People's Democratic Republic of Algeria
Ministery of Higher Education and Scientific Research
Mohamed El Bachir El Ibrahimi University of Borj Bou Arréridj
Faculty of Mathematics and Informatics
Informatics Department



DISSERTATION
Presented in fulfillment of the requirements of obtaining the degree
Master in Informatics
Specialty: Information Technologies and Communications

THEME

Secure P2P(peer-to-peer) Escrow Platform

Presented by:

1. Mekkari Zakarya
2. Drahmoun Wahid
3. Bounabi Ilyes
4. Rehahela Oussama

Publicly defended on: 3/7/2024 , In front of the jury composed of:

President: Boutouhami Sara
Examiner: Belalta Ramla
Examiner: Loukrise Samia
Supervisor: Ms. Badaoui Atika

2023/2024

إهداء

مَن قال : أنا لَها ... نالَها .

وأنا لَها، وإن ابت رغماًا عنها، أتيت بها .

الحَمْدُ لله حَبًّا وشُكْرًا وامتِناناًا على البدء والختام .

(وآخِرَ دَعْوَاهُم أن الحمْدَ لله رَب العالَمين)

قال رَسول الله صَلّى الله عَلَيْه وسَلّم:

(مَن صَنَع إلَيْكُم مَعْرُوفاًا فكافِئُوه، فَإن لَم تَجِدُوا ما تكافِئُوا لَه بِه فَادْعُوا لَه حَتّى تَرَوْا أنكم كَفَيْتُمُوهْ)

أُهَدِي هَذا التَخْرُج لِمَن كَلل عَرَق جَبِينَه وعَلّمَنِي أن النّجَاح لا يَأْتِي الا بِالصّبْرِ والاصرَارِ، وأتَمَنّى أن يُطِيل وَيُبَارِكَ الله فِي عُمْرِه . - والدي العزِيز -

إلى مَن جَعَل الله جَنّة تَحْت أقْدَامِهَا وسَهّلَت لِي الشّدَائِدَ بِدُعَائِهَا، إلى الإنسانة العظيمة التي لَطَالَمَا تَمَنّيْت أن تَقُرّ عَيْنُهَا بِرُؤيَتِي فِي هذا اليوم . - والدتي الحَبِيبَة -

لأولئك الذِين لَدَيْهِم مِيزَة كَبِيرَة في تَشْجِيع وتَحفِيزِ والذِين تَعَلّمْت مِن مُثَابَرَتِهِم وَاجْتِهَادِهِم . - أخِي وَأُخْتِي -

اشكر الله كل يوم على هديته التي هي انت - خطيبتي -

وَلِكُل مَن كَان عَوْنًا وَسَنَدًا فِي هذا الطّرِيق، لِلأصْدِقَاء الأوفِيَاء ورِفَاق السّنِين وأصْحَاب الشّدَائِد والأزْمَات . - أَصْدِقَائِي -

هَا أنَا الْيَوْم أتَمّم هَذَا الإنْجَازَ بِفَضْل الله عَزّ وَجَل، فَالحَمْدُ لله عَلى مَا وَهَبَنِي .

- مكاري زكرياء -

إهداء

ما سَلَكْنَا البدَايَات إلا بِتَيْسِيرِه، وما بَلَغْنَا النّهَايَات إلا بِتَوْفِيقِه، وما حَقّقْنَا الغَايَات إلا بِفَضْلِه، فَالْحَمْد لِلّه الّذِي وَفّقَنَا لِتَثْمِين هَذِه الخُطُوَة فِي مَسِيرَتِنَا الدّرَاسِيّة.

أُهْدِي هَذَا النّجَاح إلَى مَن قَال فِيهِم اللّه تَعَالَى

(وَقَضَى رَبُّك أَلا تَعْبُد إلا إيّاه وَبِالْوالِدَيْن إحْسَانًا)

- إلى والدي العزِيز -

لِلرّجُل الّذِي علّمَنِي الصّبْر وَالْقُوّة، إلَى مَن كَان لِي دَائِمًا قُدْوَة وَمِثَالًا أَعْلَى، وَأَتَذَكّر كُل دَعْم غَيْرِ مَشْرُوط مُنْحَتْنِي إيّاه، وَأَتَمَنّى أَن أَكُون دَائِمًا عَنْد حَسَن ظَنّك.

- إلى والدتي الحَبِيبَة -

مِن أَوّل خُطُوَة نَخْطُوهَا إلَى كُل انْتِصَار أَحْقَق، كُنْتي دَائِمًا إلَى جَانِبِي تُشْجِعُنِني وَتُدْعِمُنِني. كَلِمَات الشّكْر لَا تَكْفِي لِلتّعْبِير عَن مَدَى الامتِنَان.

- إلى أخي -

مُنْذ أَن كُنْت صَغِيرًا وَحَتّى الآن، كُنْت دَائِمًا مَعِي، تَقَاسَمْت مَعَك الضّحِك وَالْبُكَاء، وَوَقَفْت بِجَانِبِي فِي كُل تَحَدّي أَو صُعُوبَة. أَنْت شَخْص الّذِي يُمْكِن الِاعْتِمَاد عَلَيْه فِي كُل الْأَوْقَات، وأُتَمنّى لَك دَوْمًا النّجَاح.

وَخْتَامًا إلى كُل صَدِيق وَفِرْد مِن دَائِرَة حَيَاتِي، إلى كُل مَن زَرَع فِي قَلْبِي أَمَلا أَو أَضَاف لَمْسَة خَاصّة فِي مَسِيرَتِي، شُكْرًا لَكُم كُل بِاسْمِه وَمَقَامِه.

- درحمون وحيد -

إهداء

بسم الله الرحمن الرحيم.

إلى من كان لهم الفضل بعد الله في إتمام مسيرتي الدراسية.

إلى والديّ العزيزين، الذين كانوا دائماًا مصدر إلهامي ودعمي، شكراً لكما على تضحياتكما وحبكما غير المشروط.

إلى أساتذتي الكرام، الذين لم يبخلوا عليّ بالنصح والتوجيه.

إلى أصدقائي وزملائي، الذين كانوا دائماً بجانبي في لحظات الجد والاجتهاد.

إلى كل من ساهم في وصولي إلى هذه المرحلة، من قريب أو بعيد.

إلى كل من ساهم في دعمي وتشجيعي، ولو بكلمة طيبة أو دعوة صادقة، أهديكم هذا العمل تقديراً وامتناناًا.

- أسامة رحاحلة-

إهداء

بداية نرفع أكفّنا بالدعاء لله سبحانه وتعالى، الذي ألهمنا قوّة الإرادة، ووهبنا الصبرَ، ليُكلّل مسعانا بالنجاح. فله الحمدُ والشكرُ. اتقدم لمصدر الهامي والدي وعائلتي بكل امتنان وشكر لدعمي طوال

مشواري هدا على ان اصبح ما انا عليه اليوم.

اقدّم جزيل الشكر والتقدير لأستاذنا المشرف، الأستاذة بداوي عتيقة على

مجهوداتها ونصائحها الهادفة.

اتوجه بِخالص الشكر والتقدير إلى أعضاء لجنة التحكيم الموقرين، الذين تفضلوا بقراءة رسالتنا بدقة وعناية، ونُثمن عالياً ملاحظاتهم القيّمة التي

ستُساهم بلا شك في إثراء بحثنا وصقله.

لا يسعني إلا أن اعبر عن عميق امتناني لأصدقائنا وزملائنا الأعزاء، الذين كانوا خيرَ عونٌ لنا على مرّ السنين. فكانوا سنداً لنا في لحظات

الفرح، ودعامةً قويةً في أوقات الشدائد.

اشكر جميع من ساهم بشكل مباشر أو غير مباشر في إنجاز هذه الرسالة، ممّن قدموا لنا الدعم والتشجيع، وشاركوا معنا أفكارهم وخبراتهم. إذّ كلمات الشكر والعرفان لا تُوفي حقّ من ساهم في إنجاز هذا العمل، لكنّها تبقى تعبيراً خالصاً عن مشاعرنا وامتناننا العميق. نسأل الله تعالى أن يُوفقّنا لخدمة العلم والمعرفة، وأن يُسعدنا بالمزيد من

العطاء والإنجاز مع خالص الشكر والتقدير.

- بونابي الياس -

Acknowledgment

We would like to express our sincere gratitude to “Ms. Badaoui Atika”, our supervisor, for his invaluable guidance, support, and encouragement throughout the entire process of conducting this research and writing this dissertation.

We are also grateful to the members of the jury, “Belalta Ramla” and “Boutouhami Sara” and “Loukrise Samia”, for taking the time to review and evaluate our work, and for their constructive feedback and insightful comments.

We would like to extend our thanks to the business incubator of the University “Mohamed El Bachir El Ibrahimi - Bordj Bou Arreridj”, for providing us with the necessary resources and facilities to carry out this project.

Abstract

In this dissertation, we focus on designing and implementing a platform that offers a unique experience for service providers, freelance contractors, and customers.

Our platform, named "Khedemni," has been created to streamline service operations between providers and clients. Customers can search for the right professional for their required services, while service providers can showcase their offerings and interact directly with clients.

Our platform features a secure payment system, acting as a financial intermediary between the two parties. Once a service is selected and requirements are set, the client pays an amount that is held until the service is completed.

Our platform contributes to developing a safe and reliable environment for both parties. As a startup in the Algerian market, it brings mutual benefits to both service providers and customers.

Keywords :

Mobile applications , Service , UML , Money transfer , Startup , Flutter , Firebase , Financial intermediary , Dart , Platform , Self-employed contractors , Customers , Khademni , Service operations , Clients , Secure environment , Algerian market , Direct interaction , Development , Financial security

Résumé

Dans ce Mémoire, nous concentrons sur la conception et la mise en œuvre d'une plateforme offrant une expérience unique aux prestataires de services, aux travailleurs indépendants et aux clients.

Notre plateforme, nommée "Khedemni", a été créée pour rationaliser les opérations de service entre les prestataires et les clients. Les clients peuvent rechercher le professionnel approprié pour leurs services requis, tandis que les prestataires de services peuvent présenter leurs offres et interagir directement avec les clients.

Notre plateforme dispose d'un système de paiement sécurisé, agissant comme intermédiaire financier entre les deux parties. Une fois qu'un service est sélectionné et que les exigences sont définies, le client paie un montant qui est retenu jusqu'à ce que le service soit terminé.

Notre plateforme contribue au développement d'un environnement sûr et fiable pour les deux parties. En tant que start-up sur le marché algérien, elle apporte des avantages mutuels aux prestataires de services et aux clients.

Mots clé :

Applications mobiles , Service , UML , Transfert d'argent , Startup , Flutter , Firebase , Intermédiaire financier , Dart , Plateforme , Entrepreneurs indépendants , Clients , Khademni , Opérations de service , Clients , Environnement sécurisé , Marché algérien , Interaction directe , Développement , Sécurité financière

ملخص

نَعْمَل فِي هَذِه الْمُذَكِّرَة عَلَى تَصْمِيم وَتَنْفِيذ مَنْصّة تُقَدِّم تَجْرِبَة فَرِيدَة لأَصْحَاب الْخَدَمَات وَالْمُقَاوِلِين الذّاتِيِّين وَكَذَلِك الزّبَائِن.

تَم إِنْشَاء مَنْصّتنَا الْمُسَمّاة ْ خَدِّمْنِي ْ لِتَسْهِيل الْعَمَلِيّات الْخِدْمِيّة بَيْن أَصْحَاب الْخَدَمَات وَالْعُمَلَاء، حَيْث يُمْكِن لِلزّبَائِن الْبَحْث عَن الْعَامِل الْمُنَاسِب لِخَدَمَاتِهِم الْمَطْلُوبَة، وَبِالنِّسْبَة لِلْعَامِلِين، يُمْكِنُهُم عَرْض خَدَمَاتِهِم وَالتّفَاعُل الْمُبَاشِر مَع الْعُمَلَاء.

تَتَمَيّز مَنْصّتُنَا بِنِظَام آمِن لِتَحْوِيل الْأَمْوَال، حَيْث تَعْمَل كَوَسِيط مَالِي بَيْن الطّرَفَيْن، وَبِمُجَرّد اخْتِيَار الْخِدْمَة وَتَحْدِيد الْمُتَطَلّبَات، يَقُوم الْعَمِيل بِدَفْع مَبْلَغ مَالِي يُحْتَجَز حَتّى اكْتِمَال الْخِدْمَة.

تُسَاهِم مَنْصّتُنَا فِي تَطْوِير بِيئَة آمِنَة وَمَوْثُوقَة لِكِلَا الطّرَفَيْن، وَ شَرِكَة نَاشِئَة فِي السّوق الْجَزَائِرِيّة مِمّا يَعُود بِالْفَائِدَة عَلَيْهِمَا عَلَى حَد سواء

الكلمات المفتاحية :

تطبيقات الهواتف المحمولة، خدمة، لغة النمذجة الموحدة، تحويل الأموال، شركة ناشئة، فلاتر، فايربيس، وسيط مالي، دارت، منصة، مقاولون مستقلون، زبائن، خدامني، عمليات الخدمة، عملاء، بيئة آمنة، السوق الجزائري، تفاعل مباشر، تطوير، الأمن المالي.

Table of contents

List of Figures **xiii**

List of Tables **xv**

General Introduction **1**

Chapter 1: Literature Review **3**

1.1 Introduction 3

1.2 Overview About Escrow 3

1.2.1 What Does Escrow Mean 3

1.2.2 Types of Escrow 4

1.2.3 Advantages and Disadvantages of Escrow 5

1.3 Overview About Online Payment System 6

1.3.1 What is Online Payment System 6

1.3.2 Virtual Money Transfers 6

1.3.3 E-Wallets 8

1.4 Conclusion 9

Chapter 2: Peer-to-Peer Platforms and Escrow: A Comparative Overview **10**

2.1 Introduction 10

2.2 Peer-To-Peer Service Platforms 10

2.2.1 Definition 10

2.2.2 Examples of Peer-to-Peer Service Platform Around the World 11

2.2.3 Comparison of Popular Peer-to-Peer Platforms: 12

2.2.4 Advantages and Disadvantages of Peer-To-Peer Service Platform . . . 16

2.3 Conclusion 17

Chapter 3: System Requirements and Design **18**

3.1 Introduction 18

3.2 Project Management Methodology (Agile) 19

3.2.1 Details of Agile Methodology 19

3.2.2 Applying agile methodology 20

3.3 Non-Functional Requirements 21

3.4 Functional Requirements 22

3.5 System Design 24

3.5.1 System Modeling 24

3.5.2 Database Design 36

3.6 Conclusion 37

Chapter 4: Implementation **39**

4.1 Introduction 39

4.2 development frameworks, tools and APIs 39

4.2.1 Front-End 39

4.2.2 Back-End 42

4.2.3 Programming Tools: 44

4.3 Programming Tasks 45

4.4 System Architecture 46

4.4.1 High Level System Design 46

4.4.2 Model View Controller 47

4.4.3 RESTful Services 49

4.4.4 Socket.IO Overview and Workflow Explanation 50

4.5 Installation and Project Initialization 53

4.5.1 Installation Steps 54

4.5.2 Project Initialization 56

4.5.3 Pushing to GitHub Repository 58

4.6 Testing 59

4.6.1 Front End Testing 59

4.6.2 Back End Testing 59
4.7 Deployment 61
4.7.1 Docker and Docker Compose 61
4.7.2 Final Look of the Platform 65
General Conclusion **73**
References **73**
Annexes **77**

List of Figures

1.1 An Example of Depositing Money Process 8

3.1 An example of agile software development methodology: Scrum 20
3.2 Agile Methodology For Platform Development 21
3.3 khdemni use case diagram 27
3.4 Table 1. “Request a service” use case 28
3.5 Table 2. “View service requests” use case 29
3.6 Table 3. “Reserve service” use case 30
3.7 Table 4. “Manage dispute” use case 31
3.8 Sign-Up Sequence 32
3.9 Login Sequence 33
3.10 Request Service Sequence 34
3.11 Class Diagram 35
3.12 Database Modeling 36

4.1 An Excel sheet for the project progress form. 46
4.2 High level system design flow 46
4.3 (Model View Controller) Architecture in the MERN Stack 47
4.4 Project Structure. 49
4.5 What Is A Rest API 50
4.6 Real-Time Flowchart for KHDEMNI platform 51
4.7 NodeJs Installation Setup. 54
4.8 MongoDb Compass Installation Setup 55
4.9 Git Installation Setup 55
4.10 Docker and Docker Compose Installation Setup 56

4.11 running git push command 59
4.12 GitHub repo activity up to date 59
4.13 Front-End Unit Tests Example. 60
4.14 Back-End End Unit Tests Example. 60
4.15 Docker Compose Building Stage 63
4.16 Docker Compose Running Stage 64
4.17 Docker Final Deployment 64
4.18 Sign Up Platform 65
4.19 Sign In Platform 65
4.20 Home Page Platform 66
4.21 Profile In Platform 66
4.22 Platform 67
4.23 Wallet In Platform 67
4.24 Requests in Platform 68
4.25 Offers In Platform 68
4.26 Create New Offer In Platform 69
4.27 Modify Offer In Platform 69
4.28 Mobile 70
4.29 Dashboard In web application 71
4.30 Service provider In application 71
4.31 Contact Us In application 72

List of Tables

2.1 Comparison of The Most Popular Peer-to-Peer Platforms for Freelancers and Service Providers. 15

3.1 Roles and Functionalities for Different Actors 26

4.1 Tools Used in Developing the KHDEMNI Platform. 45

General Introduction

In our contemporary digital landscape, guarantees and virtual money transfers stand as pivotal concepts, crucial for bolstering trust and security in financial transactions. They underpin the modern payment system, enabling both individuals and institutions to conduct financial transfers with confidence and efficiency. As reliance on digital services grows, the demand intensifies for secure platforms that safeguard funds and streamline transactions between service providers and customers, fostering a dependable environment conducive to the proliferation of online home services. Addressing the challenges specific to Algeria, where accessing reliable home services like crafts, private education, and cleaning can be daunting, our platform, "Khadmni," ensures fund security until service completion, catering as an innovative solution tailored to the digital age.

