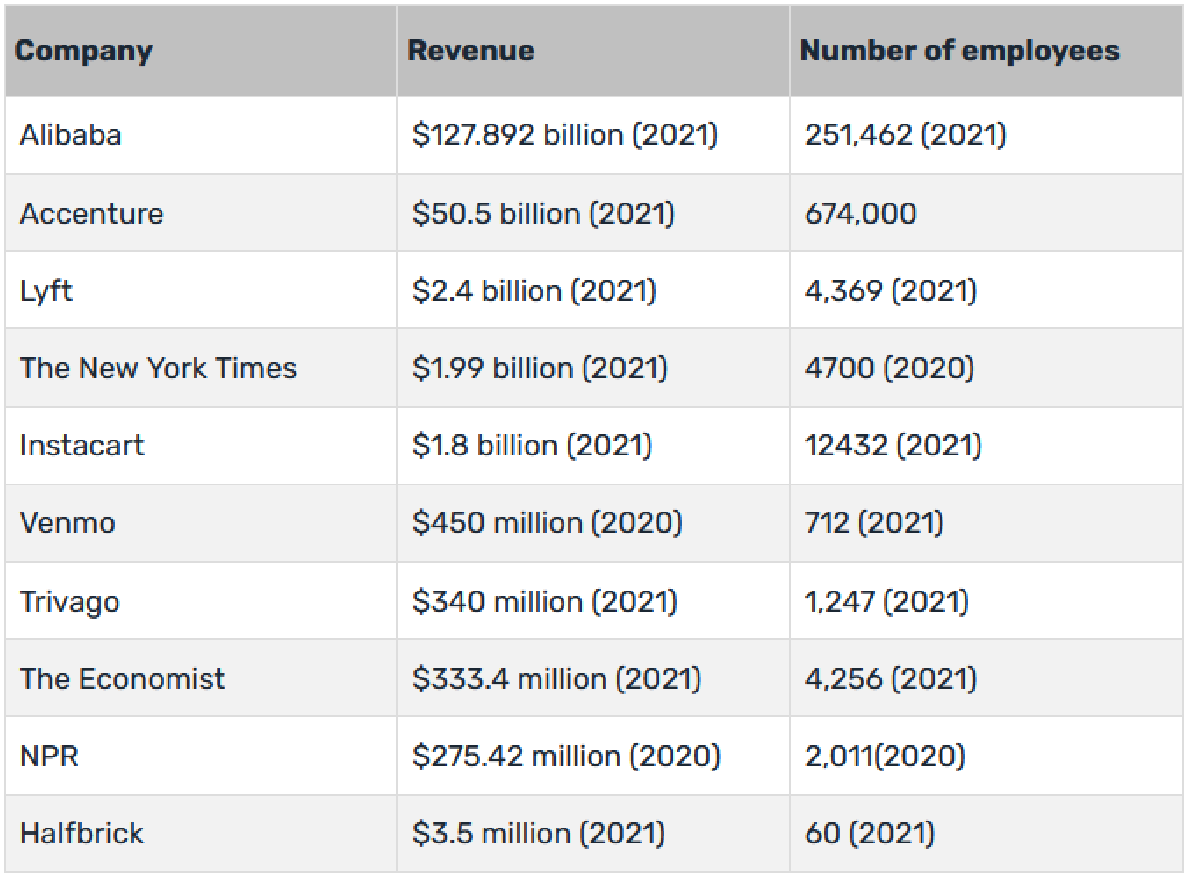
Figure 19 Firebase

Google Firebase is one of the most popular frameworks employed by professional developers and companies to build high-quality applications. Many companies and developers rely on it because of its seamless database management, cloud storage, and testing services.

Firebase offers seamless integration on Android, iOS, Unity, and the web. It also offers well-detailed documentation and numerous software development kits to ship applications easily. This is why it is utilized in many of the mobile apps we use today. Some notable applications built with Firebase are Twitch.tv, 9gag, Duolingo, Trustpilot, and Wattpad.

A wide variety of tools, tech products, and services is offered by Firebase to build real-time collaborative applications. In addition to the examples of products that were mentioned above and are built with Google Firebase, some of the largest companies (Table 3) also use Firebase. According to Stackshare [33], around 2,600 companies use the Firebase platform in their tech stack.

Table 3 Largest companies that use Firebase



Company	Revenue	Number of employees
Alibaba	$127.892 billion (2021)	251,462 (2021)
Accenture	$50.5 billion (2021)	674,000
Lyft	$2.4 billion (2021)	4,369 (2021)
The New York Times	$1.99 billion (2021)	4700 (2020)
Instacart	$1.8 billion (2021)	12432 (2021)
Venmo	$450 million (2020)	712 (2021)
Trivago	$340 million (2021)	1,247 (2021)
The Economist	$333.4 million (2021)	4,256 (2021)
NPR	$275.42 million (2020)	2,011(2020)
Halfbrick	$3.5 million (2021)	60 (2021)

5.2 Implementation

This section starts with the database structure followed by the development of the application. FixIt's mobile app consists of six screens: Login Screen, Home Screen, Reviews Screen, Search Screen, Notifications Screen, Ratings Screen and Profile Screen. In the following paragraphs, the implementation techniques for each screen and the code written are presented.

5.2.1 Database Structure

The application's data are stored in three collections in Firebase's Firestore Database. The collections are: users (Table 4), transactions (Table 5) and reviews (Table 6). Every collection consists of documents each one of them storing info of a user, transaction, or review correspondingly.

Table 4 users collection

users		
Attributes	Type	Description
uid	STRING	Auto-generated unique id of the user
address	STRING	Address of the user
Category*	STRING	Type of the account (customer / professional)
email	STRING	Unique email of the user
expoPushToken	STRING	Token that identifies the user's device for push notifications
firstName	STRING	First name of the user
lastName	STRING	Last name of the user
location	GEOPOINT	Coordinates of the user's address
phone	STRING	Phone number of the user
profileImage	STRING	User's profile image URI
reviews**	NUMBER	Total number of received reviews
stars**	NUMBER	Total number of received stars

* If the account is professional, a specific profession is selected.

** Only professional accounts have these attributes.

Table 5 transactions collection

transactions		
Attributes	Type	Description
uid	STRING	Auto-generated unique id of the transaction
customer	REFERENCE	The customer involved in the transaction
date	TIMESTAMP	Date of the last modification of the transaction
status***	STRING	Status of the transaction
worker	REFERENCE	The professional involved in the transaction

***Status can be “pending”, “completed” or “reviewed”.

Table 6 reviews collection

reviews		
Attributes	Type	Description
uid	STRING	Auto-generated unique id of the transaction
customer	REFERENCE	The customer that makes the review
date	TIMESTAMP	Date of the review
review	STRING	Review
stars	NUMBER	Stars
worker	REFERENCE	The professional that is reviewed

The “customer” and “worker” attributes of the transactions and reviews collections are of reference datatype (Figure 20). References can point to any document in any other collection.

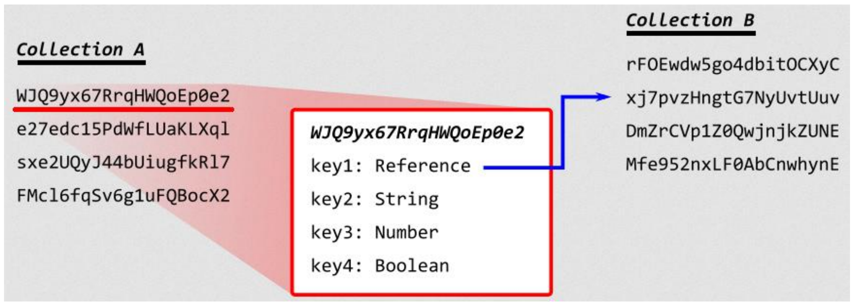
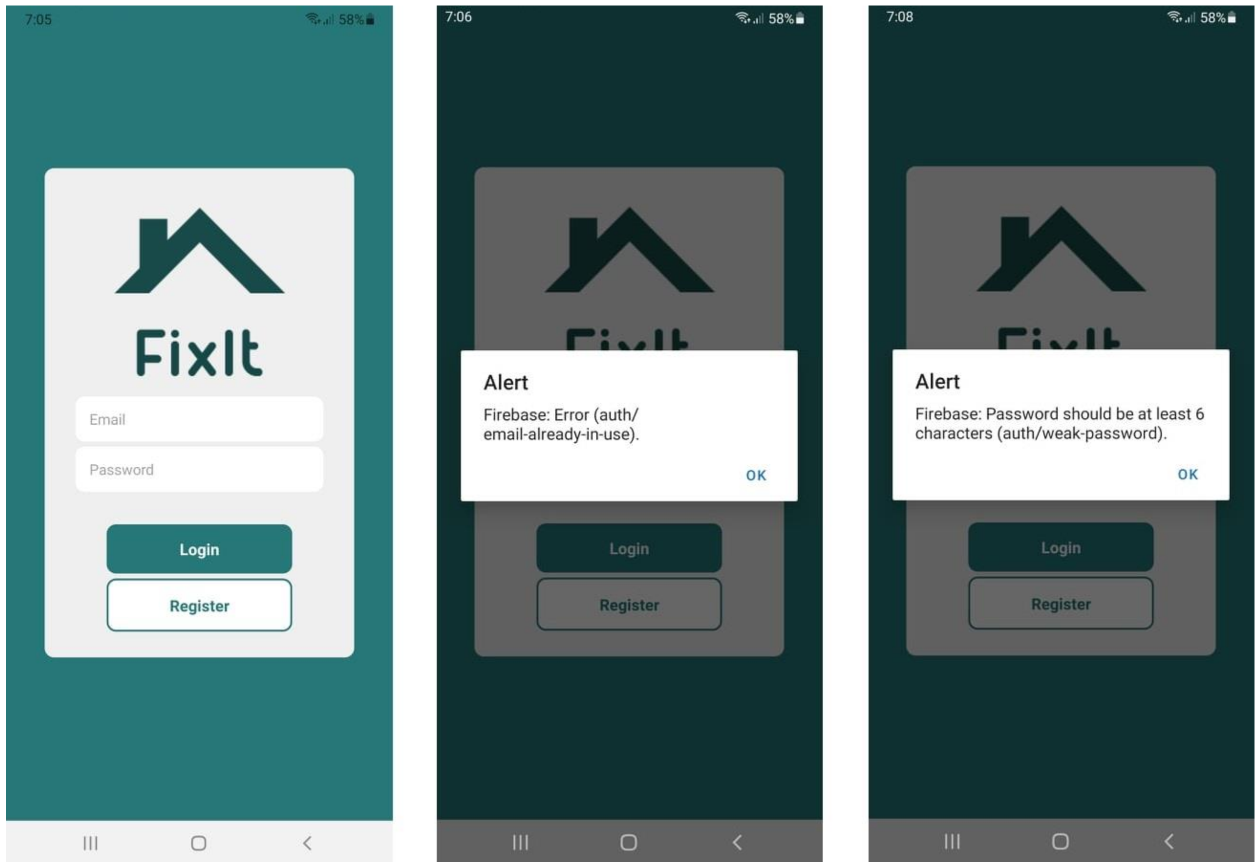


Figure 20 Reference datatype

5.2.2 Login Screen

The first screen the user sees once he downloads the app is the Login Screen (Figure 21). It has the FixIt logo, an email input field and a password input field, a login button and a register button. If it is the first time the user is using the app he has to create a new account. This process requires providing an email and a password. If the email is used by an existing profile or if the password is too weak (must be at least 6 characters long), an error window pops up to inform the user.