In Algeria, finding home services often represents a major challenge. Individuals looking for services, whether crafts, private education, healthcare, cleaning... etc., often encounter difficulties in finding reliable providers, in arranging appointments or even in accessing a diversity of services. choice in one place. This situation can lead to wasted time, and sometimes even limit access to essential services at the appropriate time.

Traditional payment methods contribute to customer apprehension, leading to hesitation in making upfront payments, and leaving service providers at risk of non-compensation. This trust deficit poses a significant hindrance to the growth of the freelance economy, as clients demand secure, efficient, and transparent service transactions. For their part, service providers may face obstacles in reaching their target audience, managing their schedule effectively, or establishing strong connections with potential clients, thereby limiting their ability to offer their skills and services optimally. In addition, the dearth of reliable platforms contributes to a lack of flexibility in payment methods, hindering clients' ability to exercise control over their financial transactions. Service quality assurance is also compromised, as clients find it challenging to

review and select service providers based on past performance, ratings, and reviews.

The overarching problematic is clear: Algeria's freelance industry faces a trust and security crisis, with an urgent need for a comprehensive solution to facilitate secure, efficient, and transparent P2P interactions, ultimately fostering growth in the freelance economy.

Our goal in this dissertation is to develop and design a secure service provision platform called "Khadmni" that helps solve some of the problems mentioned above. We have followed the agile project management methodology, which allows us to adapt to changes and ensure reaching our main goal, which is to create a platform that ensures:

- Financial guarantee system: which enhances confidence and protects payments.
- Unique user experience: Facilitating direct communication and achieving improved user experience.
- Payment after service: Enhanced monitoring and security for customers.
- Comprehensive Service Offering: Allowing service providers to promote and manage their offerings.

With a focus on using innovative technologies in payment management and communications, the platform offers a comprehensive and effective solution to the crisis of trust and security in the freelance sector in Algeria.

After we presented in the **first chapter** the design goal of the platform and the issues faced by service providers and customers, this note will also include the following chapters:

Chapter 2: It will reveal the concepts of guarantee and Online payment system and how they work.

Chapter 3: It will provide a comparison and overview of platforms similar to ours.

Chapter 4: will provide a review of how to apply the agile methodology in project management, focusing on system requirements and design and how to model our platform into several UML diagrams.

The **final chapter:** It relates to how our platform was created, the tools and programming languages used in its development, design, and the method of installing it, in addition to the system architecture of our platform, providing an evaluation of the platform's performance, in addition to screenshots of some of our platform's interfaces.

Chapter 1

Literature Review

1.1 Introduction

Escrow and virtual money transfers are among the most important concepts of our time, as these mechanisms are the basic foundation of the modern payment system . These mechanisms provide an important security guarantee that allows individuals and institutions to carry out financial transfers with confidence and ease.

In this context, secure service provisioning platforms offer an innovative and reliable solution to facilitate money transfers and provide a secure environment for users.

This chapter will explore the concepts of guarantee and virtual money transfers.

1.2 Overview About Escrow

1.2.1 What Does Escrow Mean

In financial transactions, the term **"in escrow"**refers to the temporary placement of an item, such as money or property, where a third party holds the money or assets and arranges their payment on behalf of the parties involved in the transaction.

"In escrow" is a type of legal holding account for things that cannot be released until predetermined conditions are met. The purpose of using an escrow is usually to ensure that both parties fulfill their obligations as agreed upon before the transaction is completed.[1]

1.2.1.1 How Does it Work

Escrow functions as a secure and impartial intermediary between transaction-involved parties, such as buyers, sellers, or borrowers, ensuring a smooth and equitable transaction process.

Here's a breakdown of how it typically operates:

1. **Agreement:** The involved parties reach an agreement to utilize an escrow service within their transaction. This agreement is typically documented in a contract or agreement that delineates the terms and conditions of the escrow arrangement.
2. **Deposit:** The buyer places funds or assets into an escrow account held by an impartial third party, often an escrow agent or company.
3. **Verification:** The escrow agent validates that the agreement's conditions are met, including verifying the delivery of goods or services as specified.
4. **Release:** Upon fulfilling all conditions, the escrow agent releases the funds or assets to the appropriate party as per the agreement's stipulations.

1.2.2 Types of Escrow

Different types of escrow arrangements are available, each tailored to specific transactions or industries.

Here are examples of common escrow types:

1. **Online escrow:** services are becoming increasingly popular due to the rise of online transactions. These services function as intermediaries in online sales by securely holding funds until buyers confirm receipt of goods or services. This helps reduce fraud risks and promotes trust in online business dealings.
2. **Real estate escrow:** involves placing funds, documents, and related items for property sales into an escrow account. The escrow agent holds these items until all sale conditions are met, such as inspections, financing, and legal matters. Once conditions are satisfied, the escrow agent releases funds, completing the ownership transfer.
3. **Mortgage Escrow:** Lenders have the option to mandate borrowers to create a mortgage escrow account to cover expenses like property taxes, homeowners insurance, and potentially mortgage insurance payments. The lender assumes responsibility for managing

this account and making payments on the borrower's behalf as needed.

4. **Business escrow:** is utilized in mergers, acquisitions, and commercial transactions. It encompasses depositing funds, contracts, and documents into an escrow account, where they remain until all transaction conditions are met. This process ensures that both parties fulfill their obligations before finalizing the deal.

1.2.3 Advantages and Disadvantages of Escrow

Advantages and disadvantages of using escrow services can vary based on specific circumstances and the nature of the transaction.[2]

Here are some general advantages and disadvantages:

1. **Advantages of Escrow**
   - 1.1 **Payment Protection:** For buyers, escrow offers protection by ensuring that payment is only released once they receive the goods or services as agreed.
   - 1.2 **Security:** Escrow ensures transaction security by holding funds or assets until all terms and conditions of the agreement are met.
   - 1.3 **Trust:** Escrow fosters trust among transaction parties, particularly in situations where they have limited familiarity with each other.
2. **Disadvantages of Escrow**
   - 2.1 **Cost:** Using escrow services usually incurs fees, which can add to the overall transaction costs.
   - 2.2 **Processing Time:** Escrow transactions may take longer to complete compared to direct transactions, as the escrow agent verifies and processes the necessary documents and conditions.
   - 2.3 **Potential for Escrow Disputes:** There is a potential for disputes in escrow arrangements, especially concerning the release of funds or assets, when parties disagree about meeting conditions or if one party violates the agreement.

1.3 Overview About Online Payment System

1.3.1 What is Online Payment System

An online payment system is a digital network or infrastructure that facilitates the transfer of funds between buyers and sellers in electronic transactions. This system enables users to make payments and receive money electronically, without the need for paper money or paper checks. These online payment systems revolutionize the way people shop and provide commercial services, while providing speed, convenience and flexibility.[3]

Key components of online payment systems include:

1. **Payment Gateways:** Payment gateways are interfaces that connect the online platform with the banking system, ensuring secure and reliable transactions. They authenticate and process payment requests from customers.
2. **Payment Methods:** Online payment systems support various payment methods, such as credit and debit cards, bank transfers, and digital currencies.
3. **Security:** Protecting financial transactions and user data is a priority for online payment systems. Technologies such as encryption, SSL certificates, and secure tokenization are commonly used.
4. **Compliance:** Online payment systems must adhere to regulatory standards, such as PCI DSS (Payment Card Industry Data Security Standard), AML (Anti-Money Laundering), and KYC (Know Your Customer) requirements.

1.3.2 Virtual Money Transfers

Virtual money transfer is the electronic movement of funds from one account to another over the internet. It offers a quick and efficient way for individuals and businesses to transfer money across various locations.

1.3.2.1 How Virtual Money Transfers Work

Virtual money transfers, also known as electronic funds transfers (EFT), encompass a wide range of methods for electronically transferring funds from one entity to another.

Here's a general overview of how these transfers typically work:

1. **Selection of Platform or Service:** Users select a virtual currency transfer platform based on their requirements, preferences, and available features like international transfers, low fees, and user-friendly interfaces.
2. **Account Establishment:** Users establish an account on the chosen platform or service by providing personal details, verifying their identity, and linking their bank account, debit card, or credit card for funding transfers.
3. **Initiating Transfer:** To initiate a transfer, users log in to their account on the virtual currency transfer platform or app, input the recipient's details (e.g., name, email address, bank account information), and specify the transfer amount.
4. **Verification and Confirmation:** Users may need to verify their identity or confirm transfer details through security measures like passwords, PINs, biometrics, or two-factor authentication (2FA).
5. **Processing of Transaction:** The virtual currency transfer platform processes the transaction, debiting the specified amount from the user's funding source and crediting it to the recipient's account or virtual wallet.
6. **Notification and Confirmation:** Both the sender and recipient receive notifications and confirmation of the completed transfer, including transaction specifics such as the transferred amount, exchange rates (for international transfers), and transaction IDs.

1.3.2.2 Reasons for the Prevalence of Virtual Currency Transfers

1. **Convenience:** Virtual currency transfers offer unparalleled convenience, enabling users to send and receive money from anywhere with an internet connection, without the need to visit physical bank branches.
2. **Speed:** Normally, transfers through virtual currency transfer services get processed within a short time. Most of the transfers get processed within minutes or instantly. This helps users in getting quick access to their funds.
3. **Diverse Options:** There are many virtual money transfer methods, such as peer-to-peer transfers (e.g., Venmo, PayPal), ACH transfers, and wire transfers. Each method offers varying levels of speed, convenience, and cost.
4. **Cost-Effectiveness:** Most virtual currency transfer platforms offer competitive prices

and lower fees in contrast to traditional banks (especially for international transfers), which saves users money on transaction charges.

5. **Global Reach:** Virtual currency transfer platforms operate globally, enabling users to send money internationally easily, overcoming geographical and currency barriers.
6. **User-Friendly Interfaces:** These platforms feature user-friendly interfaces, intuitive navigation, and mobile accessibility, enhancing the overall user experience.
7. **Security Measures:** Reputable virtual currency transfer services employ robust security measures like encryption, fraud detection, identity verification, and secure authentication methods to safeguard user accounts and transactions.

1.3.3 E-Wallets

E-wallets, or digital wallets, are software applications that allow users to store, manage, and use their financial information electronically. They have become increasingly popular due to their convenience and versatility.

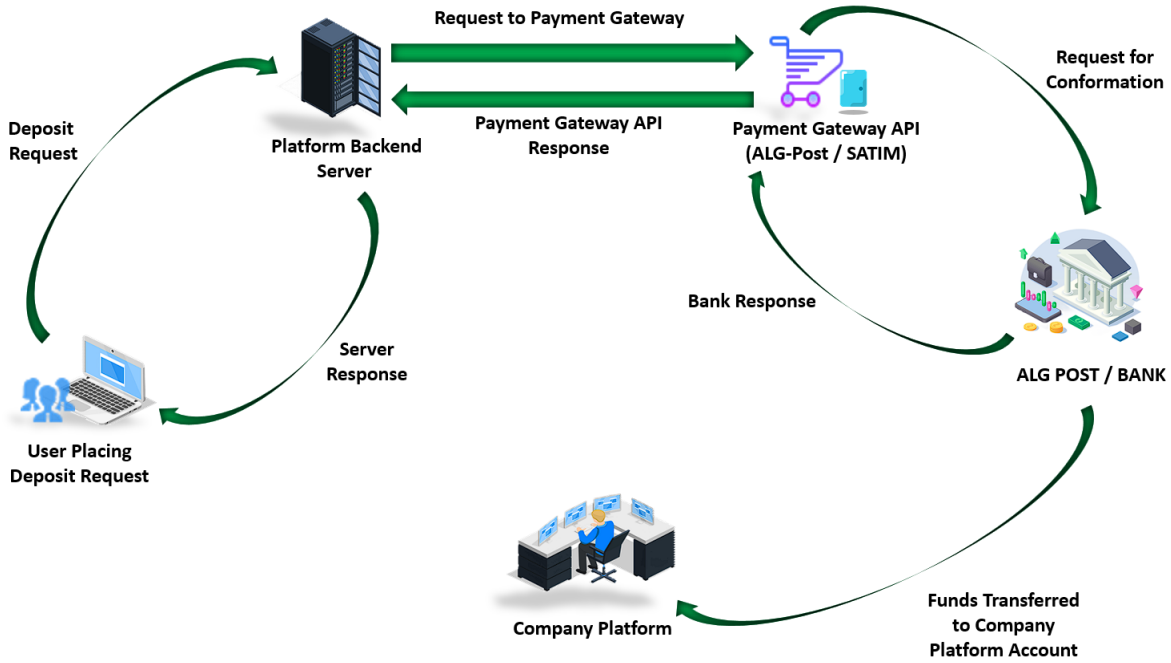


Figure 1.1: An Example of Depositing Money Process

1. **Types of E-Wallets:** There are different types of e-wallets, such as closed, semi-closed, and open, offering various levels of flexibility in terms of usage and access to funds.
2. **Functionality:** E-wallets store payment methods such as credit and debit cards, bank

account information, and even digital currencies. Users can make payments, transfer funds, and conduct other financial activities directly from the app.

3. **Integration:** Many e-wallets integrate with banks and other financial institutions, making it easy for users to link their existing accounts and manage their finances within the app.
4. **Security:** E-wallets prioritize user security through encryption, secure authentication (e.g., bio-metrics), and other measures to prevent fraud and unauthorized access.
5. **Uses:** E-wallets are versatile tools that can be used for online shopping, bill payments, peer-to-peer transfers, and more.

1.4 Conclusion

The first chapter has provided a thorough overview of Escrow and Online Payment Systems, highlighting their critical roles in ensuring trust, security, and efficiency in platforms for providing secure services. Escrow acts as a neutral party, ensuring obligations are met before funds are released, while Online Payment Systems facilitate seamless financial interactions. Understanding these pillars enables secure service provisioning platforms to offer a transparent, reliable.

Chapter 2

Peer-to-Peer Platforms and Escrow: A Comparative Overview

2.1 Introduction

Escrow services play a vital role in ensuring secure transactions within peer-to-peer platform . By utilizing escrow, both parties can engage in transactions with confidence, knowing that the funds are held securely until the agreed-upon conditions are met. This introduction will delve into the benefits of using escrow in service provider agreements, highlighting its role in fostering trust, mitigating risks, and facilitating smooth and transparent transactions.

2.2 Peer-To-Peer Service Platforms

2.2.1 Definition

A peer-to-peer (P2P) service platform is an online space where people or businesses can connect directly to trade goods, services, or information. Unlike traditional setups with middlemen like companies, P2P platforms let participants be both service providers and customers, forming a decentralized network.

Here's how a typical P2P service platform operates:

- **User Registration:** Users log in to the platform and create profiles that include service introductions, personal information, skills, and other details.

- **Service Listings:** Users can create lists or posts for the services they offer. These listings include a description of the service, price, and location.
- **Search and Discovery:** Users can search for the services they need using filters such as price and location, and they can browse the listings to find the service provider that suits them.

2.2.2 Examples of Peer-to-Peer Service Platform Around the World

2.2.2.1 Freehali

- **What is Freehali**

  Freehali® is a freelance services platform in Algeria that connects skilled individuals looking to monetize their talents with entrepreneurs seeking reliable services at affordable rates.Users can freely choose and negotiate with multiple freelancers simultaneously without any obligations. The platform offers a wide range of services such as graphic design, programming, translation, content writing, social media management, online courses, video editing, voice-overs, digital marketing, advertising campaigns, and web hosting.

  Freehali® acts as a trustworthy intermediary, ensuring clients receive reimbursement if the work doesn't meet predefined standards and ensuring freelancers get paid for meeting these standards.[4]

- **How Does it Work**

  The Freehali website provides users with listing requirements, setting a budget and the time needed to complete the work, allowing the interviewer and negotiator with freelancers to choose the best offers, as well as the payment feature in the application "baridiMob".[5]

2.2.2.2 FareNow

- **What is FareNow**

  It is a digital marketplace that connects users with freelancers and provides many different tasks, including manual labor, craft work, and transportation and more.[6]

- **How Does it Work**

It allows users to browse the list of available freelancers, search for available services, and book tasks. They can also choose between the subscription service and the hourly service.[6]

2.2.2.3 Upwork

- **What is Upwork**

  Upwork serves as an online platform that facilitates the connection between freelancers, independent professionals, and clients seeking their services. It offers a means for businesses and individuals to discover and engage talent across diverse projects like web development, graphic design, writing, marketing, and other specialized fields.[7]

  Upwork equips users with communication tools, project management resources, and streamlined payment processing systems to enhance the efficiency and effectiveness of collaborations between clients and freelancers.

- **How does it work**

  Upwork assists expanding businesses in discovering and collaborating with exceptional talent and agencies. Starting is straightforward: sign up, post your project, and review proposals to identify the ideal independent contractor or agency for your needs. After engaging your talent, utilize the platform's tools to communicate, collaborate, and securely pay for your project.[8]

  Understanding how Upwork operates is best explained in three phases:

  1. Preparation before hiring
  2. Discovering and engaging talent
  3. Executing the work

  Throughout this journey, the process may vary slightly based on the plan you select: Upwork Basic, Plus, Business, or Upwork Enterprise.

2.2.3 Comparison of Popular Peer-to-Peer Platforms:

The next table provides a comparison of popular P2P platforms: Upwork, freehali and fareNow. The comparison includes several key factors that are important for freelancers and

clients, such as the types of services offered, fee structures, user base, and other features. By examining the similarities and differences between these platforms, it becomes possible to gain a better understanding of their respective strengths and weaknesses, and to make an informed decision about which platform may be the best fit for one's individual needs:

	Upwork	**FareNow**	**Freehali**
Year Established	2013	2020	2017
Number of Users	Over 16 million	Unknown	100.000
Geographic Focus	Global	Global	Algeria
Project Types	All types	All types	Only Arabic
Payment Methods	Bank transfer, credit/debit card, PayPa	Bank transfer, credit/debit card, PayPal, local payment methods	BaridiMob and ccp

	Upwork	FareNow	Freehali
Fees	20% for the first $500m billed with a client.10% for lifetime billings with clients between$500 $10,000.5% for billings with clients that exceed $10,000	Unknown	Unknown
Quality Control	Screening process for freelancers, work monitoring tools, dispute resolution system	Unknown	Verification system for freelancers, work monitoring tools
Dispute Resolution	Arbitration for disputes between clients and freelancer	0.90	Dispute resolution system
Communication	n-app messaging system and video conferencing tools	In-app messaging system	In-app messaging system

	Upwork	FareNow	Freehali
Mobile App	iOS and Android	iOS and Android	Not Supported
Additional Features	Time tracker, collaboration tools, integration with popular project management tools, premium membership	buyer request feature	Local currency support, buyer request feature

Table 2.1: Comparison of The Most Popular Peer-to-Peer Platforms for Freelancers and Service Providers.

Upwork, Freehali, and FareNow are popular online freelance marketplaces that connect clients with freelancers from various fields. While they share some similarities, there are also some notable differences among them.

Upwork is a platform that offers a more diverse range of project types, including both hourly and fixed-price projects [9].It also provides more advanced tools and features for managing and tracking projects, such as time tracking software and dispute resolution services.

FareNow, on the other hand, a more diverse range of services , including Barber Shop ,Central Air Conditioning, Cleaning Services ,Electrical Services,Graphics and Design and fixed-price projects .

Freehali is a specail platform for algerians that focuses on providing a wide range of services, including creative work, digital marketing, writing, and programming [10].It emphasizes fixed-price projects and offers a streamlined, user-friendly interface

In summary, while all three platforms offer opportunities for freelancers to find work and for clients to hire them, each has distinct differences that cater to different needs and preferences.

The selection of Upwork, Freehali, and FareNow for comparison in the graduation disserta-

tion is justified for several academic reasons. Firstly, these platforms are considered freelancing platforms with the same features found on our platform (peer-to-peer).

Secondly, comparing these platforms allows for an analysis of their unique features, business models, and effectiveness, providing valuable information for individuals seeking freelance work or employers looking to hire freelancers.

Lastly, studying these platforms helps understand the impact of online freelancing on the economy and the future of work.

2.2.4 Advantages and Disadvantages of Peer-To-Peer Service Platform

1. **Advantages of Peer-To-Peer Service Platform**

   1.1 **Transaction Facilitation:** This platform facilitates the transaction process, such as payment processing and scheduling. It offers different payment methods, such as direct payments, escrow services, or integrated payment systems.

   1.2 **Direct Contact :** Once the user finds the service provider he is looking for, he can communicate with him via messages or contacts provided by the platform, and this gives the possibility to ask questions and negotiate details.

   1.3 **Cost-Effective:** peer-to-peer platforms usually reduce costs for both service providers and consumers by eliminating intermediaries

   1.4 **Flexibility:** Users have the freedom to choose the service provider that helps them based on price, location and needs because it offers a wide range of service providers.

   1.5 **Increased Options:** Users have access to a wide range of services and providers on peer-to-peer platforms, providing more choices and flexibility in selecting services that meet their specific needs.

   1.6 **Transparency and Accountability:** Rating and feedback systems on P2P platforms promote transparency and accountability, helping users make informed decisions and maintain trust within the community.

2. **Disadvantages of Peer-to-Peer Service Platform**

   2.1 **Trust and Safety Concerns:**Trust and Safety Concerns: Trust is essential in peer-to-peer platforms, especially when dealing with services that involve personal interactions or access to private spaces (e.g., home services). Ensuring the safety of

users and their belongings can be a significant challenge.

2.2 **Reliability and Availability:** Service availability can be inconsistent, as providers may have varying schedules or may not always be reliable. This can lead to difficulties in booking services when needed or in emergencies.

2.3 **Dependence on Technology:** Peer-to-peer platforms rely heavily on technology for matchmaking, booking, payments, and communication. Any technical glitches, cybersecurity threats, or platform outages can disrupt the user experience.

2.4 **Dispute Resolution:** Resolving disputes between users and providers can be complex, especially when there are disagreements over service quality, pricing, or other issues. Peer-to-peer platforms need robust mechanisms for handling disputes fairly and efficiently.

2.5 **Limited Service Offerings:** Peer-to-peer platforms may not offer the same range of services as traditional service providers or may lack specialized services. This can be a limitation for users with specific needs.

2.3 Conclusion

In conclusion, Peer-to-Peer (P2P) service platforms revolutionize connections between individuals and businesses for goods, services, and information exchange. By enabling direct interactions, these decentralized networks offer cost-effectiveness, flexibility, and increased options. Examples like Freehali, FareNow, and Upwork showcase the varied applications and models of P2P platforms globally. Despite their advantages, they face challenges in trust, reliability, and technology dependence. Overall, P2P platforms significantly impact the future of freelance work, highlighting the importance of direct and flexible service exchanges.

Chapter 3

System Requirements and Design

3.1 Introduction

The ongoing evolution of technology has made providing security services over the Internet a crucial aspect of modern society. Employing the Agile project management methodology is essential for the success of digital endeavors, such as our security services platform.

This chapter aims to review how to apply the agile project management methodology, based on the data we have about the system needs and design requirements. focusing on functional and non-functional requirements. It also touches on system design principles and technical foundations essential for platform effectiveness and security.

The goal of this chapter is to gain a comprehensive understanding of the framework needed to implement our project successfully, thereby delivering innovative and secure online security services to our users.

The Best Practices for Designing and Implementing

For the optimal design and implementation of the Service Providers platform, it is advisable to adopt a user-friendly interface akin to those found on popular platforms like Facebook and Upwork. This approach will not only enhance the user experience but also attract a greater number of users to the platform. To prevent clients and Service Providers from conducting business off the platform and thereby evading platform fees, it is imperative to implement effective measures aimed at combating off-platform transactions.This objective can be accomplished by instituting a clear policy that explicitly prohibits off-platform dealings and imposing penalties

for non-compliance. Furthermore, the implementation of secure payment methods that facilitate transactions exclusively on the platform will serve as an incentive for users to remain on the platform and conduct their business securely. In essence, the design and implementation of a freelance platform that embraces best practices and places emphasis on user experience and security will foster the growth and long-term sustainability of the platform.

3.2 Project Management Methodology (Agile)

The Agile methodology, developed by a group of software developers in 2001[11], is an iterative and flexible approach to project management and software development. It emphasizes collaboration, adaptability, and customer feedback with the goal of delivering value to customers more efficiently and effectively than traditional linear approaches. This approach prioritizes working software over extensive documentation, customer collaboration over contract negotiation, and adapting to change over strictly following a plan[12].

3.2.1 Details of Agile Methodology

Agile methodology is distinguished by its flexibility and adaptability, enabling teams to promptly respond to changing requirements and customer needs. It fosters constant feedback and iteration to enhance the product. Agile methodology employs short development cycles, typically lasting two to four weeks, known as sprints, to deliver working software incrementally [13]. Within each sprint, the team concentrates on delivering a set of features or functionality that can be demonstrated to customers and stakeholders for feedback .

One of Agile methodology's fundamental principles is self-organizing and cross-functional teams. These teams consist of individuals with diverse skills and backgrounds who collaborate towards a common objective. They are authorized to make decisions and adjust to changes without formal management approval [14]. Agile methodology also stresses continuous improvement. At the end of each sprint, the team conducts a retrospective meeting to review successes, areas for improvement, and actions for the next sprint. This process enables ongoing learning and enhancement of performance .

Overall, Agile methodology is a customer-centric, collaborative, and adaptable approach to project management that prioritizes working software and continuous improvement. It has gained widespread acceptance in the software development industry and beyond, proving ef-

fective in delivering high-quality products promptly [15].

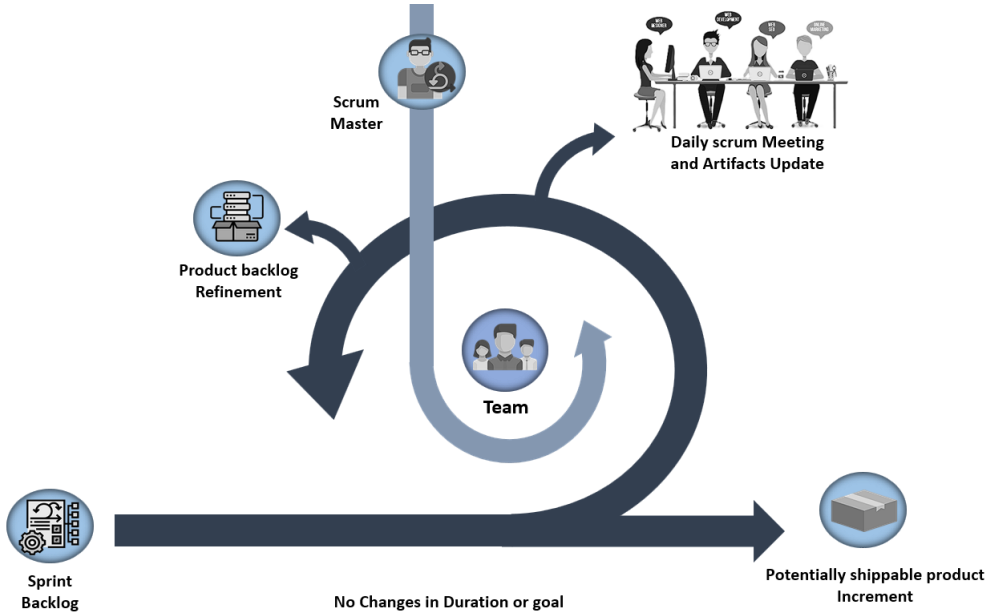


Figure 3.1: An example of agile software development methodology: Scrum

3.2.2 Applying agile methodology

Here we discuss how to apply agile methodology in our project:

1. **Form an Agile Team:** we began by assembling a cross-functional team, including developers, designers, and other necessary roles, and assigning clear responsibilities to each individual in the team within the project.
2. **Define the Vision and Goals:** We clearly defined the project's vision, achievable goals, and objectives, and made sure that every member of the team understood and aligned with these goals.
3. **Create a Product Backlog:** We have established a prioritized list of features and tasks known as the Product Backlog and keep it regularly updated based on feedback and changing requirements.
4. **Sprint Planning:** We break down the product backlog into smaller tasks that can be completed within a short time frame. This is discussed during sprint planning meetings, where the team decides what to work on in the next sprint.
5. **Review and adjust:** We hold daily meetings for a maximum of 30 minutes where we discuss progress, challenges, and plans for the day, collect feedback. This helps everyone

get along and know the progress of a project.

6. **Develop and test iteratively:** We applied an agile approach to continuous integration and testing for feature development. We conducted numerous tests and ran them repeatedly, enabling us to detect problems early on.

We have applied the Agile methodology in our work because it encourages continuous improvement and constant communication among team members, which facilitates and accelerates the development process.

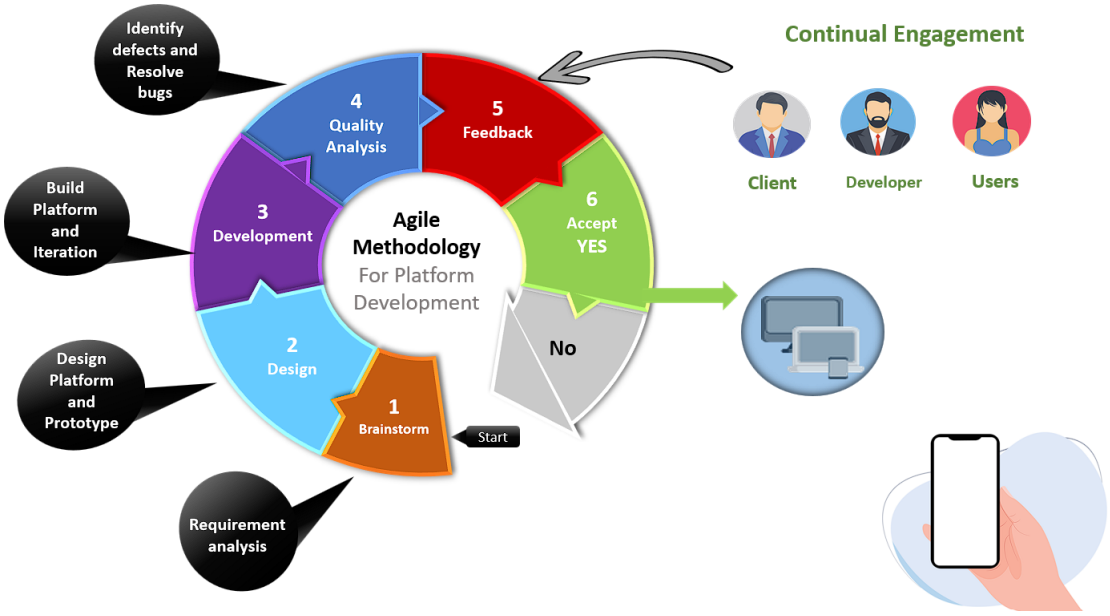


Figure 3.2: Agile Methodology For Platform Development

3.3 Non-Functional Requirements

Non-functional requirements define the qualities or characteristics that a system must possess, such as performance, security, usability, scalability, and reliability. These requirements focus on how the system should behave under various conditions and constraints, rather than specific functionalities.

Our platform comprise the following Non-Functional Requirements:

1. **Security:** The platform must contain strong measures to protect the database from unauthorized access, breaches and thefts, thus ensuring user confidence and confidentiality.
2. **Privacy:** One of the most important things a platform must have is ensuring user privacy and protecting their personal and financial information.

3. **Reliability:** Freelancers and clients need to be able to access the platform reliably, any time they need it. This means the platform should always be working smoothly and consistently.
4. **Performance:** The platform should prioritize user experience by delivering rapid response times and minimizing downtime, to provide an optimal user experience and optimize user satisfaction.
5. **Scalability:** The platform must be designed to seamlessly handle a growing user base while accommodating a large number of projects without compromising performance
6. **Maintainability:** Design with ease of maintenance in mind; the platform should be effortlessly updatable and maintainable over time.
7. **Usability:** Intuitiveness is paramount; the platform must be easy to navigate for users of all proficiency levels, fostering accessibility and engagement.
8. **Accessibility:** Every user, regardless of ability, should access the platform effortlessly, adhering to relevant accessibility standards for inclusivity.
9. **Compatibility:** Seamlessness across devices and browsers is essential, accommodating users with diverse technological setups for enhanced accessibility.
10. **Data backup:** Regular data backups are imperative to prevent loss during system failures or disasters, safeguarding user information and integrity.
11. **Monitoring:** Continuous monitoring ensures proactive detection and resolution of potential issues, maintaining service quality and reliability.
12. **Support:** Adequate resources and support channels should be available to assist freelancers and clients promptly, fostering user satisfaction and loyalty.

3.4 Functional Requirements

Functional requirements, on the other hand, delineate the specific functionalities or features that a system must provide to its users. They specify what actions the system should be capable of performing and how it should respond to particular inputs or events.

Our platform has these essential Functional Requirements:

1. **User Management:** Allows users to create and edit profiles, providing the necessary

information for effective platform interaction (e.g., contact details, preferences, qualifications).

2. **Search and Discovery:** Enables users to efficiently find relevant offers and services through powerful search and filtering tools. This includes features like nearby search for mobile users using GPS, ensuring a smooth user experience.
3. **Secure Transactions:** Facilitates secure and convenient transactions between users through an integrated payment system, ensuring financial security and peace of mind.
4. **Safeguarding User Interactions**
    - **P2P Escrow System:** Safeguards transactions by holding funds in escrow until both parties fulfill their obligations, minimizing risk and fostering trust between users.
    - **Dispute Resolution Process:** Implements a robust mechanism to address user disagreements or issues arising from platform interactions, ensuring a fair and timely resolution process.
5. **Enhancing the User Experience**
    - **Built-in Messaging System for Communication:** Allows users to communicate directly with each other through a convenient messaging system, streamlining collaboration and communication.
    - **Notification System:** Informs users of relevant platform activity, such as incoming messages, new offers, or important updates, promoting user engagement and timely action.
    - **Review and Feedback System for Improvement:** Enables users to provide and receive feedback, fostering continuous improvement and a positive user experience.
6. **Data and Insights**
    - **Comprehensive Reporting and Analytics Tools:** Offers comprehensive reporting and analytics tools, allowing users to gain valuable insights into platform usage and user behavior for informed decision-making.
    - **Invoicing and Billing:** Generates clear and accurate invoices or billing statements for transactions, ensuring transparency and clear financial records.
7. **Optional Functionality**

- **Time Tracking (if applicable):** (Optional) Allows users to track time spent on tasks or projects relevant to the platform's functionality, providing valuable data for project management and resource allocation.