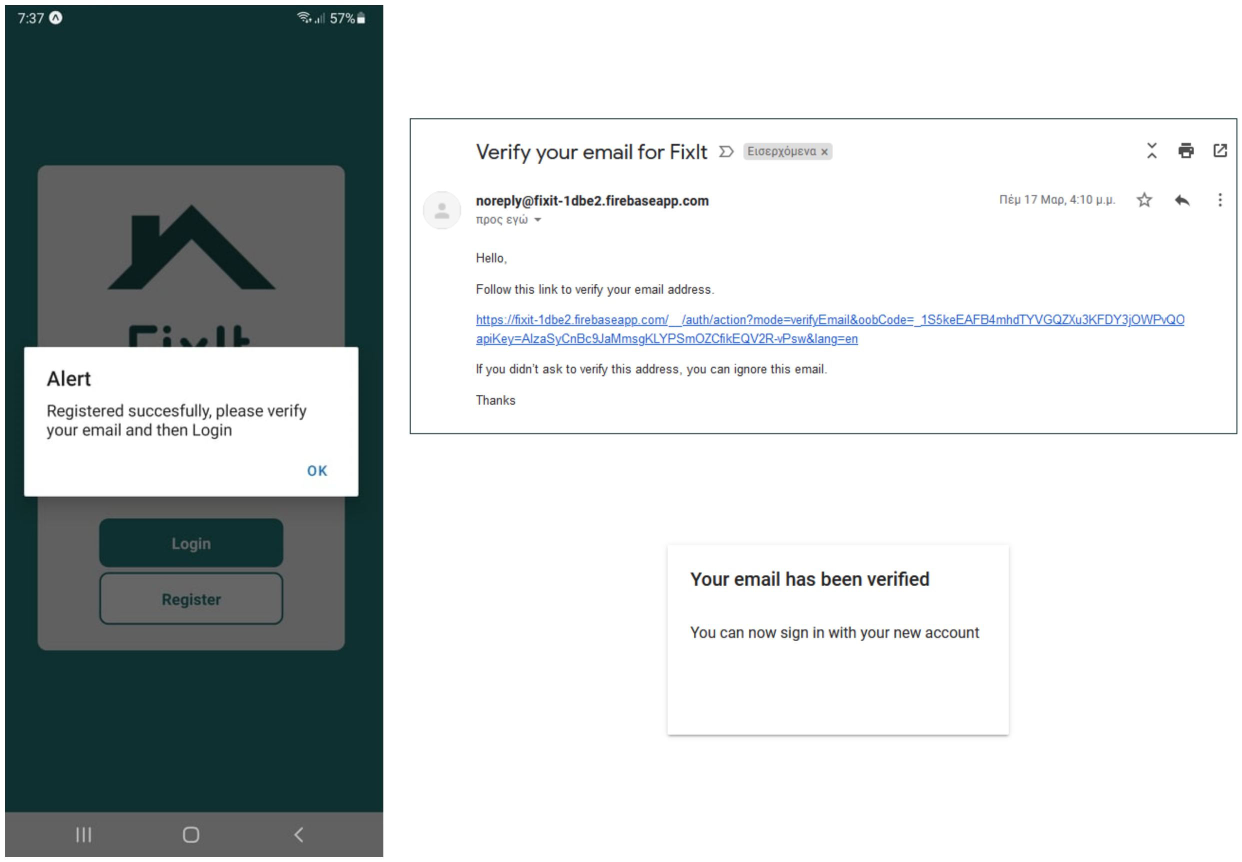
Firebase: Error (auth/email-already-in-use).

Firebase: Password should be at least 6 characters (auth/weak-password).

Figure 21 Login screen layout, errors during the registration process

Once the user provides a correct email and password, he can then tap the register button to proceed. A pop-up window informs him that to finalize the registration, email verification is required (Figure 22). This step ensures that every account is linked with a valid email address.

The user receives an automatically generated email that contains a link to verify his email address. By clicking it, the email address is verified and the sign-in process is completed.



Alert

Registered succesfully, please verify your email and then Login

OK

Verify your email for FixIt Εισερχόμενα ×

noreply@fixit-1dbe2.firebaseapp.com
προς εγώ

Πέμ 17 Μαρ, 4:10 μ.μ.

Hello,

Follow this link to verify your email address.

https://fixit-1dbe2.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=_1S5keEAFB4mhdTYVGQZXu3KFDY3jOWPvQO
apiKey=AIzaSyCnBc9JaMmsgKLYPSmOZCfikEQV2R-vPsw&lang=en

If you didn't ask to verify this address, you can ignore this email.

Thanks

Your email has been verified

You can now sign in with your new account

Figure 22 Registration alert, verification email, successful verification

The registration process is implemented by Firebase Authentication [34]. The sign-in method selected is Email/Password. Authentication of password-based accounts [35] is managed with the help of "createUserWithEmailAndPassword" function for the sign-up process and "signInWithEmailAndPassword" function for the login process. All registered users receive upon registration, an automatically generated User UID which is unique. Every time a new account is created, a new document is instantiated by the "setDoc" function [36] for the new user to be stored (as a document) in the "users" collection in Cloud Firestore [37] with his UID as key and his email. Each document will get updated, once the user fills in his profile at the Profile screen, with the rest of his personal info.

5.2.3 Home Screen

After logging in, the user is redirected to the Home screen (Figure 23). A tab menu at the bottom of the screen allows the user to navigate between screens, from the Home screen to Reviews Screen, Search Screen, Notifications Screen, Ratings Screen or Profile Screen. If the user wants to log out he can tap the logout icon at the right top corner and get redirected to the Login screen.

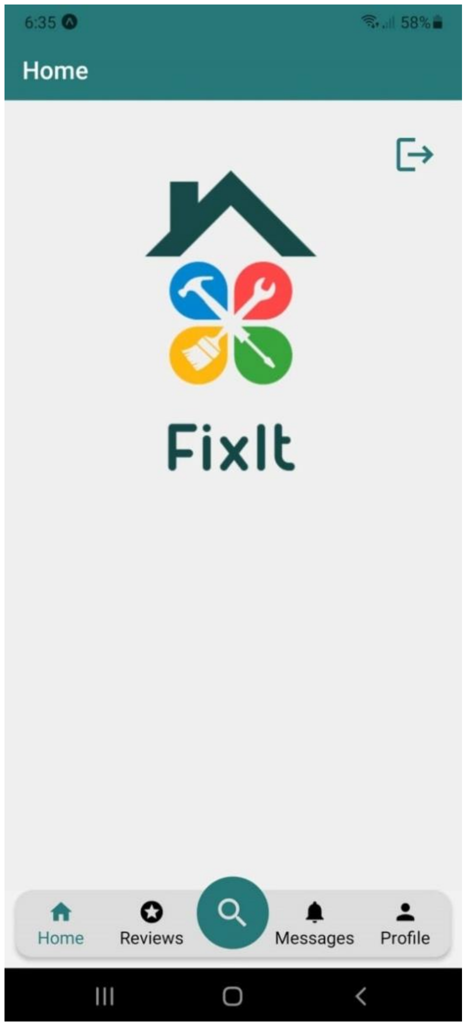
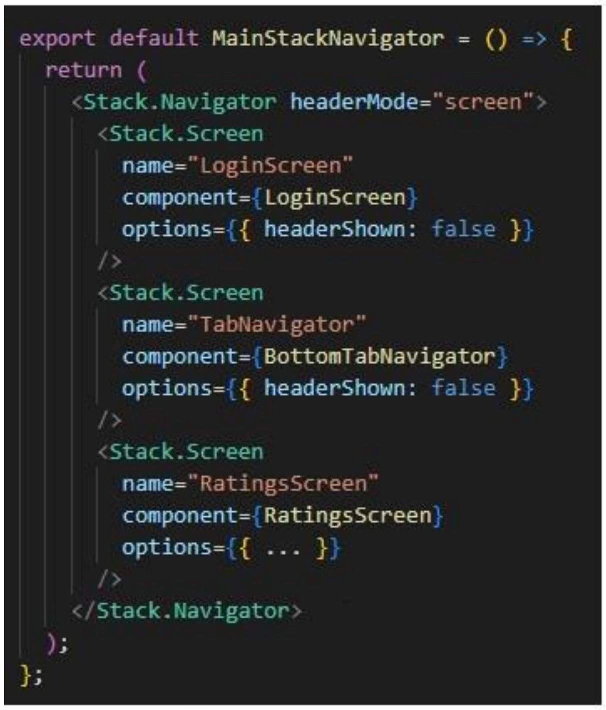
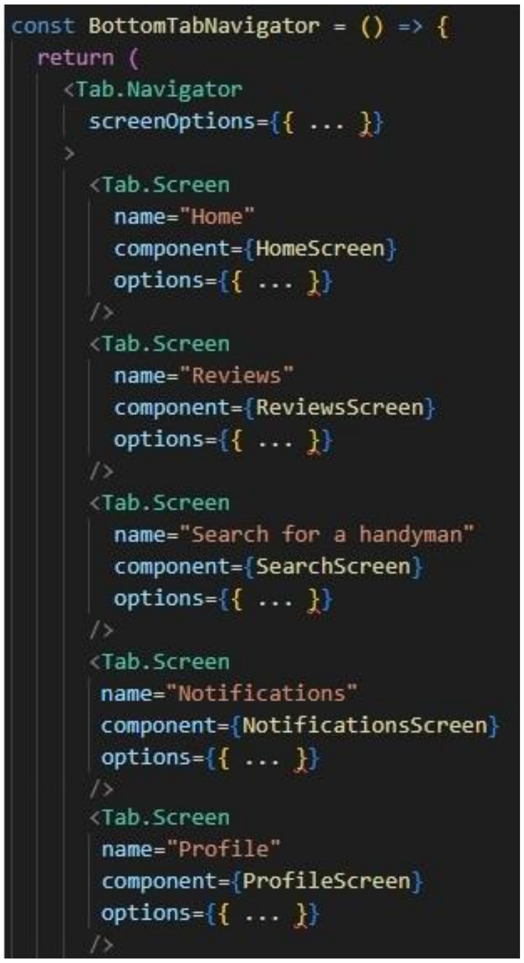


Figure 23 Home screen layout

A stack navigator [38] (Figure 24) places screens on a stack and is used to transition between the Login screen, the TabNavigator (rest of the screens) and the Notifications Screen. The tab menu is made with Bottom Tabs Navigator [39] (Figure 24) which allows the user to navigate easily between screens. The icons for each screen at the tab menu change colors so the user knows on which screen he is. Also, with the help of the "replace" function, the user can log out and be redirected to the Login screen.



```
export default MainStackNavigator = () => {
  return (
    <Stack.Navigator headerMode="screen">
      <Stack.Screen
        name="LoginScreen"
        component={LoginScreen}
        options={{ headerShown: false }}
      />
      <Stack.Screen
        name="TabNavigator"
        component={BottomTabNavigator}
        options={{ headerShown: false }}
      />
      <Stack.Screen
        name="RatingsScreen"
        component={RatingsScreen}
        options={{ ... }}
      />
    </Stack.Navigator>
  );
};
```

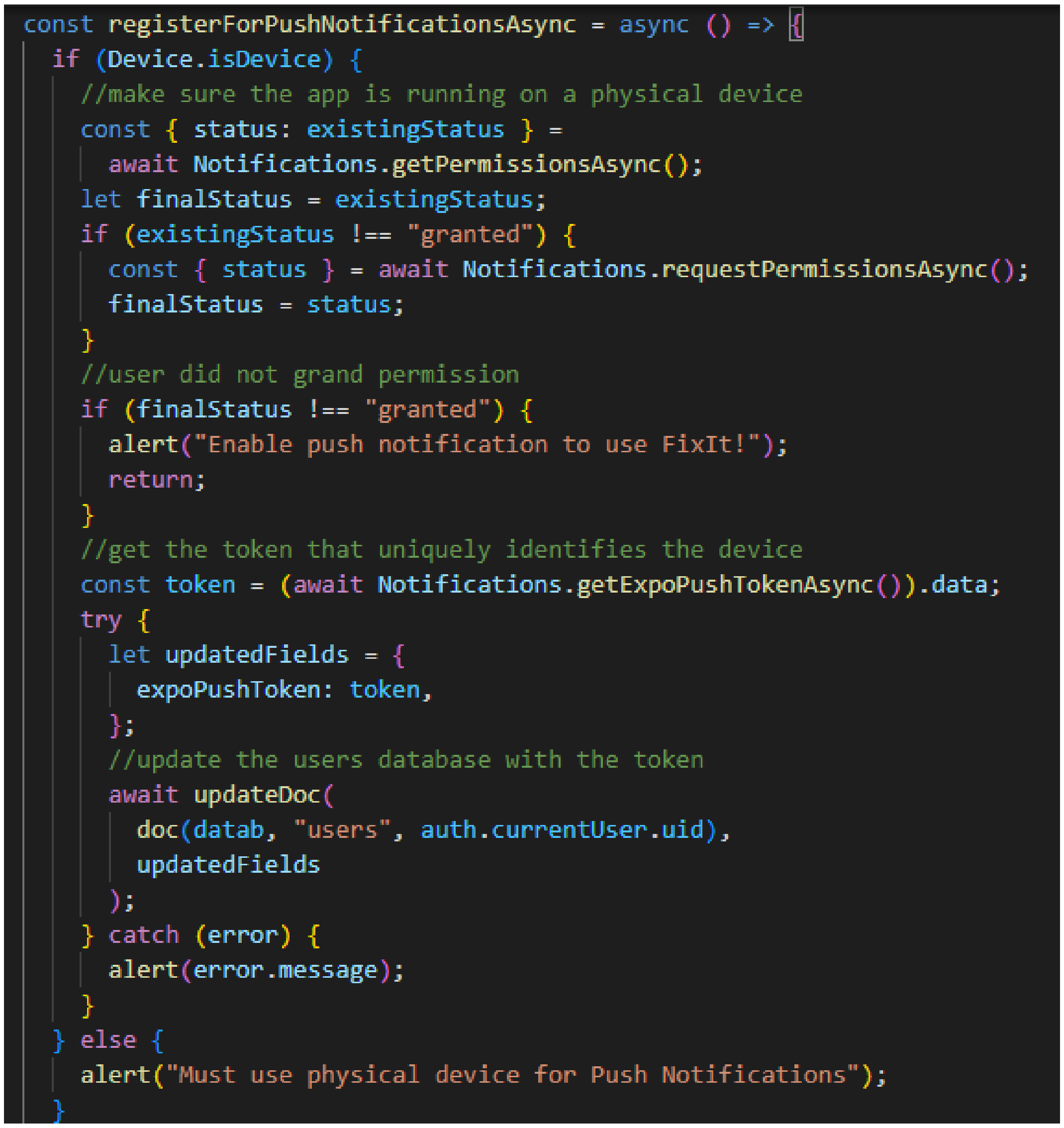


```
const BottomTabNavigator = () => {
  return (
    <Tab.Navigator
      screenOptions={{ ... }}
    >
      <Tab.Screen
        name="Home"
        component={HomeScreen}
        options={{ ... }}
      />
      <Tab.Screen
        name="Reviews"
        component={ReviewsScreen}
        options={{ ... }}
      />
      <Tab.Screen
        name="Search for a handyman"
        component={SearchScreen}
        options={{ ... }}
      />
      <Tab.Screen
       name="Notifications"
       component={NotificationsScreen}
       options={{ ... }}
      />
      <Tab.Screen
       name="Profile"
       component={ProfileScreen}
       options={{ ... }}
      />
```

Figure 24 Navigation code snapshot

Push notifications are pop-up messages, send by a server or another client via request, that appear on a user's device and help to smooth the user experience. FixIt uses Expo's notifications [40] to implement them. There are three main steps to set up push notifications with Expo: a) get the device token, b) make an HTTP request asking for a notification to be sent, and c) handle receival/interaction of that notification in the app.