3.5 System Design

3.5.1 System Modeling

We utilized UML diagrams, including use case, class, and sequence diagrams, to offer a comprehensive visual depiction of our software system from various viewpoints.

UML (Unified Modeling Language) . It is a standardized modeling language used in software engineering to visually represent systems, designs, and processes. UML provides a set of graphical notations and diagrams (use case, class, sequence) that help software developers and stakeholders communicate and understand the structure, behavior, and interactions of a system.

To produce these UML diagrams, StarUML, a software application specializing in UML-based modeling and diagramming was utilized. StarUML facilitates a user-friendly interface for the creation, modification, and examination of UML diagrams, encompassing an extensive array of functionalities. These functionalities include assistance for different UML diagram categories, model verification, code syndissertation, and extension through plugins [16].

3.5.1.1 Description based on roles (Actors/Roles).

Actor	Functionalities
Visitor	• **Login or Signup:** Access the platform by creating a new account or logging into an existing one. • **Browse and Search:** Explore available service offers through a user-friendly interface.

Actor	Functionalities
Client	• **Request Job Services:** Initiate service requests from providers. • **Job Management:** Initiate, monitor, and follow up on job statuses. • **Real-time Messaging:** Communicate with service providers to discuss job details. • **Dispute Resolution:** Raise and resolve disputes in case of conflicts. • **History and Notifications:** Access past and ongoing job details, receive real-time notifications. • **E-Wallet System:** Manage finances, including balance checks, deposits, withdrawals, and transaction history. • **Profile Editing:** Customize profile information and preferences.
Service Provider	• **Service Offer Creation**: Post service offers with detailed descriptions. • **Offer Management:** Edit, delete, or refresh posted offers. • **Professional Profile:** Showcase skills, experience, and portfolio for client review.

Actor	Functionalities
Moderator(Support Agent)	• **Dispute Management:** Mediate and resolve disputes between clients and providers. • **Job Oversight:** Monitor ongoing jobs, access messages, and track job statuses. • **Dual Role Capability:** Perform actions both as a client and a service provider.
Admin	• **Moderator Management:** Add or remove moderators for dispute resolution and oversight. • **Global Notifications:** Send real-time notifications to all platform users.

Table 3.1: Roles and Functionalities for Different Actors

3.5.1.2 Use Case Diagram

- **Definition:** The use case diagram represents the structure of the major functionalities necessary for users of the system
- **Elements of a Use Case Diagram:**
  1. **Actor:** Represents users or external processes interacting with the system. Two types are:
     - **Main Actor:** Directly involved in the use case, seeking a specific result.
     - **Secondary Actor:** Provides additional information necessary for the use case.
  2. **Use Case:** Represents a specific functionality or action the system performs in response to an actor's request, typically shown as ellipses or ovals.
  3. **Relationships between Actors and Use Cases:** Show interactions between them, including:
     - **Association:** A simple interaction between an actor and a use case.

– **Inclusion/Extension:** One use case is integrated into or extended by another use case.

3.5.1.3 Khdemni Use Case Diagram

The following figure represents the general use case diagram of our platform.

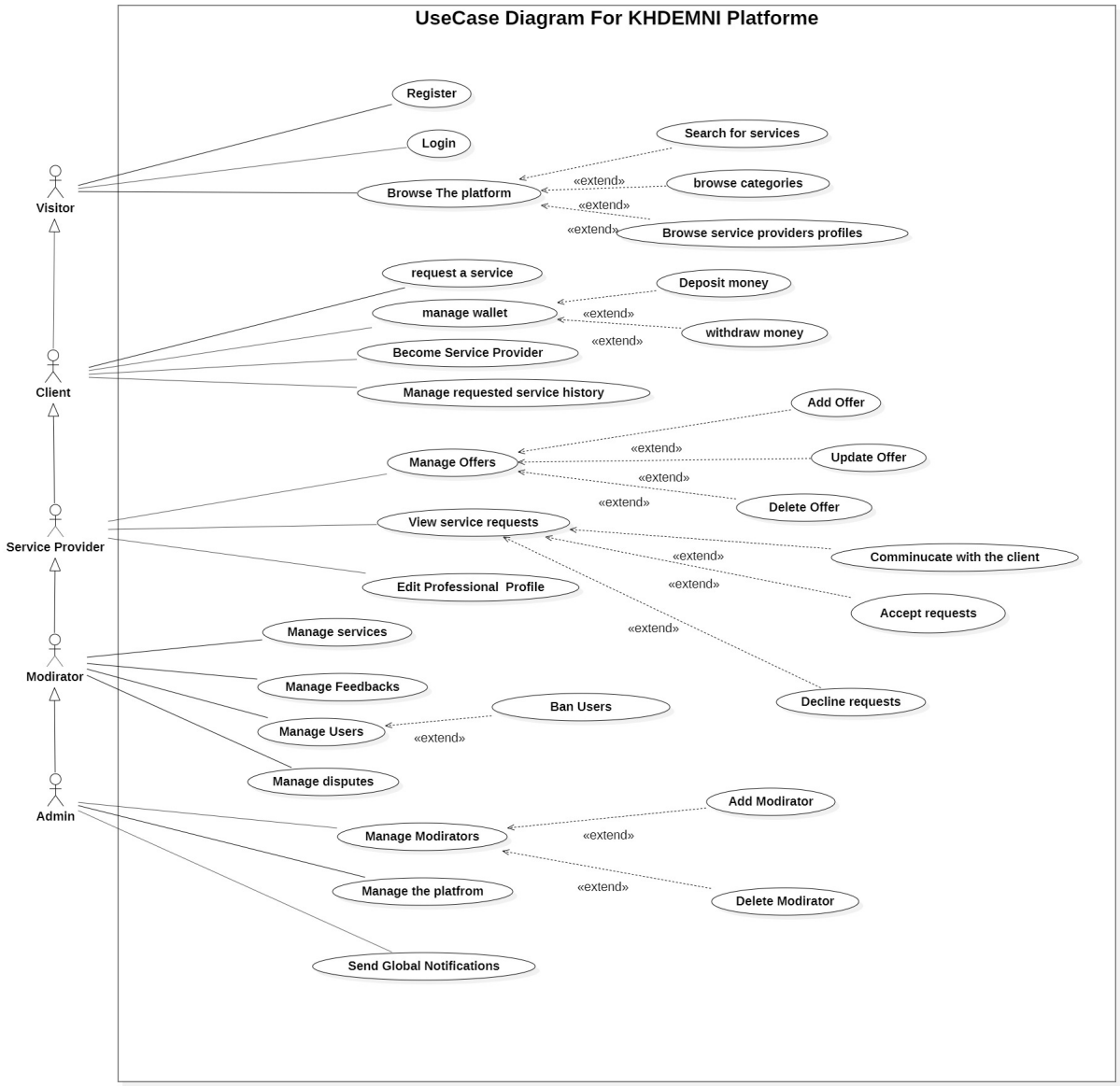


Figure 3.3: khdemni use case diagram

3.5.1.4 Text description of use cases

A CU use case highlights the functional relationships between the actors and the system being studied.

Preconditions: Define the conditions that must be met for the CU to start.

Post-conditions: Define the conditions that must be true when the CU completes successfully, whether in a nominal or alternative scenario.

1. **«Request a service» use case**

   The table shows the use case of how to request a service and displays the different actors who can access this service

Identification
Use case name: Request a Service. **Purpose:** Request a Service. **Actors:** Costumer , Service provider
Precondition • The client is connected to the platform. • The customer searched for and selected a specific service. • The customer has to check the details and availability of the service for the desired date and time. **Nominal sequences:** • The customer selects the desired service from the available options. • The customer chooses the desired date and time for the service. • The Customer checks the reservation details and confirms the request. • The system records the reservation and sends a notification to the relevant service provider. The service provider confirms availability and accepts the reservation. • The customer receives a reservation confirmation with details **Alternative sequences:** If the service provider is not available or refuses the request, the system informs the customer and suggests selecting another service provider. **Post-conditions:** 1. The service is successfully requested. 2. The service provider is notified of the request. 3. The customer receives a reservation confirmation. **Variants:** The customer can choose a quick booking option which assigns them the first available provider in their region for the selected service

Figure 3.4: Table 1. “Request a service” use case

2. **«View service requests» use case**

   The table shows the use case of how to view request a service and displays the different actors who can access this service

Identification
Use case name: View service requests. **Purpose:** View service requests. **Actors:** Service provider ,Costumer
Precondition • The service provider is connected to the platform. • The service provider has access to pending service requests. **Nominal sequences:** • The service provider views pending service requests. • The provider selects a service request for review. • The provider verifies the details of the request and service specifications. . • The provider accepts the request if they are available and willing to provide the service. • The provider start a chat with consumer to negotiate about the details of the request. • The system records the provider's decision and sends a notification to the customer. **Alternative sequences:** If the provider is not available or refuses the request, the system records the refusal and sends a notification to the customer. **Post-conditions:** • The service provider accepted or refused the service request. • The service can start a chat with the consumer who request the service. • The customer is notified of the service provider's decision. **Variants:** The provider can manage multiple pending service requests simultaneously, reviewing them one by one and making a decision on each.

Figure 3.5: Table 2. "View service requests" use case

3. **«Reserve a service» use case**

   The table shows the use case of how to Reserve a service and displays the different actors who can access this service

Identification

Use case name: Reserve a Service.

Purpose: Reserve a Service.

Actors: Client , Service Provider, System.

Precondition

- The client is logged into the platform.
- The client has selected a specific service.
- The service availability has been verified.

Nominal sequences:

- The client selects the desired service from the available options.
- The client chooses the desired date and time for the service.
- The client reviews the reservation details and confirms the request.
- The system verifies the service availability for the chosen date and time.
- The system records the reservation and sends a notification to the service provider.
- The service provider reviews the request and confirms availability.
- The system sends a confirmation of the reservation to the client with the service details.

Alternative sequences:

- Date/Time Unavailable If the chosen date or time is unavailable, the system notifies the client and suggests selecting a different time slot.
- Service Provider Declines Request If the service provider declines the request, the system notifies the client and suggests selecting a different service provider or time slot.

Post-conditions:

- The service is successfully reserved.
- The service provider is notified of the reservation.
- The client receives a confirmation of the reservation. Main Scenario

Variants:

Quick Reservation The client can choose a quick reservation option to be matched with the first available service provider in their area. Frequency of Use Frequently, whenever a client wishes to book a service on the platform.

Figure 3.6: Table 3. “Reserve service” use case

4. **«Manage a dispute» use case**

   The table shows the use case of how to Manage a dispute and displays the different actors who can access this service

Identification

Use case name: Manage Disputes.

Purpose: Manage Disputes.

Actors: Moderator , Client, Service Provider, System.

Precondition

- The moderator is logged into the platform.
- There is a dispute reported by either a client or a service provider.

Nominal sequences:

- The moderator views the list of reported disputes.
- The moderator selects a dispute to review.
- The moderator examines the details of the dispute, including messages, service details, and any attached evidence.
- The moderator contacts both the client and the service provider if additional information is needed.
- The moderator makes a decision based on the evidence and platform policies.
- The system updates the status of the dispute to resolved.
- The system notifies both the client and the service provider of the resolution.

Alternative sequences:

Additional Investigation Needed If the dispute requires further investigation, the moderator can put the dispute on hold and request additional information from either party.Escalation If the moderator is unable to resolve the dispute, it may be escalated to a senior moderator or manager for further review.

Post-conditions:

- The dispute is resolved.
- Both parties are notified of the resolution.
- The system records the outcome of the dispute.

Variants:

Automatic Dispute Resolution For certain types of disputes, predefined rules may allow for automatic resolution without moderator intervention. Frequency of Use As needed, whenever a dispute is reported on the platform.

Figure 3.7: Table 4. “Manage dispute” use case

3.5.1.5 Sequence Diagram

- **Definition:** Designed to illustrate how different system objects interact over time, show casing exchanged messages between objects. In our context, system sequence diagrams are employed to depict scenarios in which specific use cases are realized. These diagrams treat the system as a black box, focusing on the interaction between actors and the system to achieve the desired use case.

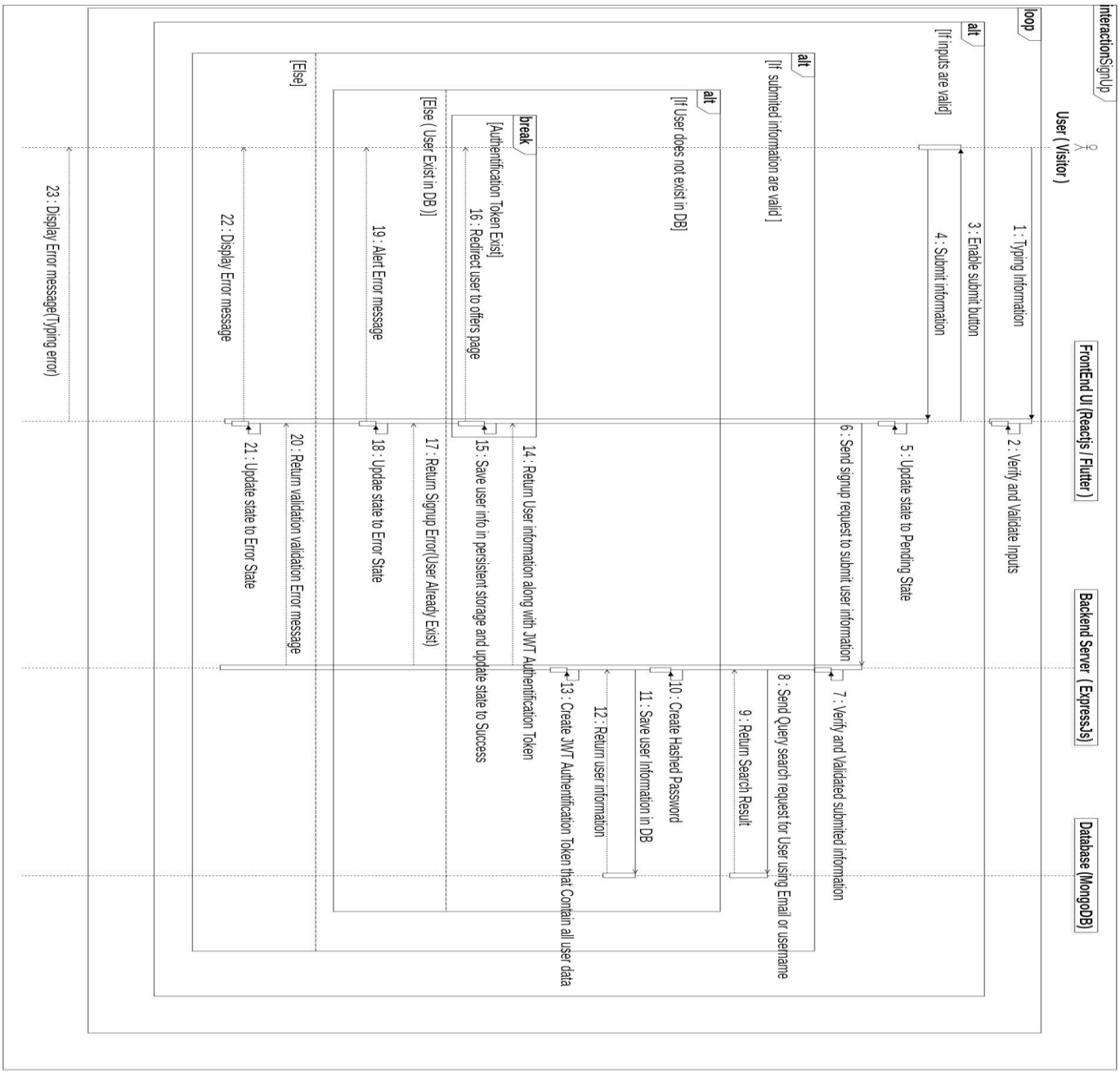


Figure 3.8: Sign-Up Sequence

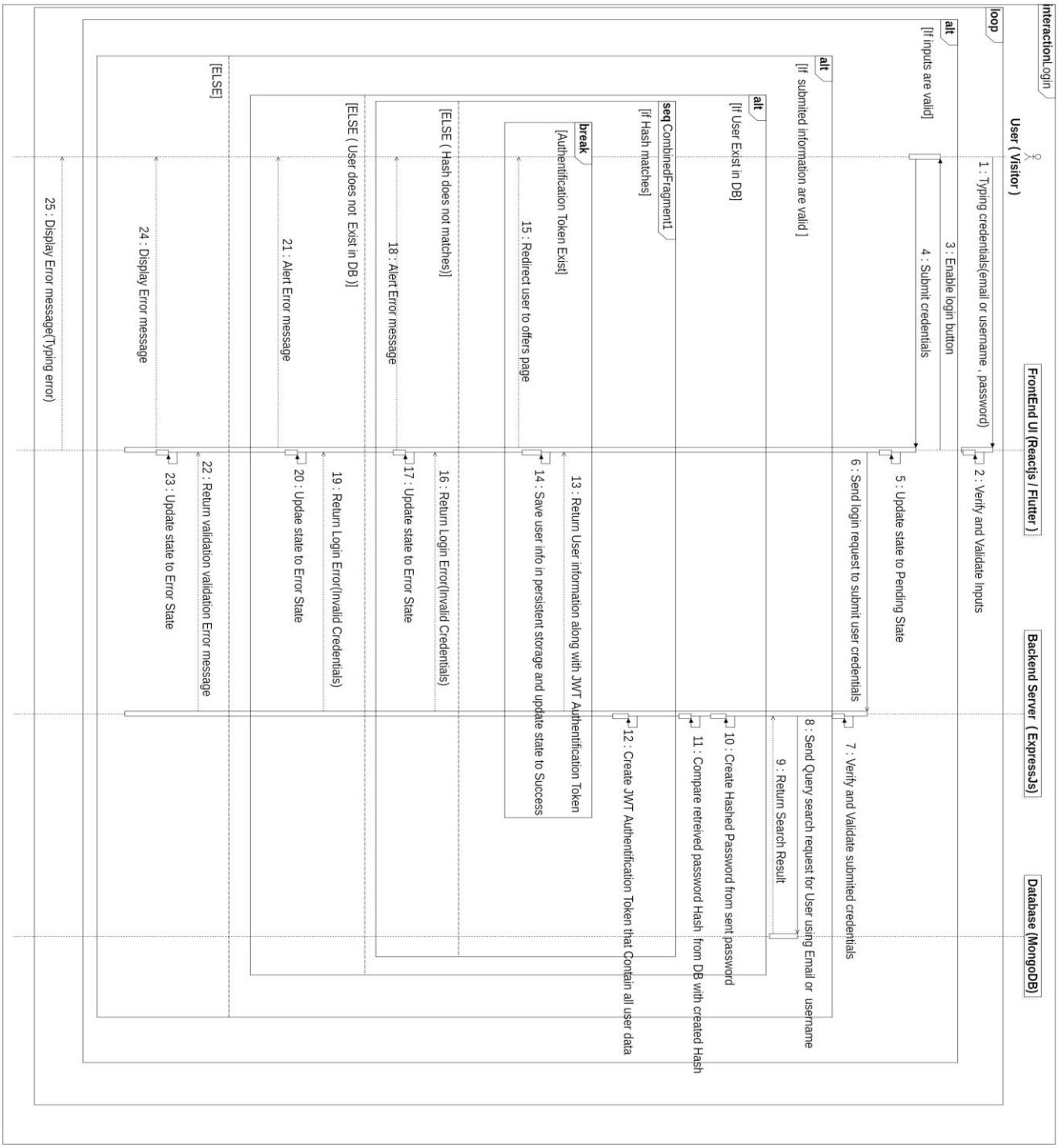


Figure 3.9: Login Sequence

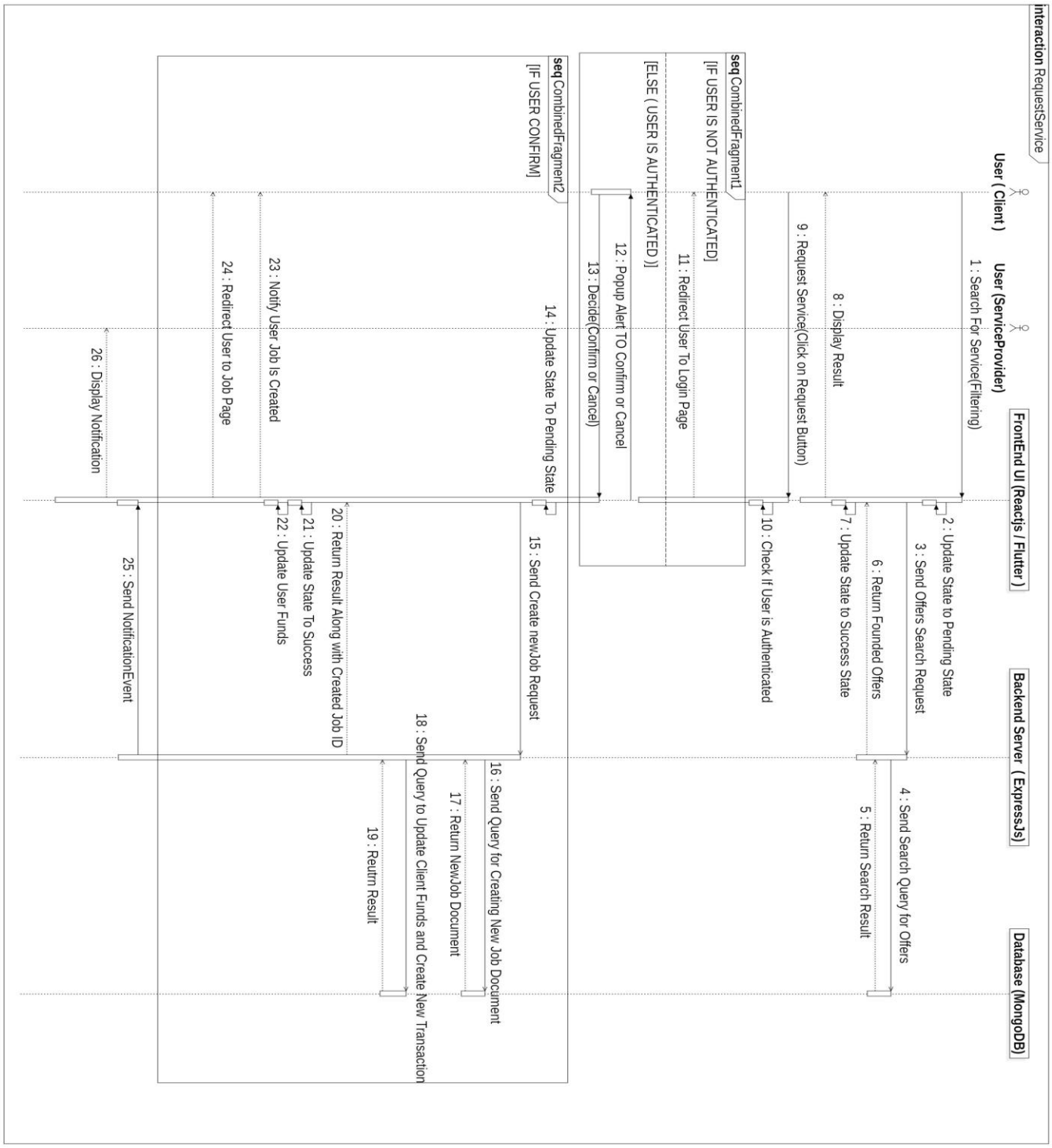


Figure 3.10: Request Service Sequence

3.5.1.6 Class Diagram

• **Definition:** Utilized for modeling the system's structure in terms of classes, their attributes, and the relationships between them.

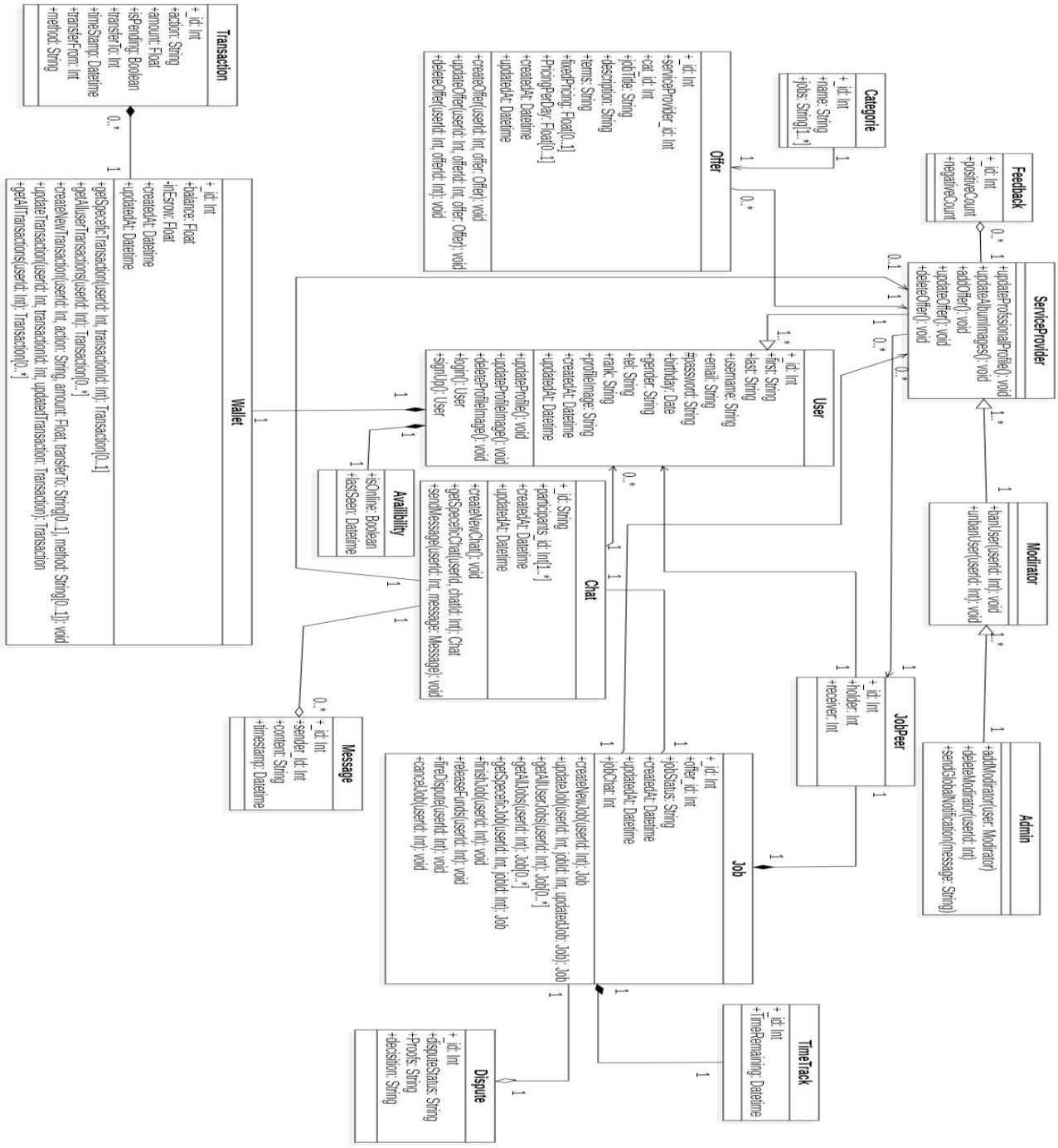


Figure 3.11: Class Diagram

3.5.2 Database Design

Database design represents a pivotal facet of system development, serving as the structural foundation of the system. It encompasses the intricate processes of conceptualizing, crafting, and administering the database, which encompasses the schema, tables, and their interconnections. The database design's efficacy, security, and dependability are paramount to safeguarding the system's operational efficiency and data integrity.This segment delineates the database design pertinent to the service providers platform. The database design methodology comprises several stages, encompassing the identification of data prerequisites, the formulation of a conceptual data model, its translation into a logical data model, and in its implementation within the physical database.[17]

3.5.2.1 Database Modeling

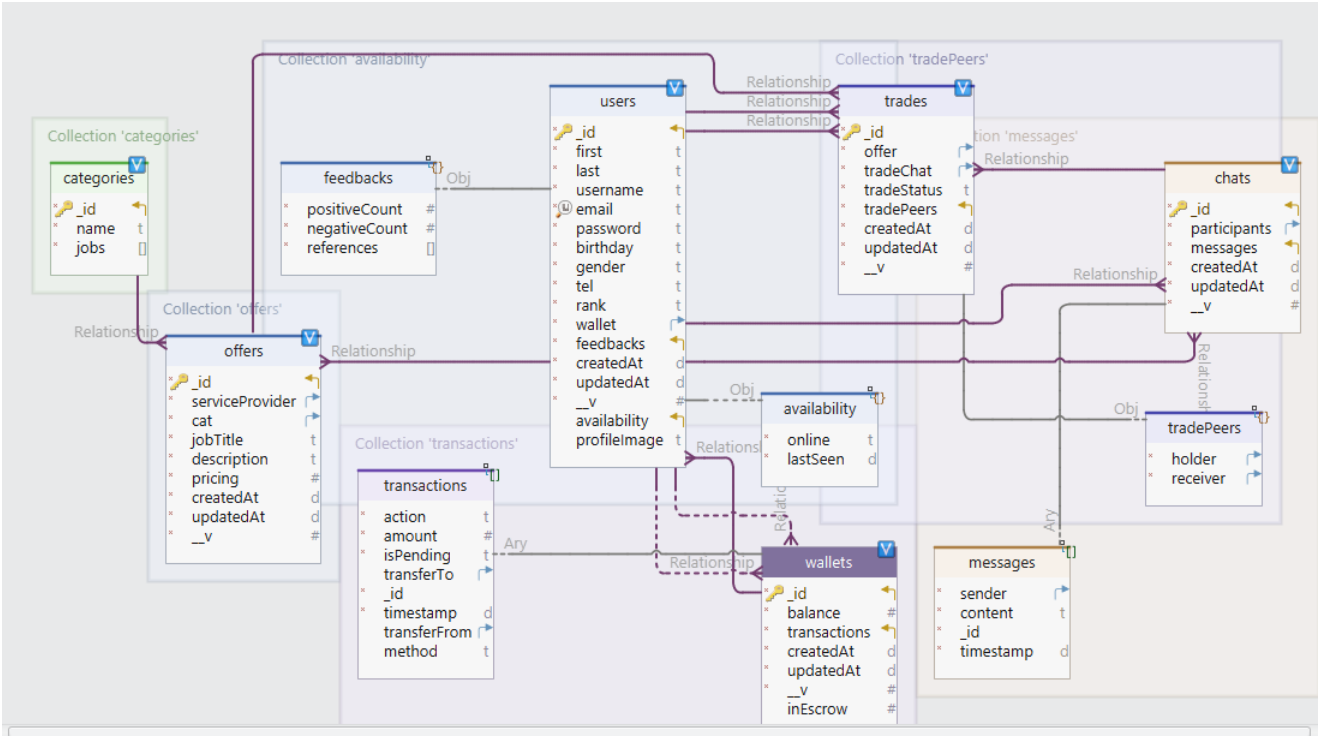


Figure 3.12: Database Modeling

3.5.2.2 Database Tables

1. **Users:** This table stores information about users, including their usernames, emails, passwords, and profile information such as date of birth, gender, profile picture, phone

number, and rating. It has relationship with the (wallets) table, while (Feedback and Availability) are objects.

2. **Feedbaks:** This object stores information about comments left by users and the number of positive and negative ratings given by customers.
3. **Availability:** This object stores information about the user's availability and the last time they were available.
4. **Wallets:** This table stores information about the user's wallet, including the balance, security features , creation or updating of a wallet,while (transactions) are array .
5. **Transactions:** this array is where information about platform transactions is stored, covering details like amount, timestamp, method, sender, receiver, and the transferTo and transferFrom fields. Additionally, it includes an "isPending" field that indicates the payment status (e.g., pending, paid, or refunded).
6. **Trades:** This table stores information about the trades that take place on the platform, including the offer, the conversation, and the time of its creation and update, and it has a relationship with the (chats, users, offers) tables , while (tradePeers) are object
7. **Chats:** This table stores information about conversations between users, the times of creating and updating the conversation, and the list of participants, and it has a relationship with a tables (users and offers) , while (messages) are array.
8. **Messages:** This array stores information about messages exchanged between users, including message content, timestamps, sender and recipient.
9. **Offers:** This table stores information about the offers available on the platform, including the name, description, price, and time of creation and updating, and it has a relationship with the (chats) table.
10. **Categories:** This table stores information about the categories available on the platform, including names and jobs .

3.6 Conclusion

In conclusion, for the optimal design and implementation of a Service Providers platform, adopting a user-friendly interface similar to Facebook and Upwork is crucial. This will enhance

user experience and attract more users. To prevent off-platform transactions, implement clear policies and secure payment methods that keep transactions within the platform. Utilizing Agile methodology ensures flexibility, collaboration, and continuous improvement, essential for delivering high-quality products efficiently. Non-functional requirements such as security, reliability, and scalability must be prioritized to ensure a robust platform. Functional requirements, including user management, secure transactions, and data insights, are vital for effective platform operation. Proper system and database design, using tools like UML and StarUML, will support the platform's structural integrity and operational efficiency, fostering long-term sustainability and growth.

Chapter 4

Implementation

4.1 Introduction

In this chapter, we will present the details related to the practical creation of our mobile application and website platform. Specifically, we begin by discussing the various programming languages used in both front-end and back-end development. We also introduce the frameworks and tools that played a crucial role in our development process.

Finally, we conclude this chapter with a presentation of the platform, highlighting the interfaces designed for both mobile and web applications.

4.2 development frameworks, tools and APIs

4.2.1 Front-End

This section covers the various programming languages and framework utilized in both front-end development for the creation of our mobile app and website platform.