Once the Home Screen is rendered the client-side should get ready for push notifications (Figure 25), the two main things needed are the user's permission to send them push notifications and the user's ExpoPushToken. The following function implements these.



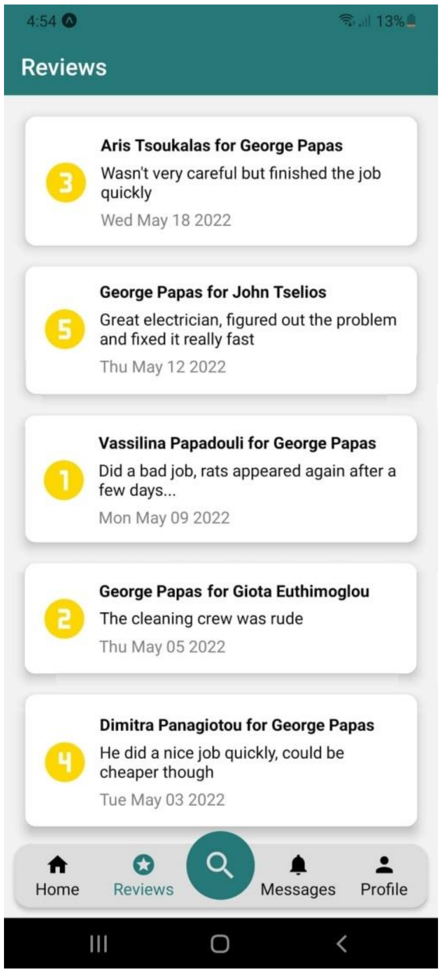
```
const registerForPushNotificationsAsync = async () => {
  if (Device.isDevice) {
    //make sure the app is running on a physical device
    const { status: existingStatus } =
      await Notifications.getPermissionsAsync();
    let finalStatus = existingStatus;
    if (existingStatus !== "granted") {
      const { status } = await Notifications.requestPermissionsAsync();
      finalStatus = status;
    }
    //user did not grand permission
    if (finalStatus !== "granted") {
      alert("Enable push notification to use FixIt!");
      return;
    }
    //get the token that uniquely identifies the device
    const token = (await Notifications.getExpoPushTokenAsync()).data;
    try {
      let updatedFields = {
        expoPushToken: token,
      };
      //update the users database with the token
      await updateDoc(
        doc(datab, "users", auth.currentUser.uid),
        updatedFields
      );
    } catch (error) {
      alert(error.message);
    }
  } else {
    alert("Must use physical device for Push Notifications");
  }
```

Figure 25 Code snapshot to register the device for push notifications

After the token is acquired, the user’s document in the “users” database gets updated with it, the “updateDoc” [36] method is used.

5.2.4 Reviews Screen

The Reviews screen (Figure 26) is accessed by selecting the reviews tab in the menu. On this screen, the users with customer accounts can see all the reviews they have left in the past, the users with professional accounts can see both the reviews they have received from their customers and the reviews they have made. Each review contains the rating, the client's and worker's name, a review and the date it was made.



3 Aris Tsoukalas for George Papas
Wasn't very careful but finished the job quickly
Wed May 18 2022

5 George Papas for John Tselios
Great electrician, figured out the problem and fixed it really fast
Thu May 12 2022

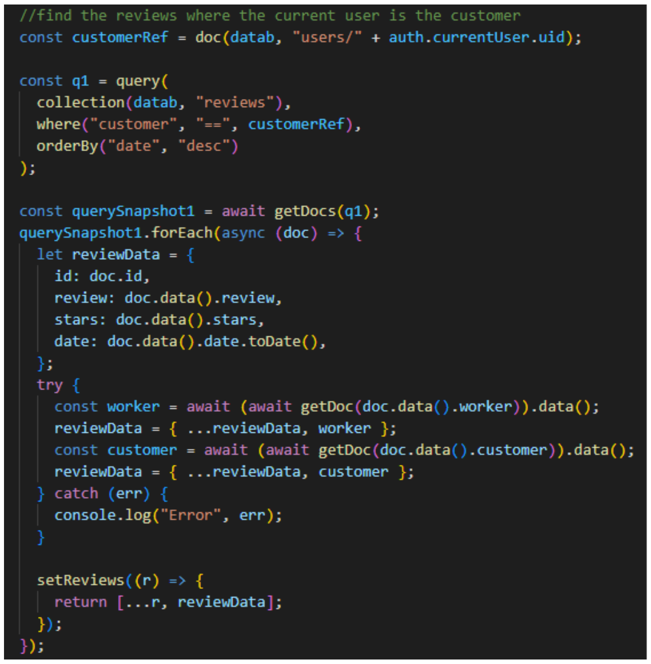
1 Vassilina Papadouli for George Papas
Did a bad job, rats appeared again after a few days...
Mon May 09 2022

2 George Papas for Giota Euthimoglou
The cleaning crew was rude
Thu May 05 2022

4 Dimitra Panagiotou for George Papas
He did a nice job quickly, could be cheaper though
Tue May 03 2022

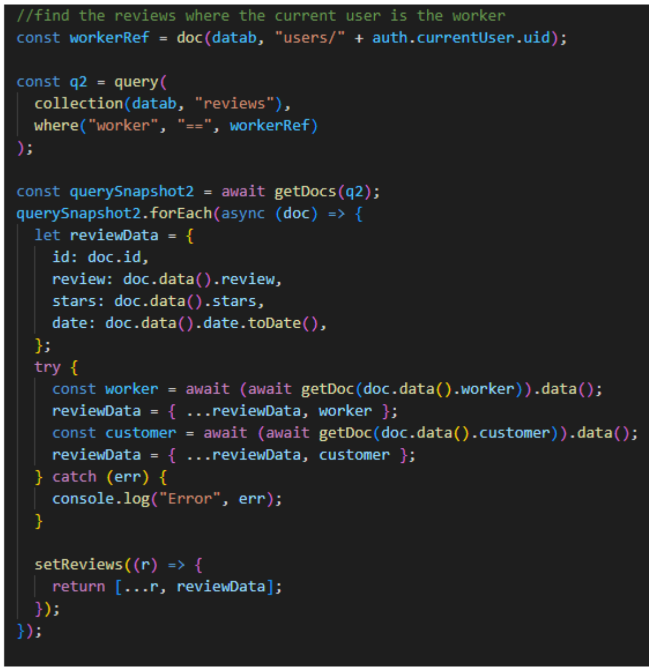
Figure 26 Reviews screen layout

To display the reviews on a Flatlist, Firestore's query [41] functionality is used to retrieve all the documents from the "reviews" collection where "worker" or "customer" is equal to the current user. However, Cloud Firestore does not support Logical OR queries, so the results of two different queries need to be merged.



```
//find the reviews where the current user is the customer
const customerRef = doc(datab, "users/" + auth.currentUser.uid);

const q1 = query(
  collection(datab, "reviews"),
  where("customer", "==", customerRef),
  orderBy("date", "desc")
);

const querySnapshot1 = await getDocs(q1);
querySnapshot1.forEach(async (doc) => {
  let reviewData = {
    id: doc.id,
    review: doc.data().review,
    stars: doc.data().stars,
    date: doc.data().date.toDate(),
  };
  try {
    const worker = await (await getDoc(doc.data().worker)).data();
    reviewData = { ...reviewData, worker };
    const customer = await (await getDoc(doc.data().customer)).data();
    reviewData = { ...reviewData, customer };
  } catch (err) {
    console.log("Error", err);
  }

  setReviews((r) => {
    return [...r, reviewData];
  });
});
```

Figure 27 Query where "customer" is equal to the current user



```
//find the reviews where the current user is the worker
const workerRef = doc(datab, "users/" + auth.currentUser.uid);

const q2 = query(
  collection(datab, "reviews"),
  where("worker", "==", workerRef)
);

const querySnapshot2 = await getDocs(q2);
querySnapshot2.forEach(async (doc) => {
  let reviewData = {
    id: doc.id,
    review: doc.data().review,
    stars: doc.data().stars,
    date: doc.data().date.toDate(),
  };
  try {
    const worker = await (await getDoc(doc.data().worker)).data();
    reviewData = { ...reviewData, worker };
    const customer = await (await getDoc(doc.data().customer)).data();
    reviewData = { ...reviewData, customer };
  } catch (err) {
    console.log("Error", err);
  }

  setReviews((r) => {
    return [...r, reviewData];
  });
});
```

Figure 28 Query where "worker" is equal to the current user

The first query (Figure 27) retrieves the documents where "customer" is equal to the current user and the second (Figure 28) the ones where "worker" is equal to the current user with the "getDocs" [42] function. The data from the two snapshots are stored in an array.

5.2.5 Search Screen

The Search screen (Figure 29) is accessed by selecting the search tab in the menu. The user can search for the home service he needs by filling in the input fields (category, location). First, the user can choose the type of professional he wants to employ from the drop-down menu. Then, the area in which the user is interested to look for professionals/businesses can be selected with the help of Google Places Autocomplete. By pressing the search button, the results that match the search criteria will appear on the map.