4.2.1.1 Language

1. **HTML** 

    - **HTML5** (Hypertext Markup Language) is the standard markup language for creating the structure and content of web pages. It defines elements such as headings, paragraphs, images, links and forms.[18]

2. **CSS** 

   - **CSS3** (Cascading Style Sheets) is a style sheet language used to control the presentation and layout of HTML documents. It allows developers to define styles for elements to create visually appealing designs.[19]

3. **Dart** 

   - **Dart** is a general-purpose programming language developed by Google. It is designed to be easy to learn yet powerful enough to build complex applications, especially for web, mobile, and server-side development. Dart has features that make it suitable for both front-end and back-end development, making it a versatile language for creating modern applications.[20]

4.2.1.2 Framework

1. **Tailwind** 

   - **Tailwind** CSS v3.0 is specifically crafted and tested to be compatible with the latest stable versions of Chrome, Firefox, Edge, and Safari, while lacking support for any version of Internet Explorer, including IE 11. It is a popular utility-first CSS framework used for building modern and responsive web designs. The framework provides a set of pre-designed utility classes that developers can use to style their HTML elements without writing custom CSS. [21]

2. **Flutter** 

   - **Flutter** is a software development framework by Google that lets developers build mobile, web, and desktop applications from a single codebase. It uses the Dart programming language and offers a range of widgets and tools for creating attractive and high-performing user interfaces. The framework is praised for delivering fast, smooth user experiences on various platforms and devices. The Google community regularly updates Flutter with new features and enhancements, ensuring its ongoing popularity among developers. It is free and open-source.[22]

4.2.1.3 Libraries and packages

1. **ReactJs** 

   - **React.js**, commonly known as **React**, is an open-source JavaScript library maintained by Facebook. It enables the creation of views that are rendered in HTML. React adheres to a component-based architecture, wherein user interfaces (UIs) are constructed using reusable and encapsulated components. These components have the ability to manage their state and properties autonomously, leading to a codebase that is modular and simpler to maintain.[23, chapter-1 page-5]

2. **React-Router Dom** 

   - **React-Router** is a library that handles URLs and history, similar to how Express manages server-side routing. It functions by parsing URLs and associating specific code with them based on their components. Additionally, it aids in transitioning between views and ensuring seamless navigation using the browser's Back button. This makes it a user-friendly option compared to constructing routing systems from scratch.[23, chapter-1 page-12]

3. **Redux toolkit** 

   - **Redux Toolkit** (also known as "RTK" for short) is a package designed to make Redux development easier and boost productivity by offering a collection of tools and utilities. It aims to streamline typical Redux tasks, minimize boilerplate code, and includes essential API methods and dependencies for building Redux applications.[24]

4. **Socket.IO** 

   - **Socket.IO** is a library that enables real-time, bidirectional communication between web clients and servers. The Socket.IO client is the part of the library that runs on the client side (browser) and facilitates communication with the server via WebSocket or other transport mechanisms.[25]

5. **Axios** 

   - **Axios** is a widely-used JavaScript library that makes it easier to send HTTP requests from both web browsers and Node.js environments. It provides a user-friendly API

for handling asynchronous requests and managing responses.

Axios comes with several useful features including interceptors, cancellation support, automatic JSON parsing, and error handling capabilities. When used on the server side, Axios uses the native node.js http module, while on the client side (browser), it uses XMLHttpRequests for communication.[26]

4.2.2 Back-End

4.2.2.1 JavaScript 

- **JavaScript** is a dynamic programming language for client-side scripting in web development. It adds interactivity and dynamic behaviour to web pages, such as form validation, animation, and event handling.[27]

4.2.2.2 NodeJs 

- **Node.js** is a JavaScript runtime environment that allows developers to run JavaScript code outside of a web browser. It's built on Chrome's V8 JavaScript engine and provides an event-driven, non-blocking I/O model that makes it lightweight and efficient for building scalable network applications. Node.js is commonly used for server-side scripting, backend development, and building web applications.[23, chapter-1 page-7]

4.2.2.3 JSON 

- **JSON**, short for JavaScript Object Notation, serves as the primary encoding method for data in both request and response bodies. It represents a lightweight data interchange format that humans find easy to read and write, while machines can effortlessly parse and generate it. JSON finds widespread use in transmitting data between servers and web applications, and it is also commonly employed for data storage in files or databases.[23, chapter-5 page-71]

4.2.2.4 MongoDB 

- **MongoDB** is the database used in the MERN stack. It is a NoSQL document-oriented database, with a flexible schema and a JSON based query language. I'll discuss a few

things that MongoDB is (and is not) here.

- **NoSQL**

  NoSQL, short for "non-relational," refers to databases that do not adhere to the traditional table-and-column structure of relational databases. There are two key aspects that distinguish NoSQL databases. Firstly, they can horizontally scale by spreading the workload across multiple servers, albeit sacrificing strong consistency for brief periods. Secondly, NoSQL databases are not limited to relational data models; they allow for data representation as objects or documents, eliminating the need for object-relational mapping layers in programming. This flexibility reduces impedance mismatch, streamlining the development process in platforms like MongoDB.[23, chapter-1 page 10]

- **Document-Oriented**

  MongoDB is a flexible, document-oriented database that organizes data into collections, similar to tables. It uses documents akin to rows but is more adaptable and efficient, with each document having a unique identifier for quick access. Unlike relational databases that use multiple tables for complex data, MongoDB stores such data as a single document, simplifying operations and enabling atomic updates. It supports nested objects and indexing on nested fields, providing advantages over relational databases. However, MongoDB's approach may lead to denormalized storage, potentially causing duplication and increased space usage. Nonetheless, considering decreasing storage costs and infrequent operations, MongoDB's benefits often outweigh these drawbacks.[23, chapter-1 page-10,11]

4.2.2.5 JWT

- **JWT** stands for JSON Web Token, which is a compact and self-contained method of securely transmitting information between parties as a JSON object. It can be signed using a public/private key pair with RSA or ECDSA methods, or a JWTs using a secret (using the HMAC algorithm).[28]

4.2.2.6 Libraries and framework

1. **Express Js**

- **Express** is a framework for Node.js that simplifies writing server code by allowing you to define routes for handling HTTP requests. It handles parsing request details and provides functions for setting response data. Express supports custom middleware and various template engines, making it versatile for web development. For single-page applications, Express isn't focused on server-side rendering, as dynamic content is typically generated on the client side with technologies like React in the MERN stack. Overall, Express offers standard functionalities for building web servers in Node.js.[23, chapter-1 page-9 chapter-2 page-24,72]

2. **Mongoose**

- **Mongoose** is a popular Node.js library used for working with MongoDB, a NoSQL database. It simplifies the process of interacting with MongoDB by providing a schema-based solution for modeling data, data validation, and query building. [23, chapter-13 page-319]

4.2.3 Programming Tools:

Tool Name	Definition
Visual Studio Code 	(VS Code) is a free code editor from Microsoft available on Windows, Linux, and macOS. It's like an IDE, making it versatile for different programming languages and tasks.[29]
Postman 	Postman is a popular API client tool used for testing, developing, and documenting APIs. and send HTTP requests to APIs,Postman supports various request types, including GET, POST, PUT, DELETE, and more.[30]
Vite 	Vite is a frontend build tool that enhances modern web development by offering a fast development server with hot module replacement (HMR). This allows developers to view changes instantly without reloading the entire page.[31]
Git 	A free, open-source distributed version control system that enables developers to track code changes and collaborate with one another.[32]

Tool Name	Definition
Github 	A web-based Git repository hosting service that offers a platform for code collaboration, version control, and project management[33].
Docker 	Docker is an open-source platform that automates application deployment, scaling, and management using containerization.[34]

Table 4.1: Tools Used in Developing the KHDEMNI Platform.

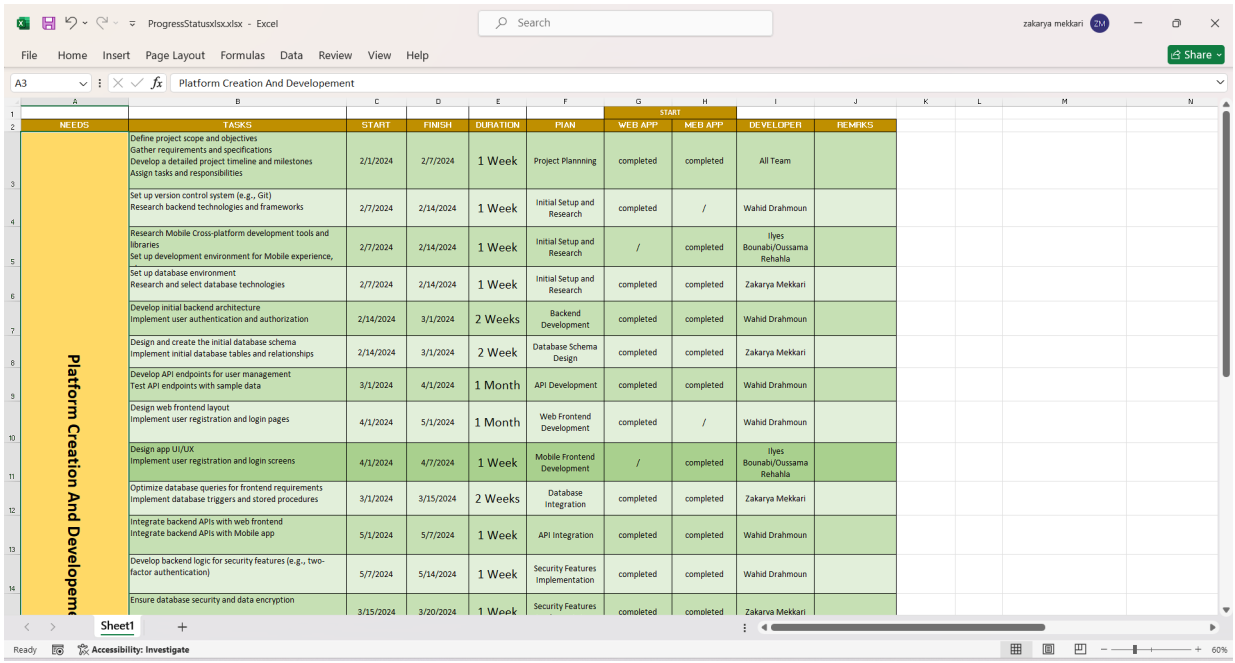
4.3 Programming Tasks

The Agile methodology was used in developing the KHdemni platform to effectively manage project needs and tasks. Initially, the project's requirements were identified, prioritized, and assigned points based on their importance to gauge the project's progress rate. This approach enabled the team to concentrate on the most critical needs and features first, ensuring the timely development and implementation of the platform's essential functionalities.

Determining Tasks

To ensure efficient completion and tracking of tasks, an Excel template was used to record all completed and remaining tasks. This approach facilitated monitoring the project's progress, identifying bottlenecks, and mitigating risks by tracking tasks and project progress in real time.

Using Agile methodology to manage project needs and tasks streamlined the development process, ensuring that the platform was developed to meet users' needs. This approach allowed for greater flexibility, adaptability, and collaboration, resulting in a more robust and reliable platform that met the project's requirements.



NEEDS	TASKS	START	FINISH	DURATION	PIAN	START WEB APP	START MEB APP	DEVELOPER	REMIKS
Platform Creation And Developerm	Define project scope and objectives Gather requirements and specifications Develop a detailed project timeline and milestones Assign tasks and responsibilities	2/1/2024	2/7/2024	1 Week	Project Plannning	completed	completed	All Team	
	Set up version control system (e.g., Git) Research backend technologies and frameworks	2/7/2024	2/14/2024	1 Week	Initial Setup and Research	completed	/	Wahid Drahmoun	
	Research Mobile Cross-platform development tools and libraries Set up development environment for Mobile experience,	2/7/2024	2/14/2024	1 Week	Initial Setup and Research	/	completed	Ilyes Bounabi/Oussama Rehahla	
	Set up database environment Research and select database technologies	2/7/2024	2/14/2024	1 Week	Initial Setup and Research	completed	completed	Zakarya Mekkari	
	Develop initial backend architecture Implement user authentication and authorization	2/14/2024	3/1/2024	2 Weeks	Backend Development	completed	completed	Wahid Drahmoun	
	Design and create the initial database schema Implement initial database tables and relationships	2/14/2024	3/1/2024	2 Week	Database Schema Design	completed	completed	Zakarya Mekkari	
	Develop API endpoints for user management Test API endpoints with sample data	3/1/2024	4/1/2024	1 Month	API Development	completed	completed	Wahid Drahmoun	
	Design web frontend layout Implement user registration and login pages	4/1/2024	5/1/2024	1 Month	Web Frontend Development	completed	/	Wahid Drahmoun	
	Design app UI/UX Implement user registration and login screens	4/1/2024	4/7/2024	1 Week	Mobile Frontend Development	/	completed	Ilyes Bounabi/Oussama Rehahla	
	Optimize database queries for frontend requirements Implement database triggers and stored procedures	3/1/2024	3/15/2024	2 Weeks	Database Integration	completed	completed	Zakarya Mekkari	
	Integrate backend APIs with web frontend Integrate backend APIs with Mobile app	5/1/2024	5/7/2024	1 Week	API Integration	completed	completed	Wahid Drahmoun	
	Develop backend logic for security features (e.g., two-factor authentication)	5/7/2024	5/14/2024	1 Week	Security Features Implementation	completed	completed	Wahid Drahmoun	
	Ensure database security and data encryption	3/15/2024	3/20/2024	1 Week	Security Features	completed	completed	Zakarya Mekkari	

Figure 4.1: An Excel sheet for the project progress form.

4.4 System Architecture

4.4.1 High Level System Design

High-level system design refers to the process of conceptualizing and outlining the architecture and components of a system at a broad level. This involves defining the major components, their interactions, and the overall structure of the system without delving into detailed implementation or technical specifics.

Figure 4.2 illustrates the main components that constitutes our system.

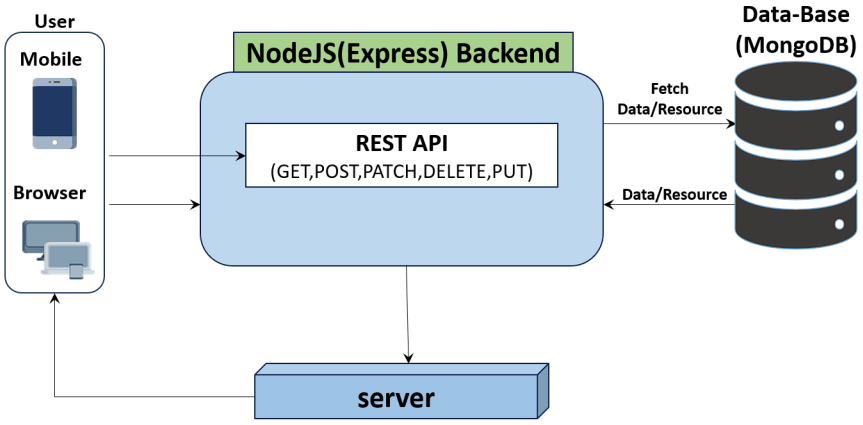


Figure 4.2: High level system design flow

The system comprises three main components: the mobile and web UI, which share a MongoDB server with a REST API. This setup is designed to enhance scalability, flexibility, and, most importantly, secure data processing and transfer.

4.4.2 Model View Controller

The Model-View-Controller (MVC) architecture figure(4.3) is a design pattern used in software development to separate an application into three interconnected components: the Model, the View, and the Controller. This separation helps manage the complexity of large-scale applications by promoting modality and facilitating code maintainability and scalability.[35]

In the context of the MERN stack, which comprises MongoDB, Express.js, React, and Node.js,

The MVC pattern can be implemented as follows:

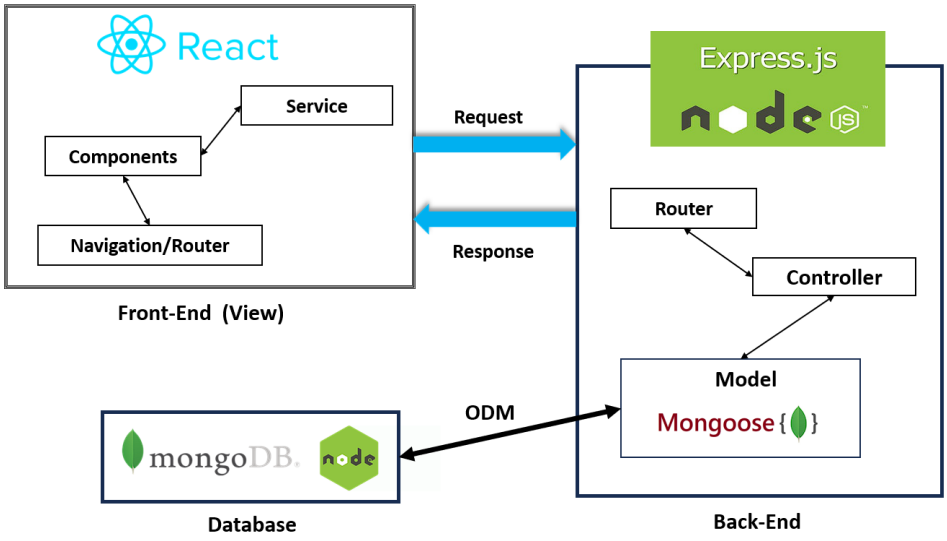


Figure 4.3: (Model View Controller) Architecture in the MERN Stack

1. **Model:** This segment embodies the data and business logic of the application, responsible for managing data manipulations.[36] In the MERN stack, the Model is handled by MongoDB along with Mongoose (an ODM for MongoDB).

2. **View:** The View represents the user interface components of the application. It displays the data to the users and interacts with them. The View component is responsible for presenting information to users and receiving their input. In the MERN stack, the View is primarily handled by React.

3. **Controller:** The Controller serves as a mediator between the model and the view, taking charge of processing user inputs, deciding the suitable view for presentation, and initiating the necessary actions or modifications in the model.

   In the MVC architecture, the data flow and interactions typically unfold in this manner the user engages with the View, which in turn prompts a request that the Controller handles. Subsequently, the Controller updates the Model according to the request, and the View fetches the updated data from the Model to showcase it to the user. In the MERN stack, the Controller is implemented using Express.js in conjunction with Node.js.