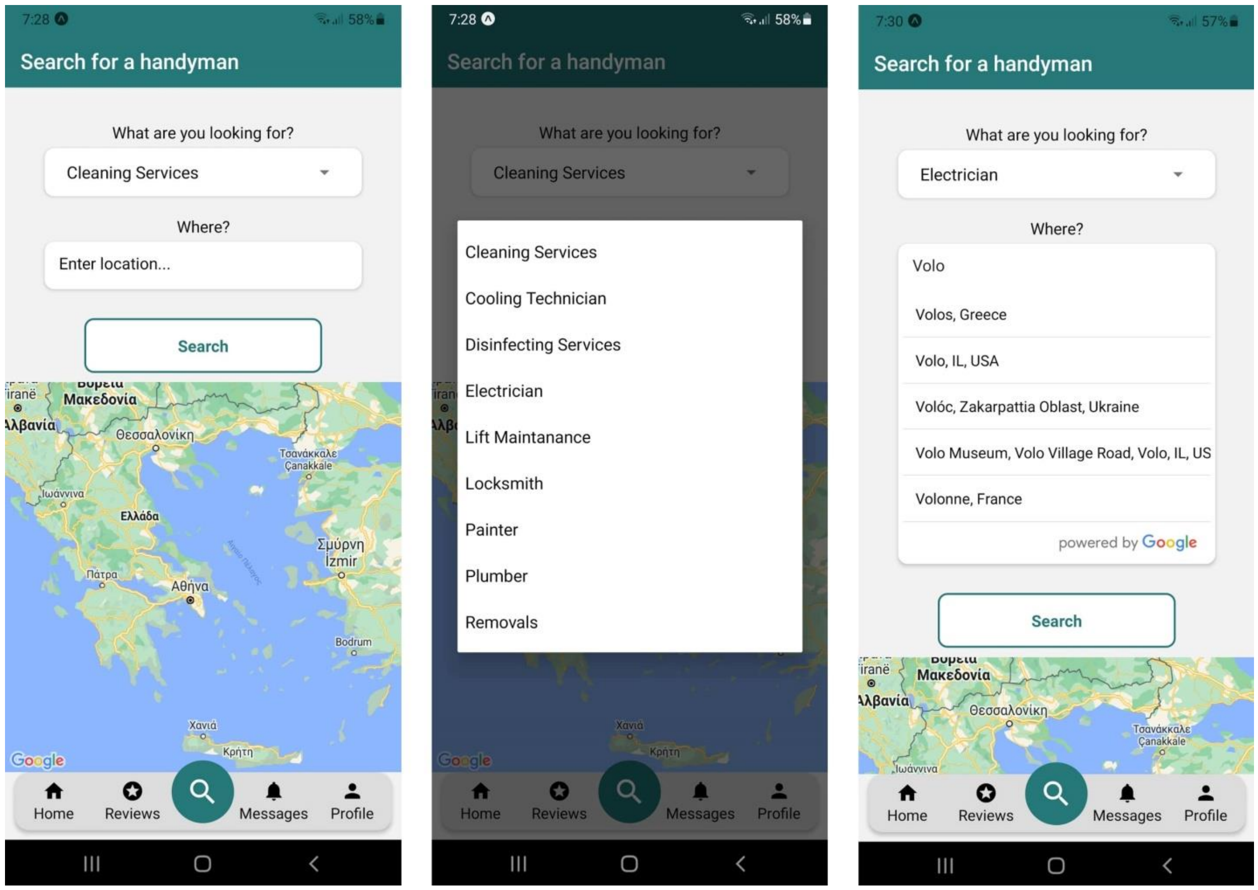
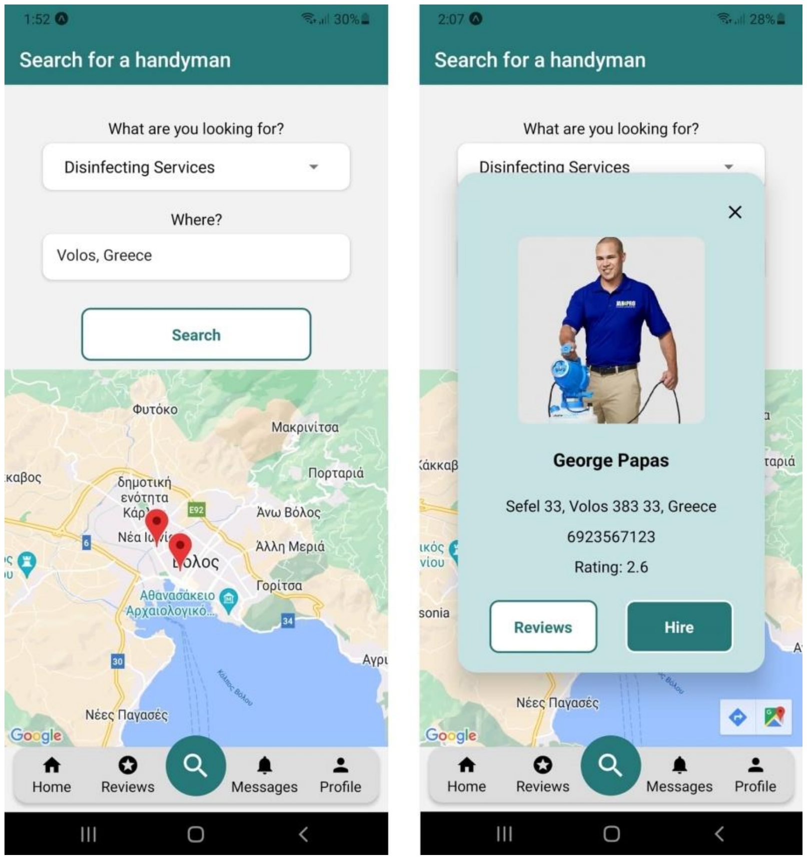


Figure 29 Search screen layout, category drop-down, location auto-complete

The map will zoom in on the area selected by the user and the professionals that match the search criteria will appear as markers (Figure 30). By pressing a marker, the name of the corresponding professional will appear as a tag. By tapping on the name, a pop-up window with the professional's info will appear. The user can then select to see the reviews of this professional or hire them (Figure 30).



Disinfecting Services

Volos, Greece

Disinfecting Services

George Papas

Sefel 33, Volos 383 33, Greece

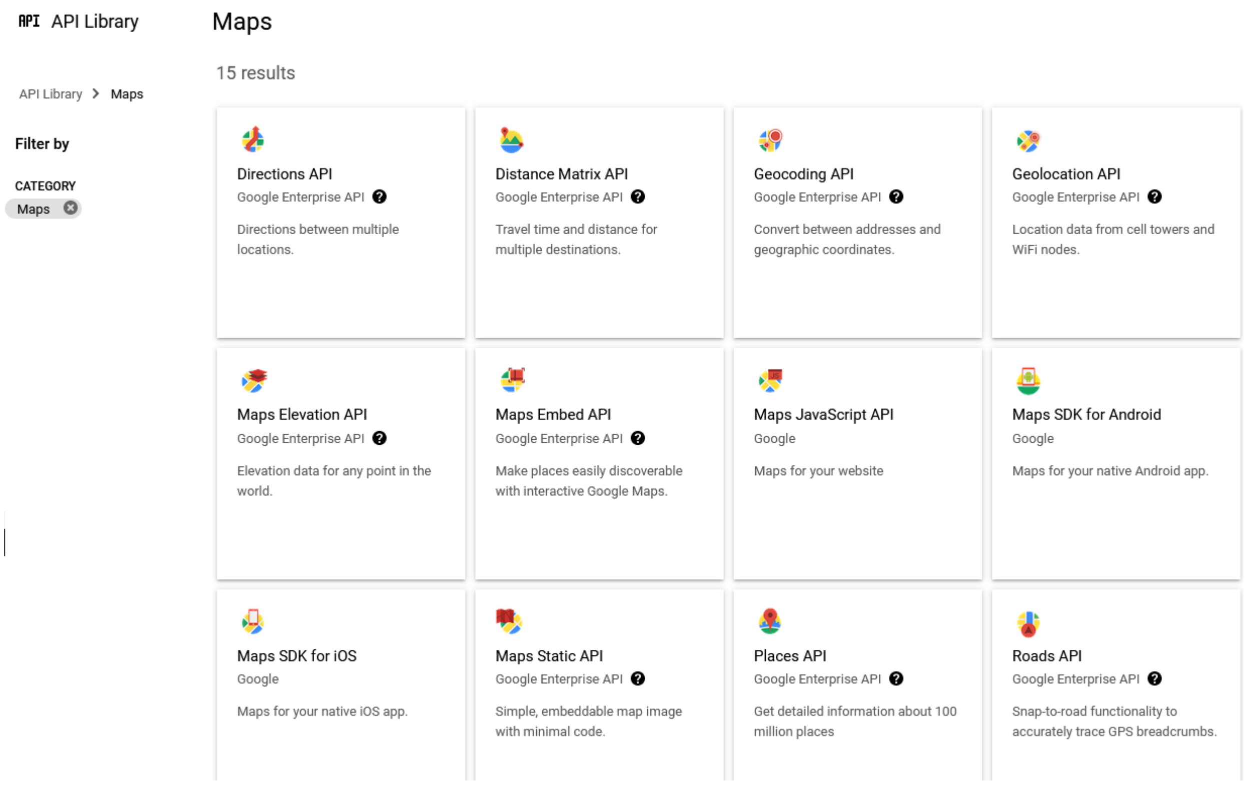
6923567123

Rating: 2.6

Figure 30 Search results, professional's info with reviews/hire options

For the autocomplete component of the location input field, the "react-native-google-places-autocomplete" [43] is used. A "Google Places API key" is required and "Google Places API" should be enabled (Figure 32). The key and the API are provided by Google Cloud Platform [44], a suite of cloud computing services. The Google API client libraries are used to access APIs for products such as Maps (Figure 31). Must be noted that any Google Cloud resources that are allocated and used must belong to a project.

The map component uses "react-native-maps" [45]. Inside the "MapView" component [46], the "initialRegion" prop sets the region that is to be displayed on the map when the component mounts. The latitude, longitude, and delta values of the "initialRegion" are chosen so Greece is displayed when the Search screen mounts.


API Library

Maps

15 results

API Library > Maps

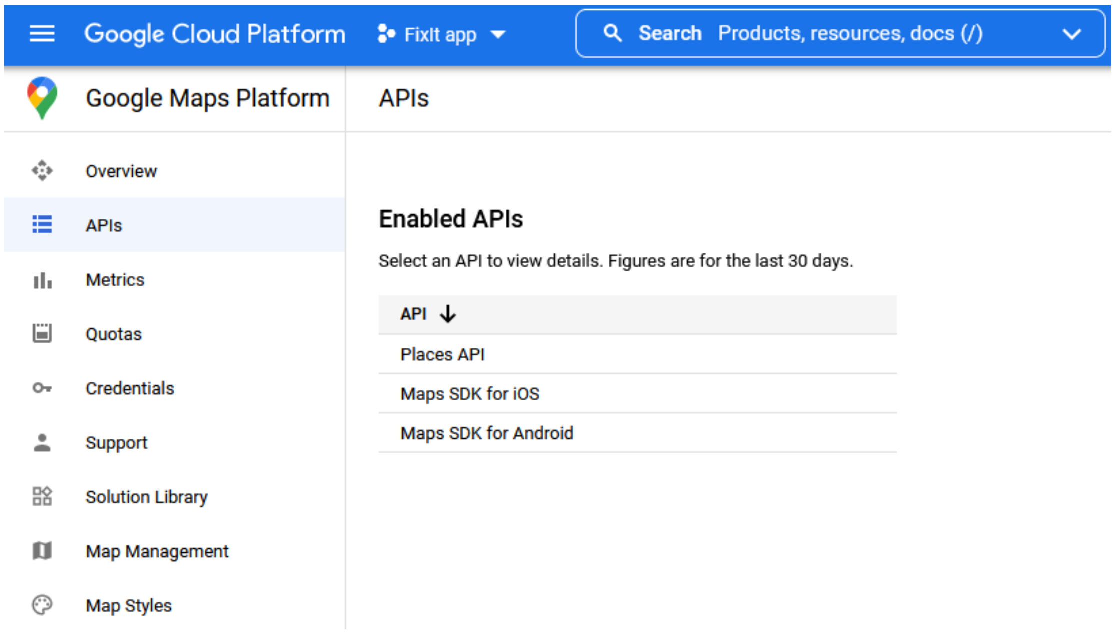
Filter by

CATEGORY

Maps

Directions API Google Enterprise API Directions between multiple locations.	Distance Matrix API Google Enterprise API Travel time and distance for multiple destinations.	Geocoding API Google Enterprise API Convert between addresses and geographic coordinates.	Geolocation API Google Enterprise API Location data from cell towers and WiFi nodes.
Maps Elevation API Google Enterprise API Elevation data for any point in the world.	Maps Embed API Google Enterprise API Make places easily discoverable with interactive Google Maps.	Maps JavaScript API Google Maps for your website	Maps SDK for Android Google Maps for your native Android app.
Maps SDK for iOS Google Maps for your native iOS app.	Maps Static API Google Enterprise API Simple, embeddable map image with minimal code.	Places API Google Enterprise API Get detailed information about 100 million places	Roads API Google Enterprise API Snap-to-road functionality to accurately trace GPS breadcrumbs.

Figure 31 Google Maps APIs



APIs

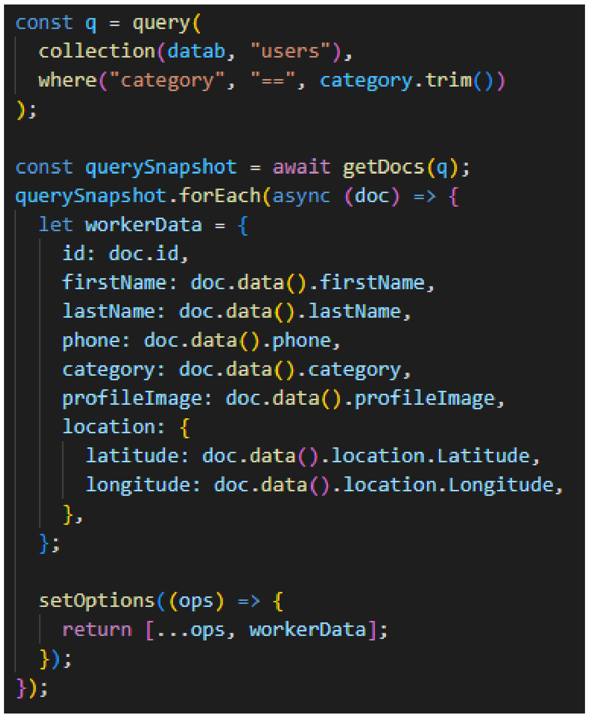
Enabled APIs

Select an API to view details. Figures are for the last 30 days.

API ↓
Places API
Maps SDK for iOS
Maps SDK for Android

Figure 32 Enabled APIs

To implement the search, Firestore’s query [41] functionality is needed to specify which documents need to be retrieved from a collection. The query (Figure 33) returns all the documents from the “users” collection where “category” is equal to the category selected from the drop-down menu (built with Expo’s Picker). After creating a query object, the “getDocs” [42] function is used to retrieve the results. The data from the snapshots are stored in an array.



```
const q = query(
  collection(datab, "users"),
  where("category", "==", category.trim())
);

const querySnapshot = await getDocs(q);
querySnapshot.forEach(async (doc) => {
  let workerData = {
    id: doc.id,
    firstName: doc.data().firstName,
    lastName: doc.data().lastName,
    phone: doc.data().phone,
    category: doc.data().category,
    profileImage: doc.data().profileImage,
    location: {
      latitude: doc.data().location.Latitude,
      longitude: doc.data().location.Longitude,
    },
  };

  setOptions((ops) => {
    return [...ops, workerData];
  });
});
```

Figure 33 Search code snapshot

In order to animate the user to the selected region, the “animateToRegion” function is used. The latitude, longitude, and delta values of the target region (the location selected by the user) are passed as arguments as well as the time seconds in which the animation should be completed.

The markers that represent the professionals/businesses that match the search criteria are made with the “Marker” component. In the “coordinate” prop of each marker, the latitude and longitude of the professional's address are passed.

When the tag of a marker is pressed, a modal window opens containing the professional's information (profile image, name, phone, address, rating). The profile image is loaded from Cloud Storage [47] with the help of “getDownloadURL”. Also, the professional's name, phone, address and rating are displayed.

If the user presses the Reviews button, the Ratings Screen opens containing all the reviews the selected worker has received. The implementation of the Ratings Screen is presented in paragraph 5.2.5.

If the user presses the Hire button, a push notification is sent to the selected professional (Figure 35). Expo can handle sending these notifications off to APNs and FCM (Figure 34). This is achieved by sending an HTTPS POST request to Expo’s servers with the ExpoPushToken.

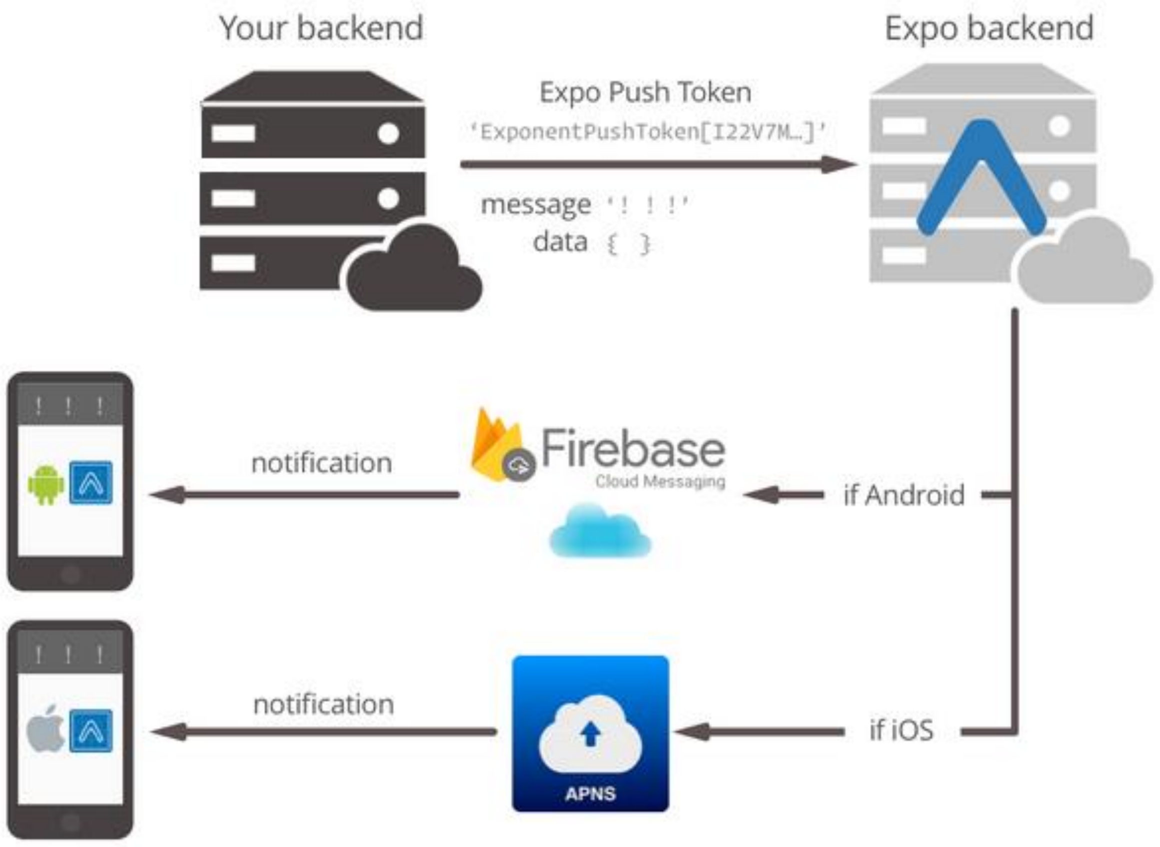
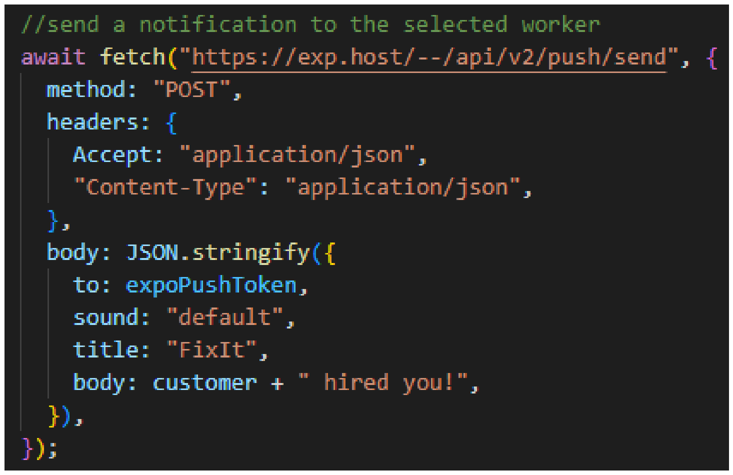


Figure 34 A push notification’s travel through several systems to recipient devices



```
//send a notification to the selected worker
await fetch("https://exp.host/--/api/v2/push/send", {
  method: "POST",
  headers: {
    Accept: "application/json",
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    to: expoPushToken,
    sound: "default",
    title: "FixIt",
    body: customer + " hired you!",
  }),
});
```

Figure 35 Code snapshot to send a push notification to the worker

In this way, the professionals get notified that someone hired them. The “hired you” notification (Figure 36) is received both when the app is in the foreground and the background.

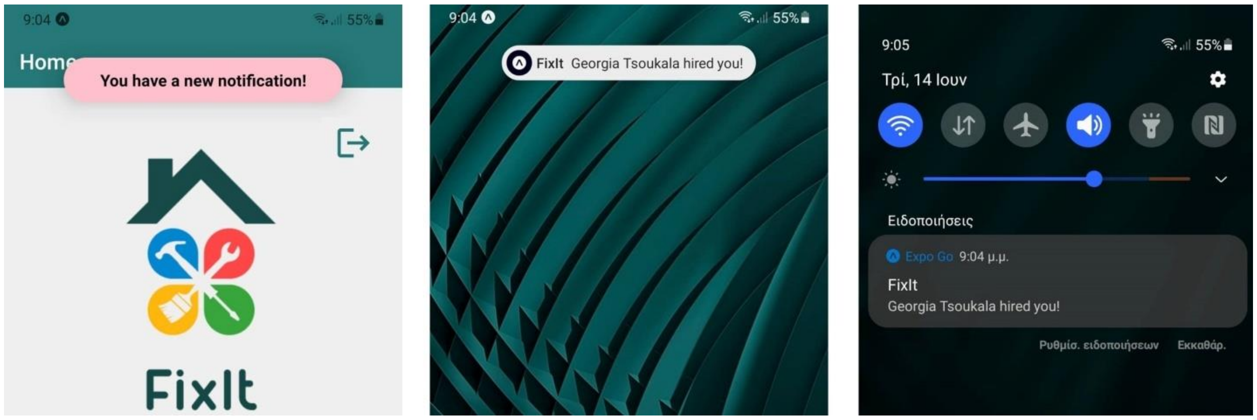
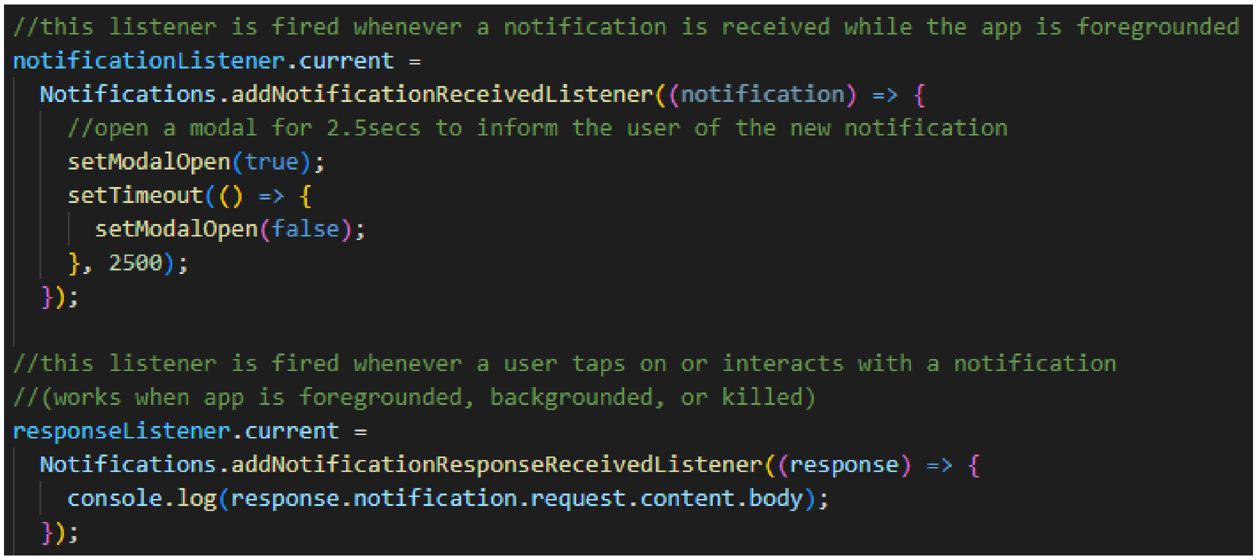


Figure 36 Push Notification when a professional is hired

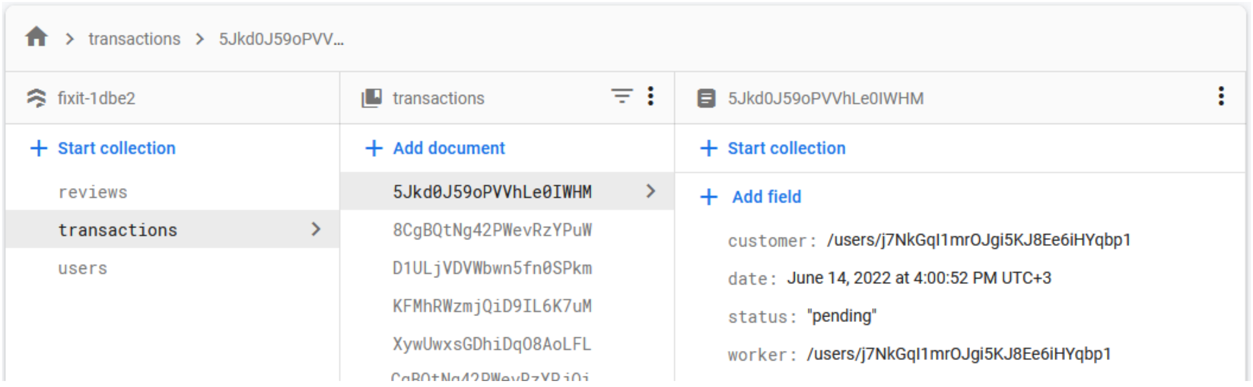
Expo provides the capabilities to update the UI based on a notification, or navigate to a particular screen if a notification is selected. Handling the notifications is simple and straightforward across all platforms with the help of listeners (Figure 37).



```
//this listener is fired whenever a notification is received while the app is foregrounded
notificationListener.current =
  Notifications.addNotificationReceivedListener((notification) => {
    //open a modal for 2.5secs to inform the user of the new notification
    setModalOpen(true);
    setTimeout(() => {
      setModalOpen(false);
    }, 2500);
  });

//this listener is fired whenever a user taps on or interacts with a notification
//(works when app is foregrounded, backgrounded, or killed)
responseListener.current =
  Notifications.addNotificationResponseReceivedListener((response) => {
    console.log(response.notification.request.content.body);
  });
```

Figure 37 Notification listeners

Each transaction is also added by the “addDoc” [36] function in the “transactions” collection containing a reference to the customer, a reference to the worker, the date and “pending” status (Figure 38). When the job is finished the status will get updated to “completed” and then once a review is been made it will get updated to “reviewed”.