   The MVC pattern advocates for separating concerns, resulting in a more organized, modular, and easier-to-maintain application. This approach improves code reusability, scalability, and testability by breaking down the application logic into distinct components with clearly defined responsibilities.

4.4.2.1 Integrating with Back-End APIs

The ReactJS front-end seamlessly integrates with the Node.js Express back-end, using libraries like Axios for the web app and libraries like Pub for the mobile app to facilitate smooth communication through HTTP requests. Functions and methods are thoughtfully designed within the ReactJS code-base to handle various API calls, ensuring efficient data transmission to the back-end and processing of responses.

By leveraging Axios and Pub or similar libraries, robust interaction between the ReactJS front-end and Node.js Express back-end is achieved, enabling effective data exchange and synchronization. The Axios and Pub client includes request interceptors to handle authentication. The
axiosClient.interceptors.request.use() method adds a request interceptor that retrieves the access token from local storage and adds it to the Authorization header of each request. This ensures that the back-end APIs are accessed with the appropriate authentication.

The ‘axiosClient‘ instance can be imported and used throughout the ReactJS code-base to make HTTP requests to the Node.js Express back-end APIs. The features and methods of the Axios and Pub library can be leveraged to customize API calls, handle responses, and implement data processing and error-handling logic specific to the application’s requirements.

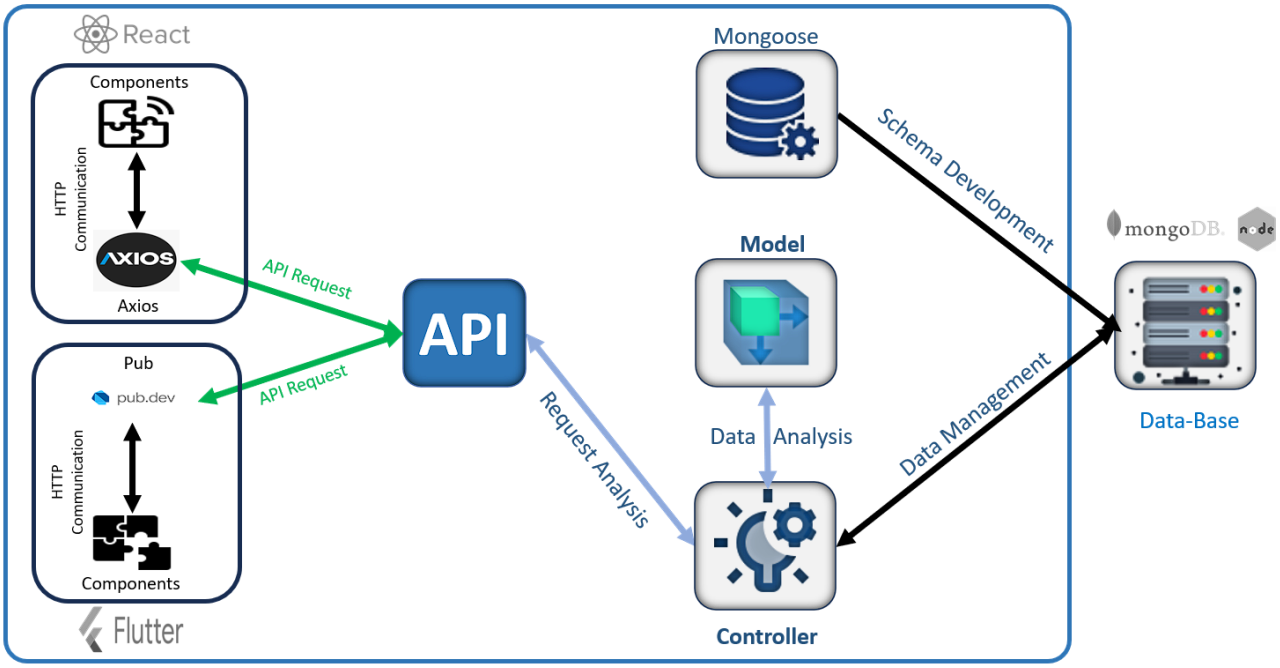


Figure 4.4: Project Structure.

4.4.3 RESTful Services

RESTful services, often referred to as RESTful APIs, are an architectural style for designing networked applications. They provide a type of web service that follows the principles of Representational State Transfer (REST), emphasizing simplicity, scalability, and maintainability.

Generally, when developing a RESTful API, we implement the MVC architecture on the server side to manage API requests. These requests are directed to the relevant controllers, which communicate with the models to fetch or modify data and then provide the response in a format suitable for consumption by the mobile app, often JSON.

Below are a few reasons explaining the popularity of RESTful APIs:

1. **Simplicity:** RESTful services use a uniform interface for interactions between clients and servers. This interface is typically based on standard HTTP methods such as GET , POST, PUT , DELETE , and others.

2. **Scalability:** RESTful services are stateless, meaning that each request from a client to the server must contain all the information necessary for the server to fulfill the request. The server does not store any client state between requests, improving scalability and reliability.

3. **Flexibility:** RESTful services represent resources in a format easily comprehensible to clients, commonly employing formats such as JSON (JavaScript Object Notation). Clients can request varying representations of a resource as per their requirements. RESTful APIs are versatile, compatible with any programming language or platform due to their reliance on standard protocols like HTTP and JSON.
4. **Security:** RESTful APIs can achieve security through standard authentication mechanisms such as OAuth, guaranteeing access to resources only for authorized users. They offer a flexible and scalable approach to developing web services, facilitating seamless integration with other systems while ensuring a high level of security.

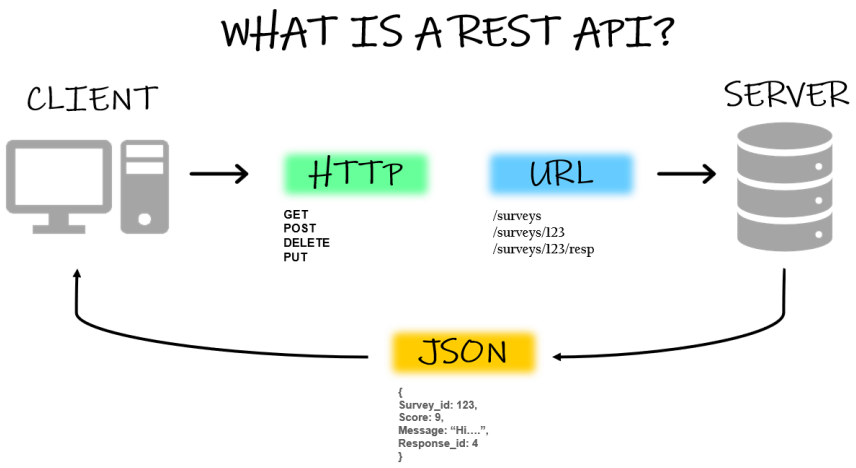


Figure 4.5: What Is A Rest API

4.4.4 Socket.IO Overview and Workflow Explanation

Socket.IO is a library that enables real-time, bidirectional, and event-based communication between web clients and servers. It consists of a client-side library that runs in the browser and a server-side library for Node.js. Socket.IO is built on WebSockets but provides additional features such as:

- **Automatic reconnection**: Automatically attempts to reconnect when a connection is lost.
- **Multiplexing support**: Allows multiple namespaces, enabling different endpoints to be communicated simultaneously.
- **Event handling**: Supports custom events that can be defined and triggered on both the

client and server sides.

- **Fallback to polling**: If WebSockets are not supported or fail, Socket.IO falls back to long-polling or other transport methods to maintain communication.

4.4.4.1 Workflow and Flowchart Explanation

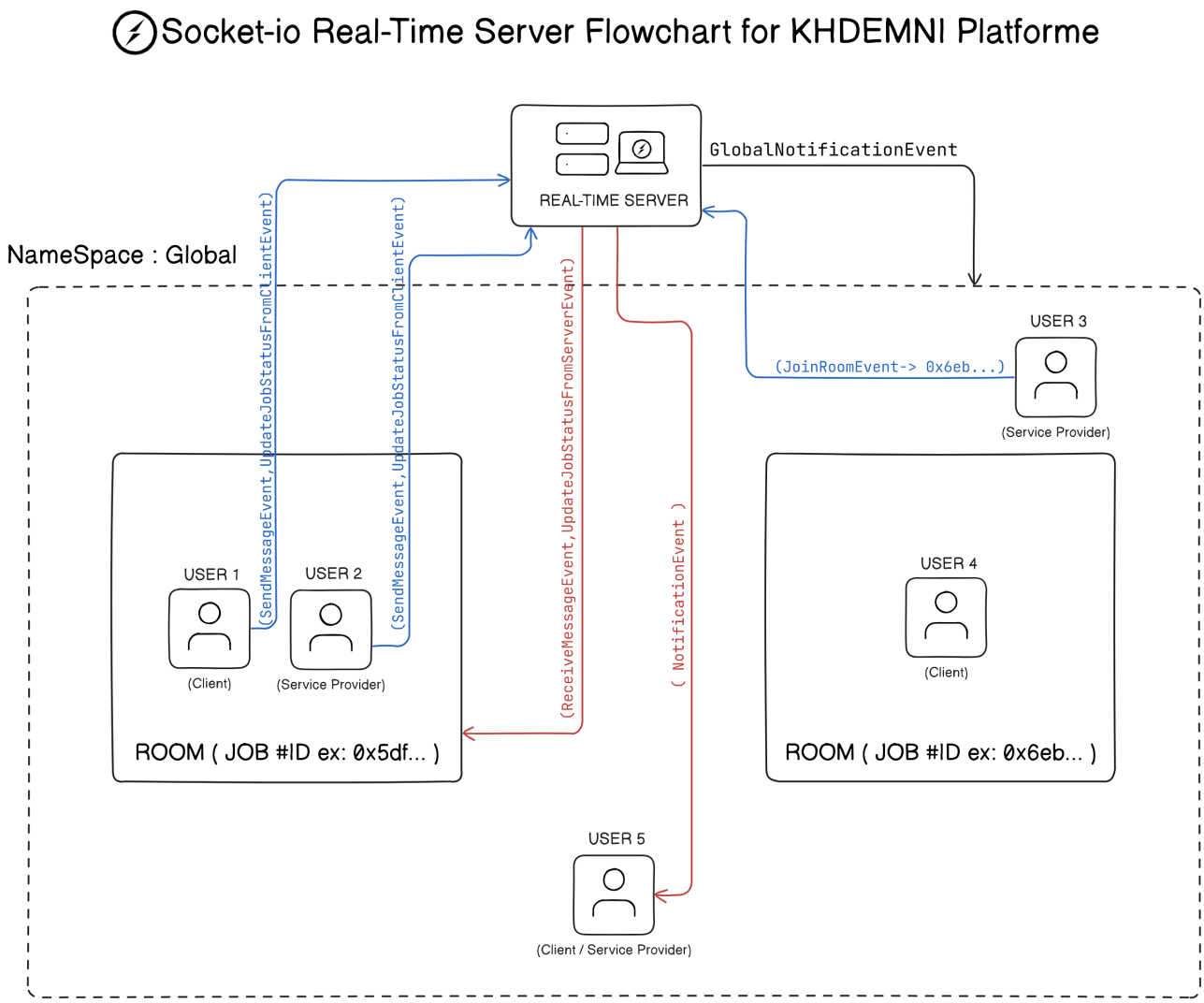


Figure 4.6: Real-Time Flowchart for KHDEMNI platform

4.4.4.2 General Workflow

1. **Connection Establishment**: Users (clients or service providers) connect to the real-time server via Socket.IO.
2. **Namespace Usage**: Communication happens within a specific namespace (Global in this case), allowing for organized handling of events.

3. **Room Management**: Each job has a dedicated room identified by a unique job ID, enabling isolated communication related to that specific job.
4. **Event Handling**: Various events (*SendMessageEvent*, *ReceiveMessageEvent*, *GlobalNotificationEvent*, etc.) are defined to manage the communication flow and status updates.

4.4.4.3 Flowchart Breakdown

1. **Components**
   - **Real-Time Server**: Central hub that manages connections, rooms, and events.
   - **Users**: Can be clients, service providers, or both, interacting within specific rooms based on job IDs.
2. **Events**
   - **GlobalNotificationEvent**:
     - Sent from the server to all clients connected to the Global namespace.
     - **Purpose**: To broadcast notifications globally to all users.
   - **JoinRoomEvent**:
     - Triggered by users to join a specific room associated with a job ID.
     - **Example**: *JoinRoomEvent -> 0x6eb...* indicates a user joins the room for job ID *0x6eb...*.
   - **SendMessageEvent (User to Server)**:
     - Users send messages or updates to the server within the context of a room.
     - **Example**: User 1 and User 2 send messages or status updates related to their job.
   - **ReceiveMessageEvent and UpdateJobStatusFromServerEvent (Server to User)**:
     - The server forwards messages or updates to users in the room.
     - Ensures all participants in a room receive the necessary communication.
   - **NotificationEvent**:
     - Sent by the server to users to notify about updates or changes.
     - **Example**: User 5 receives notifications from the server.

4.4.4.4 Specific User Interactions

1. **User 1 and User 2:**
    - Both are in the same room associated with job ID *0x5df...*.
    - They send messages and status updates to the server, which are then relayed back as necessary.
2. **User 3:**
    - Joins a different room with job ID *0x6eb...*.
    - Receives global notifications and specific updates pertinent to the job.
3. **User 4:**
    - Similarly, joins the room for job ID *0x6eb...*.
    - Participates in job-specific communication and receives updates.
4. **User 5:**
    - Interacts as both client and service provider.
    - Receives notifications from the server, indicating updates relevant to their jobs.

The flowchart demonstrates a real-time server architecture using Socket.IO for the KHDEMNI platform. It effectively illustrates the communication flow between users and the server, highlighting the use of namespaces and rooms for organized, job-specific interactions. Events are utilized to handle messages, status updates, and notifications, ensuring smooth and efficient real-time communication.

4.5 Installation and Project Initialization

- **Prerequisites** Before starting the installation, we ensured that the following tools were installed on our machines:
    - Node.js (version 17)
    - Vite
    - MongoDB and MongoDB Compass (GUI for MongoDB)
    - Git

- Flutter SDK
- Docker and Docker Compose

- **Project Structure** The KHDEMNI project repository is structured as follows:
  - **Client** Front-end (React.js and Flutter)
  - **Web** React.js application
  - **mobile** Flutter application
  - **server** Back-end (Node.js)

4.5.1 Installation Steps

4.5.1.1 Installing Necessary Tools

1. **Installing Node.js (version 17):**
   - Downloading and installing Node.js from the https://nodejs.org/.

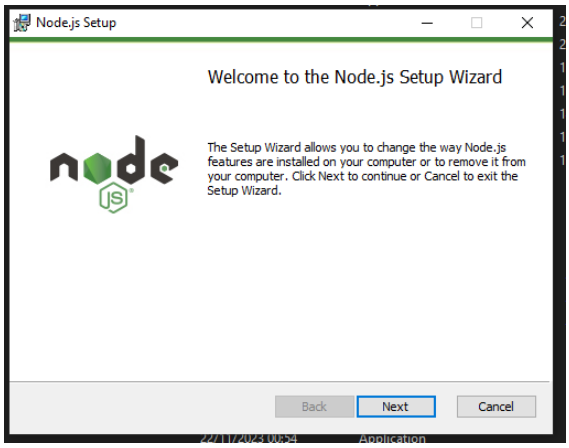


Figure 4.7: NodeJs Installation Setup.

2. **Installing Vite:**
   - Vite will be installed as a development dependency in our React.js project.
3. **Installing MongoDB and MongoDB Compass:**
   - Downloading and installing MongoDB from the official website https://www.mongodb.com/try/download/community
   - Downloading and installing MongoDB Compass for a GUI interface from the official website https://www.mongodb.com/products/compass

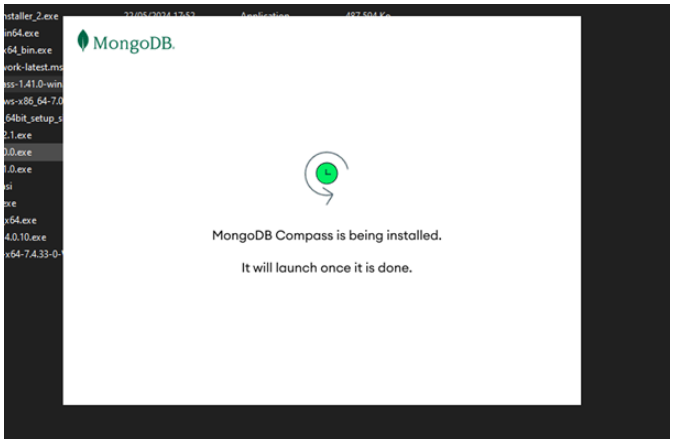


Figure 4.8: MongoDb Compass Installation Setup

4. **Installing Git:**

   - Downloading and installing Git from the official website https://git-scm.com/.

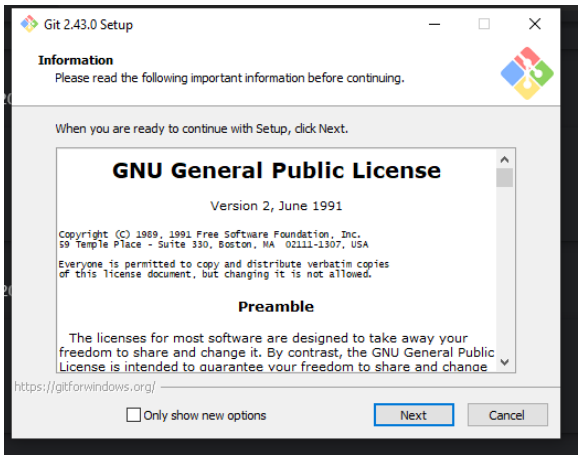


Figure 4.9: Git Installation Setup

5. **Installing Flutter SDK:**

   - We Followed the Flutter installation guide on https://flutter.dev/docs/get-started/install.