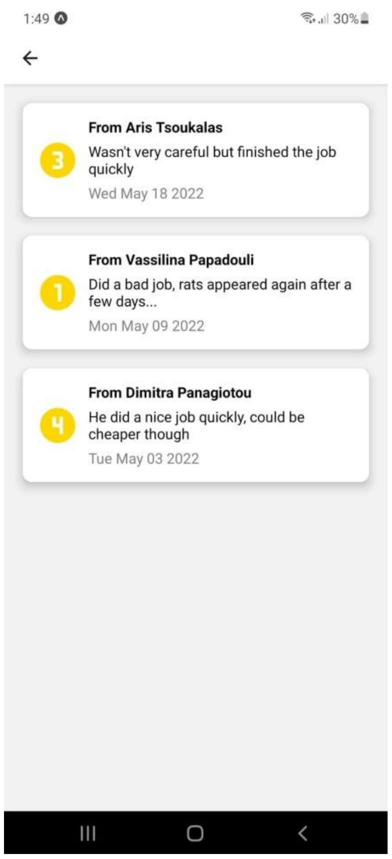


```
reviews
transactions
users

5Jkd0J59oPVVhLe0IWHM
8CgBQtNg42PWevRzYPuW
D1ULjVDVWbwn5fn0SPkm
KFMhRWzmjQiD9IL6K7uM
XywUwxsGDhiDq08AoLFL

customer: /users/j7NkGql1mrOJgi5KJ8Ee6iHYqbp1
date: June 14, 2022 at 4:00:52 PM UTC+3
status: "pending"
worker: /users/j7NkGql1mrOJgi5KJ8Ee6iHYqbp1
```

Figure 38 Transactions' document with pending status

5.2.6 Ratings Screen

The user is navigated to the Ratings screen (Figure 39) by pressing the Reviews button on the pop-up window of the selected professional. On this screen, the user can see all the reviews a worker has received from other customers. Each review contains the rating, the client's name and review and the date it was made.



From Aris Tsoukalas
Wasn't very careful but finished the job quickly
Wed May 18 2022

From Vassilina Papadouli
Did a bad job, rats appeared again after a few days...
Mon May 09 2022

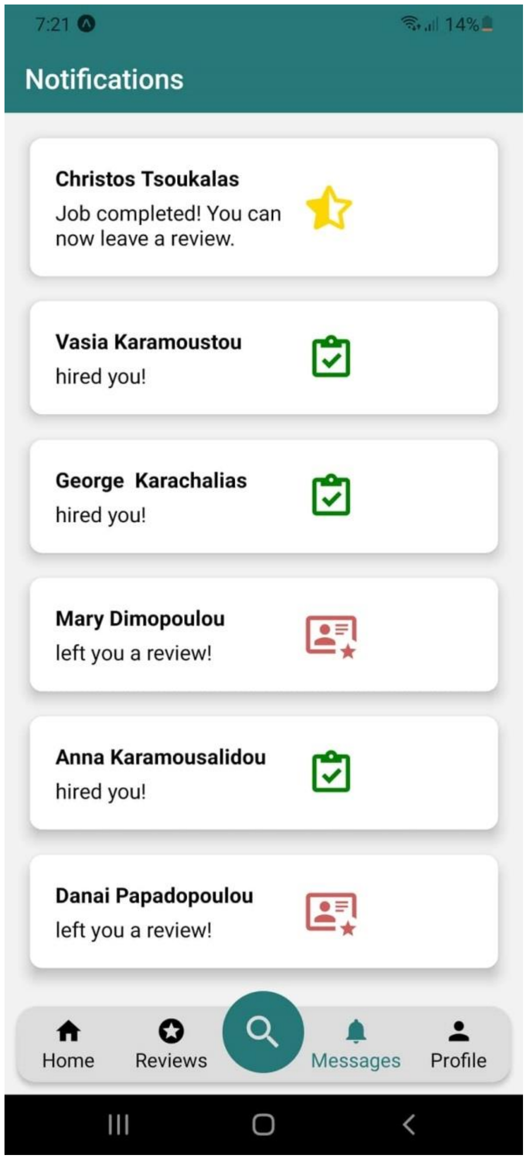
From Dimitra Panagiotou
He did a nice job quickly, could be cheaper though
Tue May 03 2022

Figure 39 Ratings screen layout

To display the reviews on a Flatlist, Firestore's query [41] functionality is used to retrieve all the documents from the "reviews" collection where "worker" is equal to the selected professional. After creating a query object, the "getDocs" [42] function is used to retrieve the results. The data from the snapshots are stored in an array.

5.2.7 Notifications Screen

The Notifications screen (Figure 40) is accessed by selecting the messages tab in the menu. On this screen, the users can see all the notifications they have received. Users with customer accounts receive a notification, once the professional they hired has finished, in order to leave a review. Users with professional accounts can see both the notifications to leave a review to the professionals they hired and the notifications that someone hired them or left them a review. Each notification contains the client's or worker's name, a message and an icon/button depending on the kind of notification.



Notifications

Christos Tsoukalas
Job completed! You can now leave a review.

Vasia Karamoustou
hired you!

George Karachalias
hired you!

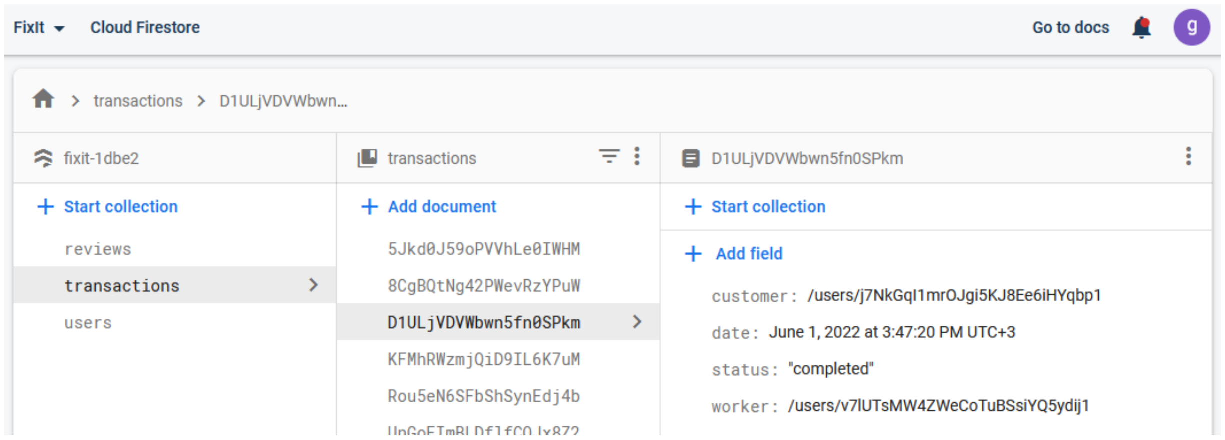
Mary Dimopoulou
left you a review!

Anna Karamousalidou
hired you!

Danai Papadopoulou
left you a review!

Figure 40 Notifications screen layout

If the user presses the icon of a “hired you” notification, the status of the transaction changes from “pending” to “completed” (Figure 41), the notification ceases to be visible on the professional’s notification screen and a notification is sent to the client (Figure 42) to inform them that the job is completed.

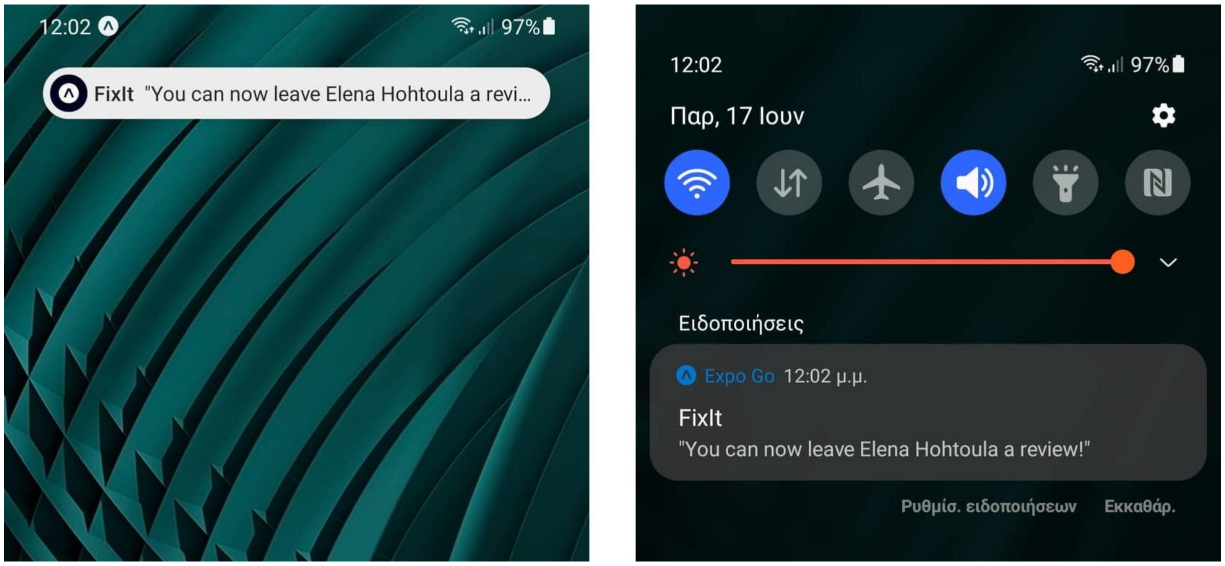


reviews
transactions
users

5Jkd0J59oPVVhLe0IWHM
8CgBQtNg42PWevRzYPuW
D1ULjVDVWbwn5fn0SPkm
KFMhRWzmjQiD9IL6K7uM
Rou5eN6SFbShSynEdj4b
UnGoFImBLDf1fC0Jx872

customer: /users/j7NkGql1mrOJgi5KJ8Ee6iHYqbp1
date: June 1, 2022 at 3:47:20 PM UTC+3
status: "completed"
worker: /users/v7lUTsMW4ZWeCoTuBSsiYQ5ydij1

Figure 41 Transactions' document with completed status

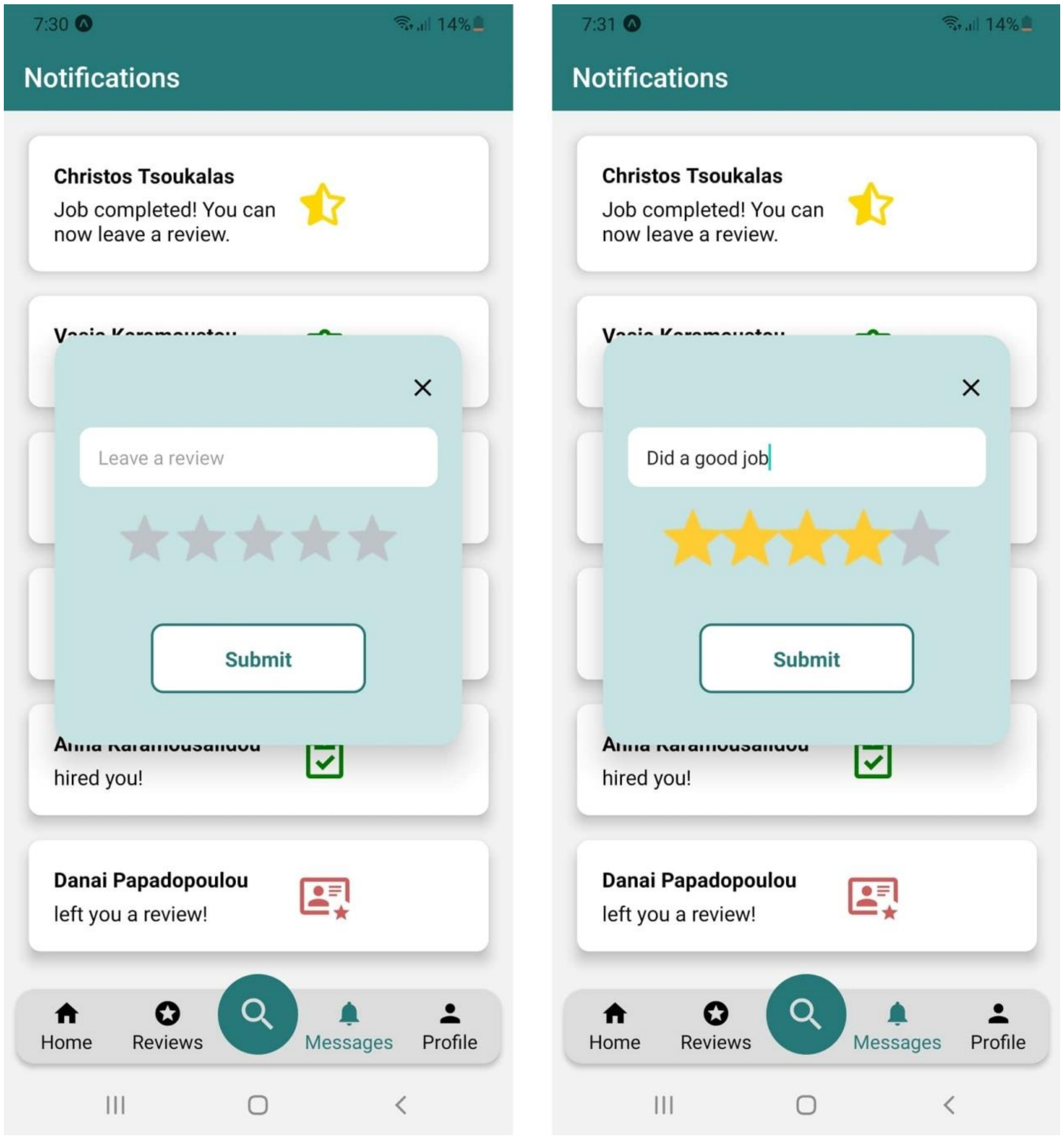


FixIt "You can now leave Elena Hohtoula a revi...

FixIt
"You can now leave Elena Hohtoula a review!"

Figure 42 Push Notification when a job is completed

When a user receives a “job completed” notification, by pressing the notification’s icon they can leave a review (Figure 43) of the service they received.



Christos Tsoukalas
Job completed! You can now leave a review.

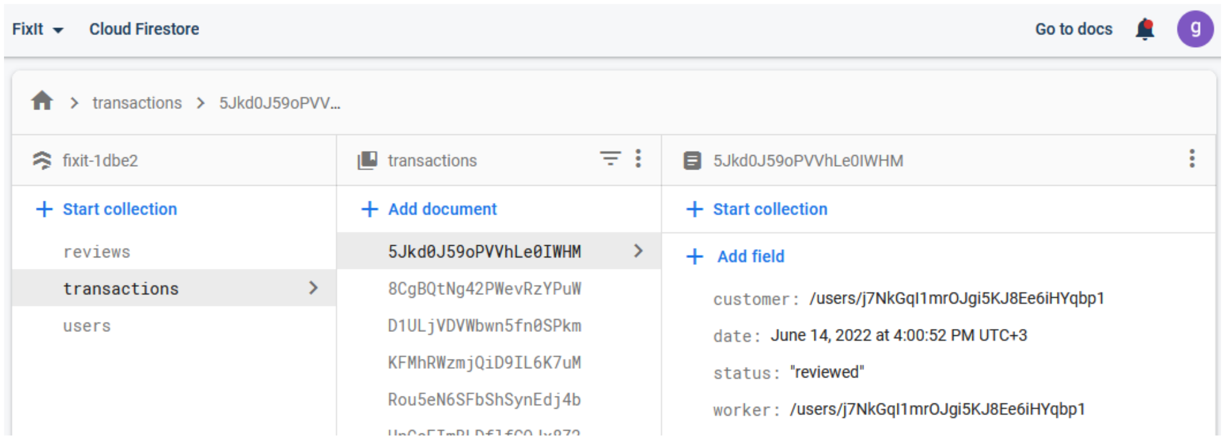
Anna Karamousandou
hired you!

Danai Papadopoulou
left you a review!

Did a good job

Figure 43 Review pop-up

After the review is submitted, the transaction’s status changes from “completed” to “reviewed” (Figure 44) and a notification is sent to the professional (Figure 45) to inform them that they have been reviewed.



```
customer: /users/j7NkGql1mrOJgi5KJ8Ee6iHYqbp1
date: June 14, 2022 at 4:00:52 PM UTC+3
status: "reviewed"
worker: /users/j7NkGql1mrOJgi5KJ8Ee6iHYqbp1
```

Figure 44 Transactions' document with reviewed status

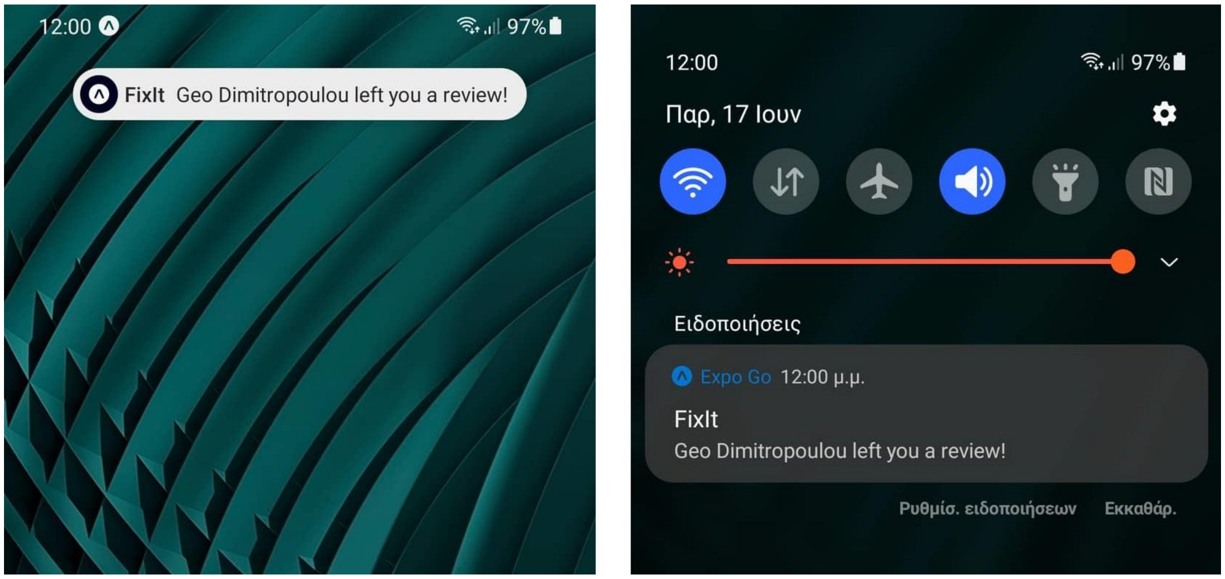


Figure 45 Push Notification when a review is made

When a user receives a “reviewed” notification, by pressing the notification’s icon they get redirected to the Reviews screen and the transaction is deleted from the database with the “deleteDoc” [48] function.

5.2.7 Profile Screen

The Profile screen (Figure 46) is accessed by selecting the profile tab in the menu. The user can edit his profile by selecting a profile image and filling in the input fields with his personal info (first name, last name, phone number, address). If the account is for professional use, a profession has to be selected from the drop-down menu for the user to be put in the correct job category and be able to appear on searches, otherwise, "customer" should be selected.

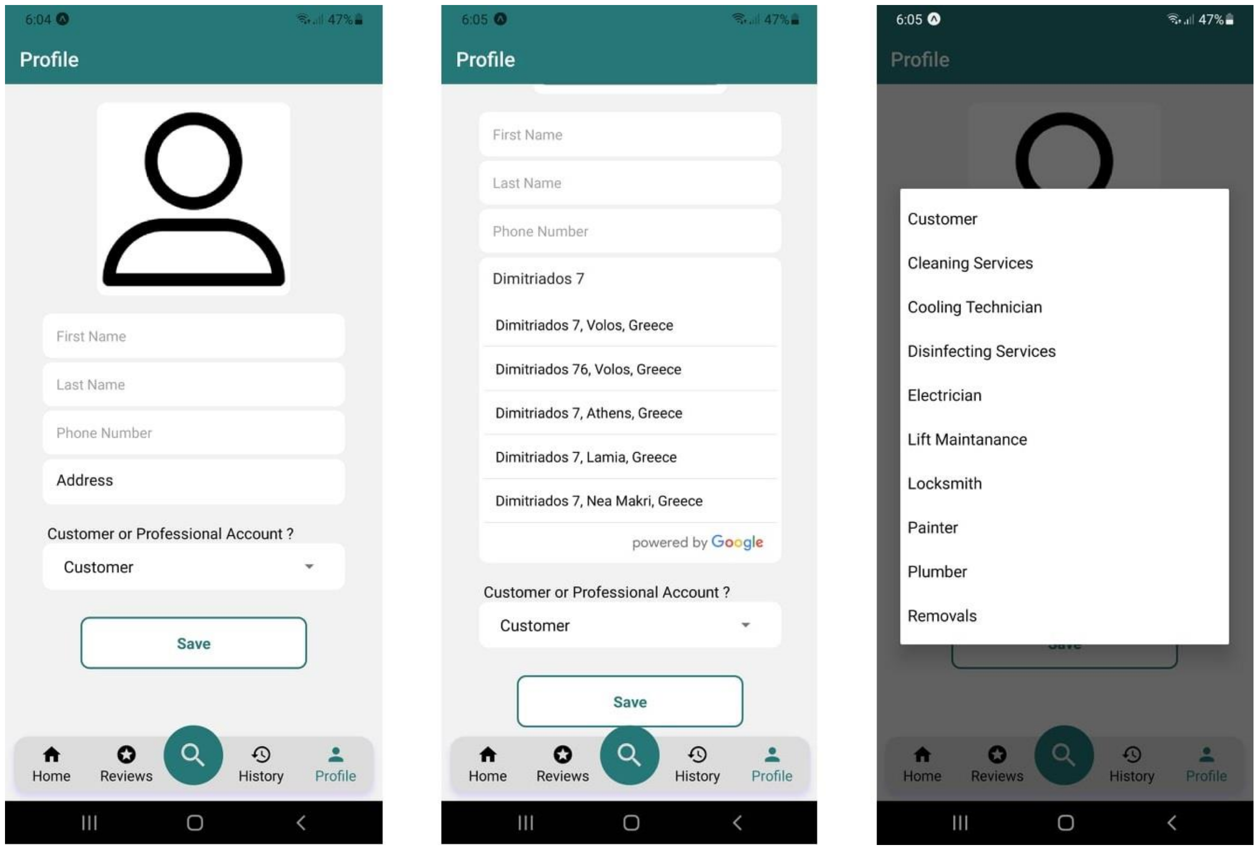
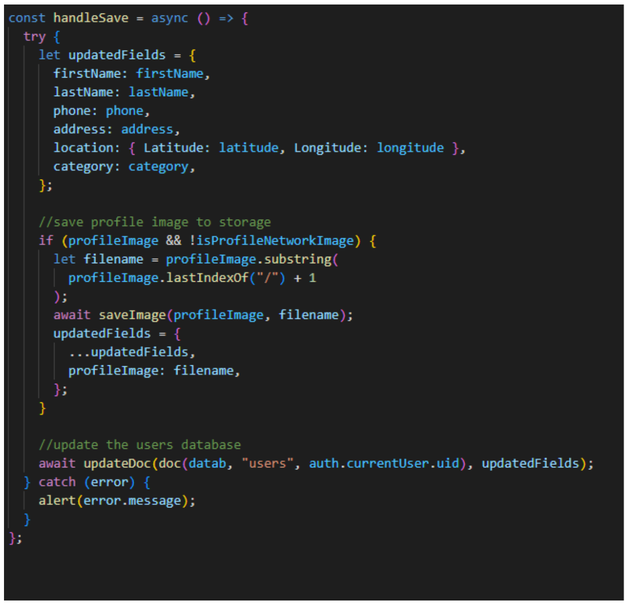


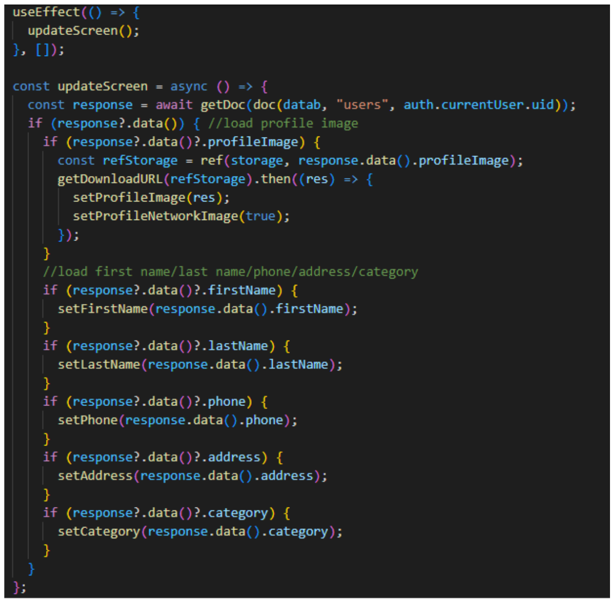
Figure 46 Profile screen layout, address auto-complete, category drop-down

The user is able to select an image from his phone's photo gallery as his profile picture by tapping the profile icon at the top of the Profile screen. With the help of Expo's ImagePicker [49], which provides access to the system's UI for selecting images and videos from the phone's library or taking a photo with the camera, an image URI of the selected picture is returned. This URI is used at the 'source' property of the Image component [50] that is used to display the profile image. Also, the user is asked to enter his first name, last name and phone number in the corresponding input fields. The user's address can be set with the help of "GooglePlacesAutocomplete" mentioned previously on the Search screen. Both the full address and the latitude and longitude info are stored. The drop-down menu is built with Expo's Picker similarly to the one at the Search screen.

By pressing the Save button, the user's profile gets updated (Figure 47). The profile picture's URI is uploaded to Cloud Storage [47] with the help of "uploadBytes" function [51]. To update the user's document in the "users" database with the submitted personal info, the "updateDoc" [36] method is used. First name, last name, phone, address and category are of String data type, while Geopoint is used for the latitude and longitude.



```
const handleSave = async () => {
  try {
    let updatedFields = {
      firstName: firstName,
      lastName: lastName,
      phone: phone,
      address: address,
      location: { Latitude: latitude, Longitude: longitude },
      category: category,
    };

    //save profile image to storage
    if (profileImage && !isProfileNetworkImage) {
      let filename = profileImage.substring(
        profileImage.lastIndexOf("/") + 1
      );
      await saveImage(profileImage, filename);
      updatedFields = {
        ...updatedFields,
        profileImage: filename,
      };
    }

    //update the users database
    await updateDoc(doc(datab, "users", auth.currentUser.uid), updatedFields);
  } catch (error) {
    alert(error.message);
  }
};
```



```
useEffect(() => {
  updateScreen();
}, []);

const updateScreen = async () => {
  const response = await getDoc(doc(datab, "users", auth.currentUser.uid));
  if (response?.data()) { //load profile image
    if (response?.data()?.profileImage) {
      const refStorage = ref(storage, response.data().profileImage);
      getDownloadURL(refStorage).then((res) => {
        setProfileImage(res);
        setProfileNetworkImage(true);
      });
    }
    //load first name/last name/phone/address/category
    if (response?.data()?.firstName) {
      setFirstName(response.data().firstName);
    }
    if (response?.data()?.lastName) {
      setLastName(response.data().lastName);
    }
    if (response?.data()?.phone) {
      setPhone(response.data().phone);
    }
    if (response?.data()?.address) {
      setAddress(response.data().address);
    }
    if (response?.data()?.category) {
      setCategory(response.data().category);
    }
  }
};
```

Figure 47 Save and reload code

Every time the Profile screen is rendered, the user's information is mounted (Figure 47). This is implemented with React's Hook, useEffect [52] which is triggered every time an update happens to the component (Profile screen).

6 Evaluation & Testing

6.1 Functional Requirements Testing

To test whether the functional requirements defined in section 4.4.1 were met by the application, all aspects of the application were tested during and after the implementation. The following table (Table 7) presents the cases that were tested to evaluate the correct functionality of FixIt. Each test case has a unique identifier, followed by the functional requirement it tests, the test input, the pass criteria, and the result.

Table 7 Functional Requirements test cases

Functional Requirements test cases					
ID	**Tested Requirement**	**Test Content**	**Input**	**Pass Criteria**	**Result**
T01	FR01	Checks that the app allows the user to register.	User fills out the sign-up form with his email and password.	User receives a verification email.	PASS
T02	FR02	Checks that the app allows the user to log in.	User fills out the login form with his email and password.	Application redirects the user to the home screen.	PASS
T03	FR03	Checks that the app allows the user to log out.	User presses the logout button.	Application redirects the user to the login screen.	PASS
T04	FR04	Checks that the app shows an error message in case of error during login or registration.	User enters the wrong email or password or registers with a small password.	Application shows a pop-up alert with an error message.	PASS
T05	FR05	Checks that the user can select a photo from their phone as a profile picture.	User presses the profile icon on the profile screen.	The phone’s image gallery opens.	PASS
T06	FR06	Checks that the user can choose if the account is personal or professional.	User presses the dropdown menu on the profile screen.	User can choose between “customer” and a profession option.	PASS

T07	FR07	Checks that the user can edit their profile information.	User changes a field of their profile and presses save.	Application displays the new edited profile info.	PASS
T08	FR08	Checks that the user is able to select the category of the worker they want to search for.	User presses the dropdown menu on the search screen.	User can choose between available profession options.	PASS
T09	FR09	Checks that the user is able to search for a professional in a specific area.	User enters an area in the search field.	User can select an area with the help of autocomplete.	PASS
T10	FR10	Checks that the app displays on a map all the workers that match the search criteria.	User presses the search button on the search screen.	Application displays all the workers that match the search criteria as markers on the map.	PASS
T11	FR11	Checks that the app displays all the necessary information about a professional.	User presses a marker on the map on the search screen.	Application displays a pop-up window with the professional's information.	PASS
T12	FR12	Checks that the user is able to see the reviews each professional has received.	User presses the reviews button of the pop-up window.	Application redirects the user to the ratings screen.	PASS
T13	FR13	Checks that the user is able to leave a review of the worker they hired.	User presses a “job completed” notification.	Application displays a pop-up window where the user can rate the professional.	PASS
T14	FR14	Checks that the user is able to view all past reviews they made.	User opens the reviews screens.	Application displays all the past reviews the user has made.	PASS
T15	FR15	Checks that the user (professional) is able to view all the reviews they have received.	User opens the reviews screens.	Application displays all the past reviews the professional has received.	PASS

6.2 Non-Functional Requirements Evaluation

The following table (Table 8) presents whether the non-functional requirements of FixIt which were defined in Section 4.4.2 were successfully fulfilled. Each test case has a unique identifier, followed by the functional requirement it tests, the pass criteria, and the result.

Table 8 Non-Functional Requirements test cases

Non-Functional Requirements test cases				
ID	**Tested Requirement**	**Test Content**	**Pass Criteria**	**Result**
T01	NFR01	Checks whether the GUI is easy to use.	Any user can easily navigate around the app.	PASS
T02	NFR02	Checks whether all screens have a similar style.	The application offers the same theme and style for menus, buttons and layouts on all screens.	PASS
T03	NFR03	Checks whether the system displays detailed notification/error messages to the user.	The system shows a detailed message for every action a user performs or if an error occurs.	PASS
T04	NFR04	Checks whether the system requires a password from each user.	If the user enters an incorrect email or password access is denied.	PASS
T05	NFR05	Checks whether the system has a fast response time.	This was tested by using all the available features of the application.	PASS
T06	NFR06	Checks whether the system has any possible bugs in its operations.	This was tested by the produced test cases in the previous section.	PASS
T07	NFR07	Checks whether the system is able to run on all devices regardless of their OS.	This was tested by running the app both on Android and iOS devices.	PASS
T08	NFR08	Checks whether the system is capable of being maintained/repaired.	New features can be added and bugs can be fixed.	PASS
T09	NFR09	Checks whether the system can be easily changed to meet new requirements.	The application accepts new features by adding functions or modifying the existing ones.	PASS
T10	NFR10	Checks whether the system is able to handle a considerable number of users.	This was tested by running the app with a big number of users.	PASS

7 Conclusions

7.1 Concluding Remarks

In this Thesis, the main goal was to study the current way people consume home services, the way professionals market their businesses and the benefits of online on-demand home services. Also, another goal was to design and develop a mobile application where clients can connect with the most suitable tradesperson.

The theoretical analysis of the home services industry from the clients' and the professionals' points of view, as well as the study of existing applications resulted in a draft of the FixIt app with all the required features.

Also, analyzing different software development process models helped to decide which was the most useful for the development of this project's mobile application. The agile development model was selected because it would allow adapting the requirements during the development process, improving the application's design by adding more features and fixing arising bugs. At the same time, principles of mobile interface design were studied as they are important for the successful creation of an app.

Researching for the most suitable tools to develop an application that runs on every device regardless of the operating system was also critical. The application that was finally developed provides significant benefits to the users, as it enables the customers to search for the right professional and the professional to find new clientele. With further improvement from future developers, the already useful application could become a fully operational product.

To conclude, all goals were accomplished since a study of the home services market was made and a mobile application that met all the functional and non-functional requirements was developed.

7.2 Future Work

Regarding future work, there are possible improvements that can be made to improve the application and add new functionality. Some of them are:

a. Price lists: List the services each professional provides as well as the cost of each service. The price list could help the customers who want to know the cost of services so that they can decide which professional they will hire.
b. Schedule appointments: Professionals will provide their schedule so clients can book services in an instant and easily schedule home care and maintenance projects.
c. Chat: Customers will be able to communicate with the professionals through a chat thread, where they can chat or call to discuss anything related to an offered service.
d. Online payments: The platform could be cashless so there will be no need for cash to change hands. The payments will be automatically charged to the credit or debit card of the client. A customer's invoice will include the number of hours worked plus any agreed-upon expenses.

Code Repository

This project's code is hosted on Github in the following repository:

https://github.com/GeorgiaTsoukala/FixIt-App/tree/tsalapata

Bibliography

[1] React Native: https://reactnative.dev/

[2] Firebase: https://firebase.google.com/

[3] Mirco Franzago, Ivano Malavolta, Henry Muccini: Stakeholders, Viewpoints and Languages of a Modelling Framework for the Design and Development of Data-Intensive Mobile Apps. February 2015.

[4] Shweta Sarma: Home Services in 2020 and the Changing Role of Logistics.

[5] Huete-Alcocer: A Literature Review of Word of Mouth and Electronic Word of Mouth: Implications for Consumer Behavior. July 2017.

[6] James Karthick: Types of Advertisement.

[7] Grubor Aleksandar, Olja Jaksa: Internet Marketing as a Business Necessity. June 2018.

[8] Suzana Z. Gildin: Understanding the power of word-of-mouth. 2003.

[9] David M. Gardner: Deception in Advertising: A Conceptual Approach. June 2022.

[10] Fatima Naz: Word of Mouth and Its Impact on Marketing. January 2014.

[11] Tashrifa Haider: A Study on the Influences of Advertisement on Consumer Buying Behavior. 2017.

[12] Uchechi Cynthia Ohajionu, Soney Mathews: Advertising on social media and benefits to brands. January 2015.

[13] Robert Zinko, Angela Patrick, Christopher P. Furner , Shalanda Gaines, Mi Dya Kim, Matthew Negri, Elsy Orellana, Shelby Torres, Carmen Villarreal: Responding to Negative Electronic Word of Mouth to Improve Purchase Intention. 2021.

[14] Nadine Höchstötter, Dirk Lewandowski: What Users See – Structures in Search Engine Results Pages. 2009.

[15] Pei-yu Chen, Shin-Yi Wu: The Impact and Implications of On-Demand Services on Market Structure. September 2013.

[16] Florina Pinzaru, Alexandra Zbuchea: Mobile Applications: From Business to Social Implication. September 2017.

[17] TaskRabbit: https://www.taskrabbit.com/

[18] Urban Company: https://www.urbancompany.com/

[19] Angi: https://www.angi.com/

[20] Douleutaras: https://www.douleutaras.gr/

[21] Brightlocal: Local Consumer Review Survey. January 2022.

[22] Iqbal H. Sarker, Md. Faisal Faruque, Ujjal Hossen, Atikur Rahman: A Survey of Software Development Process Models in Software Engineering. International Journal of Software Engineering and its Applications. November 2015.

[23] Jiujiu Yu: Research Process on Software Development Model. IOP Conference Series Materials Science and Engineering. August 2018.

[24] Kai Petersen, Claes Wohlin, Dejan Baca: The Waterfall Model in Large-Scale Development. International Conference on Product-Focused Software Process Improvement. June 2009.

[25] Claes Wohlin: Improving through an Incremental Approach. Proceedings 2nd European Industrial Symposium on Cleanroom Software Engineering, Berlin. Germany, 1995.

[26] Dhruv Doshi, Labdhi Jain, Kunj Gala: Review of the spiral model and its applications. International Journal of Engineering Applied Sciences and Technology. 2021.

[27] Nripesh Kumar Nrip, Gunjan Behl: A study of agile software development model. March 2012

[28] B. Ballard: Designing the Mobile User Experience. 2007.

[29] Jonathan Stark: Principles of Mobile Interface Design. March 2012.

[30] Leonardo Corbalán, Pablo Javier Thomas, Lisandro Delia, Germán Cáseres, Juan Fernandez Sosa, Fernando Tesone, Patricia Pesado : A Study of Non-functional Requirements in Apps for Mobile Devices. July 2019

[31] Visual Studio: https://code.visualstudio.com/

[32] Expo: https://expo.dev/

[33] Who uses Firebase: https://stackshare.io/firebase

[34] Firebase Authentication: https://firebase.google.com/docs/auth

[35] Password-Based Accounts: https://firebase.google.com/docs/auth/web/password-auth

[36] Add data to Firestore: https://firebase.google.com/docs/firestore/manage-data/add-data

[37] Cloud Firestore Data model: https://firebase.google.com/docs/firestore/data-model

[38] Stack Navigator: https://reactnavigation.org/docs/stack-navigator/

[39] Bottom Tab Navigator: https://reactnavigation.org/docs/bottom-tab-navigator/

[40] Notifications: https://docs.expo.dev/versions/latest/sdk/notifications/

[41] Queries in Cloud Firestore: https://firebase.google.com/docs/firestore/query-data/queries

[42] Get data with Firestore: https://firebase.google.com/docs/firestore/query-data/get-data

[43] Autocomplete: https://www.npmjs.com/package/react-native-google-places-autocomplete

[44] Google Cloud overview: https://cloud.google.com/docs/overview

[45] React Native Maps: https://github.com/react-native-maps/react-native-maps

[46] Map view: https://docs.expo.dev/versions/latest/sdk/map-view/

[47] Cloud Storage for Firebase: https://firebase.google.com/docs/storage

[48] Delete data: https://firebase.google.com/docs/firestore/manage-data/delete-data

[49] Image Picker: https://docs.expo.dev/versions/latest/sdk/imagepicker/

[50] Image: https://reactnative.dev/docs/image

[51] Upload files: https://firebase.google.com/docs/storage/web/upload-files

[52] Effect Hook: https://reactjs.org/docs/hooks-effect.html