6. **Install Docker and Docker Compose:**

   - Downloading and installing Docker Desktop from the https://www.docker.com/products/docker-desktop.

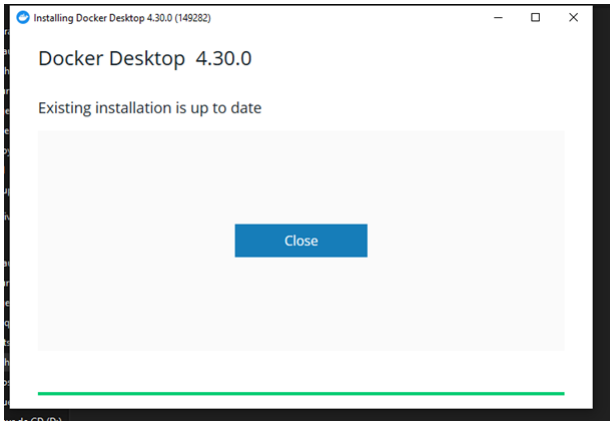


Figure 4.10: Docker and Docker Compose Installation Setup

4.5.2 Project Initialization

4.5.2.1 Front End (React.js)

1. **Initializing a React Project with Vite:**
   - we Navigate to the ‘client/web‘ directory:

```
mkdir -p khdemni/client/web
cd khdemni/client/web
npm create vite@latest . --template react

```

2. **Installing Project Dependencies:**
   - Installing necessary packages:

```
npm install

```

3. **Set Up Tailwind CSS:**
   - Installing Tailwind CSS:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init

```

   - we Configure Tailwind CSS by editing the ‘tailwind.config.js‘ file and adding the necessary import statements in your CSS files.

4. **initializing Git Repository:**

- we Initialize a Git repository and make the initial commit:

```
git init
git add .
git commit -m "Initial commit"

```

4.5.2.2 Mobile (Flutter)

1. **Initializing a Flutter Project:**

   - we Navigate to the ‘client/mobile‘ directory and create a new Flutter project:

```
mkdir -p khdemni/client/mobile
cd khdemni/client/mobile
flutter create .

```

2. **Installing Project Dependencies:**

   - Installing Flutter packages:

```
flutter pub get

```

3. **Initialize Git Repository:**

   - Initialize a Git repository and make the initial commit:

```
git init
git add .
git commit -m "Initial commit"

```

4.5.2.3 Back End (Node.js)

1. **Initialize a Node.js Project:**

   - we Navigate to the ‘server‘ directory and initialize a new Node.js project:

```
mkdir -p khdemni/server
cd khdemni/server
npm init -y

```

2. **Install Necessary Packages:**

   - Install essential packages including Express.js and Nodemon:

```
npm install express
npm install --save-dev nodemon

```

3. **Set Up Nodemon:**

   - Adding a ‘start‘ script in ‘package.json‘ to use Nodemon:

```
"scripts": {
  "start": "nodemon index.js"
}

```

4. **Initialize Git Repository:**

   - Initializing a Git repository and making the initial commit:

```
git init
git add .
git commit -m "Initial commit"

```

4.5.3 Pushing to GitHub Repository

1. **Create a GitHub Repository:**

   - we Created a new repository on GitHub. We named it ‘khdemni‘ , and this the repository URL : https://github.com/azerazerty/khdemni.git.

2. **Push Local Repository to GitHub:**

   - Adding the remote repository URL:

```
git remote add origin https://github.com/azerazerty/khdemni.
  git

```

   - Push the local repository to GitHub:

```
git branch -M main
git push -u origin main
```

```
AZER AZERT@DESKTOP-93CNK3F MINGW64 /e/bricoula_project (main)
$ git push origin main --force
Enumerating objects: 183, done.
Counting objects: 100% (183/183), done.
Delta compression using up to 4 threads
Compressing objects: 100% (108/108), done.
Writing objects: 100% (113/113), 2.50 MiB | 50.00 KiB/s, done.
Total 113 (delta 47), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (47/47), completed with 36 local objects.
To https://github.com/azerazerty/khdemni.git
   29ec072..bbeba9b  main -> main

AZER AZERT@DESKTOP-93CNK3F MINGW64 /e/bricoula_project (main)
```

Figure 4.11: running git push command

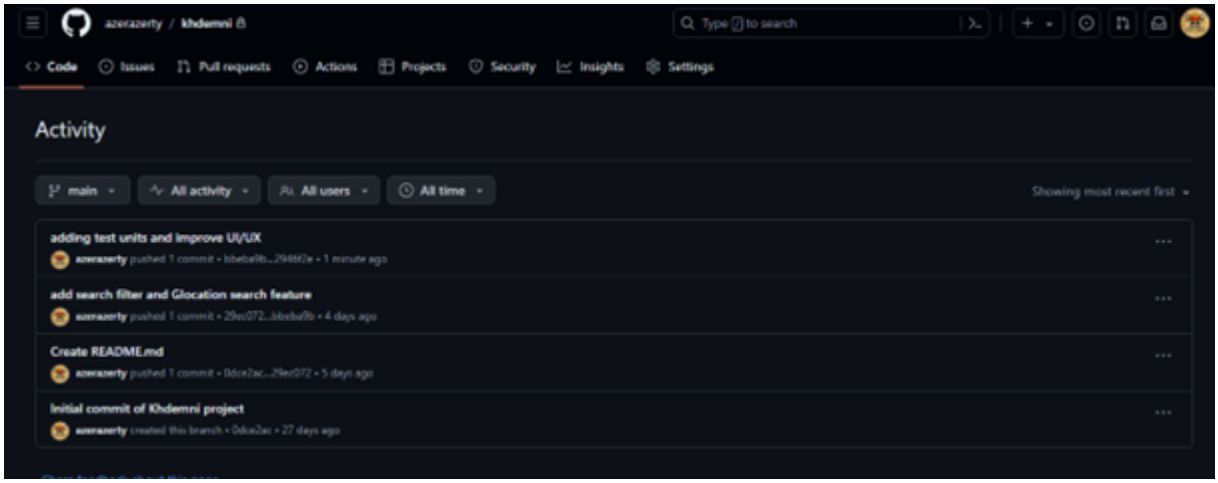


Figure 4.12: GitHub repo activity up to date

4.6 Testing

4.6.1 Front End Testing

1. **Unit Testing with Jest:**
   - We Ensure Jest is installed in our React project.
   - We Write unit tests in the ‘tests‘ directory with a ‘.test.js‘ extension.
   - In this example we are Writing unit tests in the ‘tests‘ directory for Form Component

4.6.2 Back End Testing

1. **Unit Testing with Jest:**

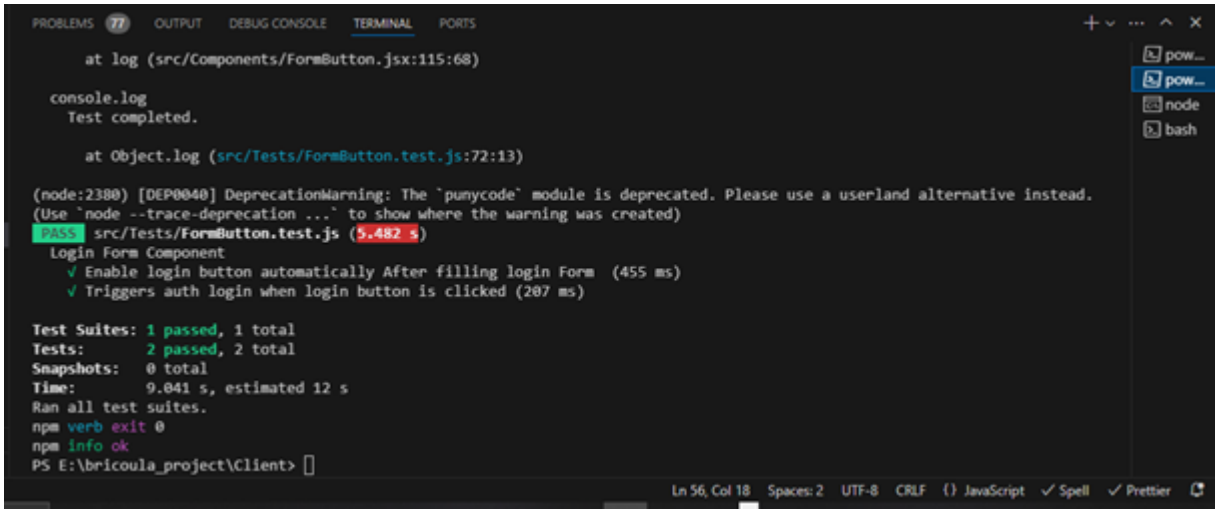


Figure 4.13: Front-End Unit Tests Example.

- We Ensure Jest is installed in our Node.js project.
- We Write unit tests in the ‘tests‘ directory with a ‘.test.js‘ extension.
- In this example we are Writing unit tests in the ‘tests‘ directory for testing different Api Endpoints



Figure 4.14: Back-End End Unit Tests Example.

4.7 Deployment

4.7.1 Docker and Docker Compose

4.7.1.1 Create Docker Files

1. **Front End Docker-file:**

   - Create a ‘Docker-file‘ in the ‘client/web‘ directory:

```
# Use the latest Node runtime as the base image
FROM node:latest as builder
# Set the working directory in the container
WORKDIR /app
# Copy package.json and package-lock.json to the working
  directory
COPY package.json package-lock.json ./
# Install dependencies
RUN npm install
# Copy the .env file
COPY .env .env
# Copy the rest of the application code
COPY . .
# Build the React app
RUN npm run build
# Use a smaller image for production
FROM nginx:alpine
# Copy the built app to the nginx web server directory
COPY --from=builder /app/dist /usr/share/nginx/html
# Expose port 80
EXPOSE 80
# Start nginx
CMD ["nginx", "-g", "daemon off;"]
```

2. **Back End Docker-file:**

   - Create a ‘Dockerfile‘ in the ‘server‘ directory:

```
# Use the latest Node runtime as the base image
FROM node:latest
# Set the working directory in the container
```

```
WORKDIR /app
# Copy package.json and package-lock.json to the working
  directory
COPY package.json package-lock.json ./
# Install dependencies
RUN npm install
# Copy the .env file
COPY .env .env
# Copy the rest of the application code
COPY . .
# Expose the backend port
EXPOSE 5000
# Start the backend server
CMD ["npm", "start"]
```

4.7.1.2 Create Docker Compose File

- Create a ‘docker-compose.yml‘ file in the root directory:

```
version: "3.8"

services:
  frontend:
    build:
      context: ./Client
    ports:
      - "4173:80"
    env_file:
      - ./Client/.env
    depends_on:
      - backend

  backend:
    build:
      context: ./server
    ports:
      - "3000:3000"
    env_file:
      - ./server/.env
```

```
    depends_on:
      - mongo

  mongo:
    image: mongo:latest
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db

volumes:
  mongo-data:
```

4.7.1.3 Running Docker Compose

- To build and start the containers, we run the following command in the root directory:

```
docker-compose up -build
```

This command will build the Docker images for both the front end and back end, and then start the containers, making the application accessible on our local machines at the specified ports.

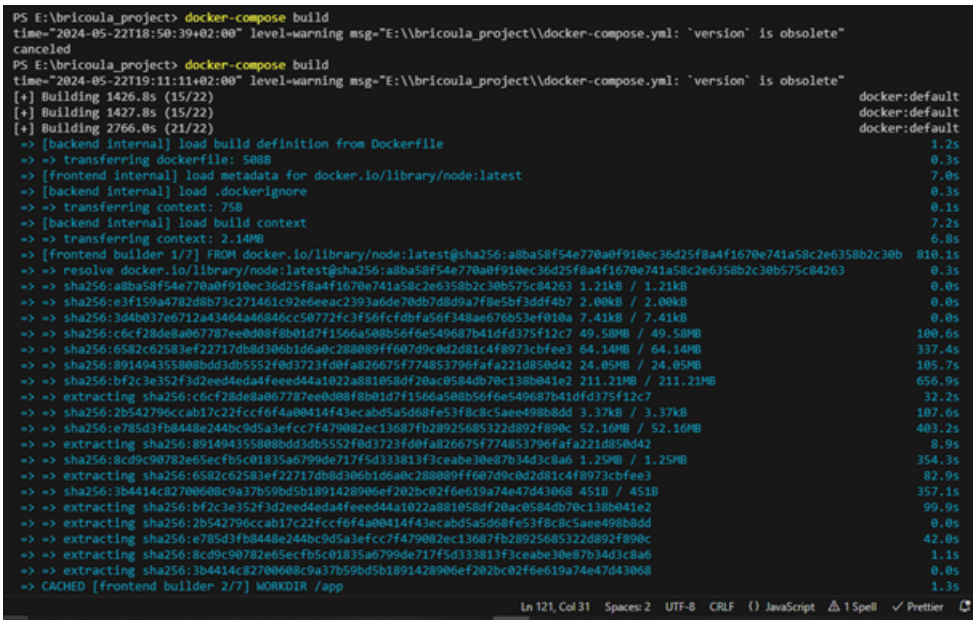


Figure 4.15: Docker Compose Building Stage

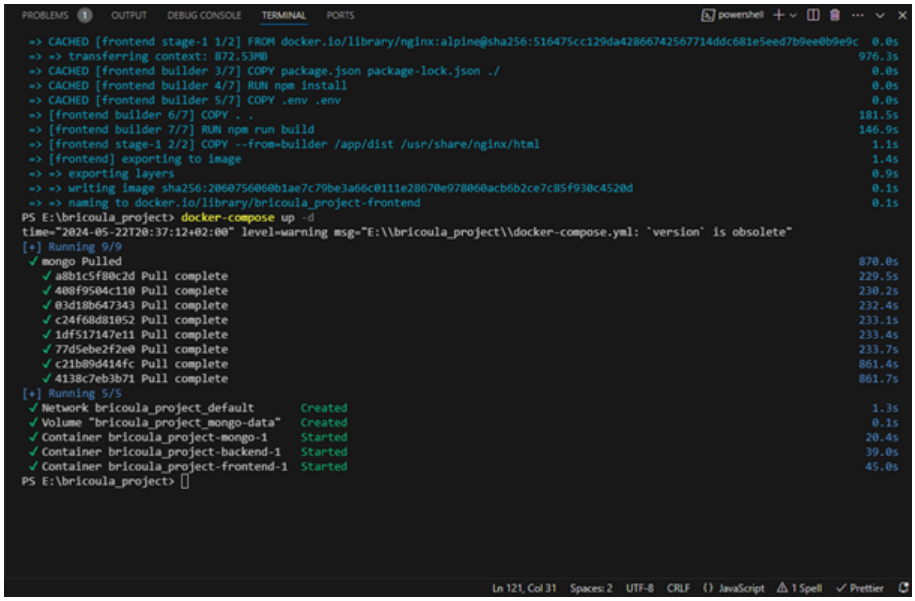


Figure 4.16: Docker Compose Running Stage

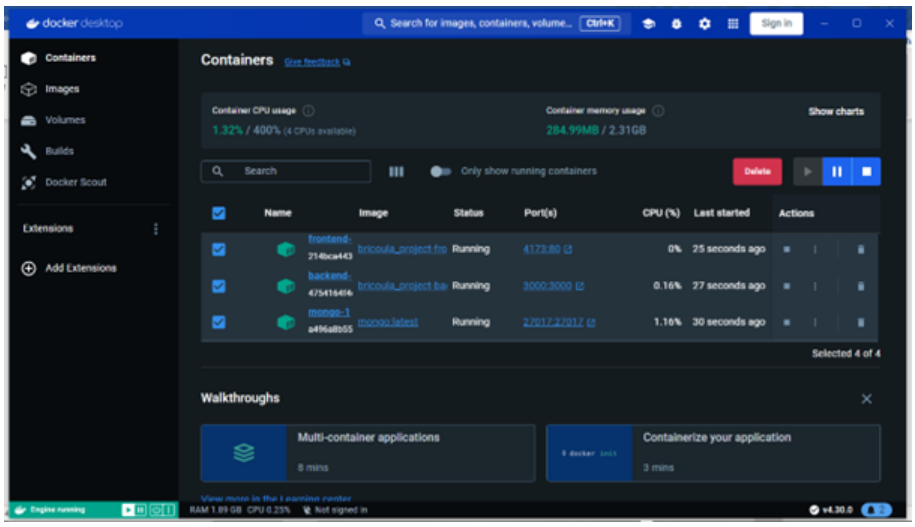


Figure 4.17: Docker Final Deployment

The implementation of KHDEMNI involves setting up a robust environment for both development and production. By utilizing modern tools and technologies such as Node.js, Vite, MongoDB, Tailwind CSS, and Docker, we have designed the project to be scalable and maintainable. The use of Jest for testing ensures the reliability of the application, while Docker and Docker Compose facilitate smooth deployment and scalability.

4.7.2 Final Look of the Platform

After implementing the necessary features and conducting thorough testing, the platform has achieved its final design and functionality. The user interface is intuitive and user-friendly, enabling users to easily navigate through different sections and perform various actions.

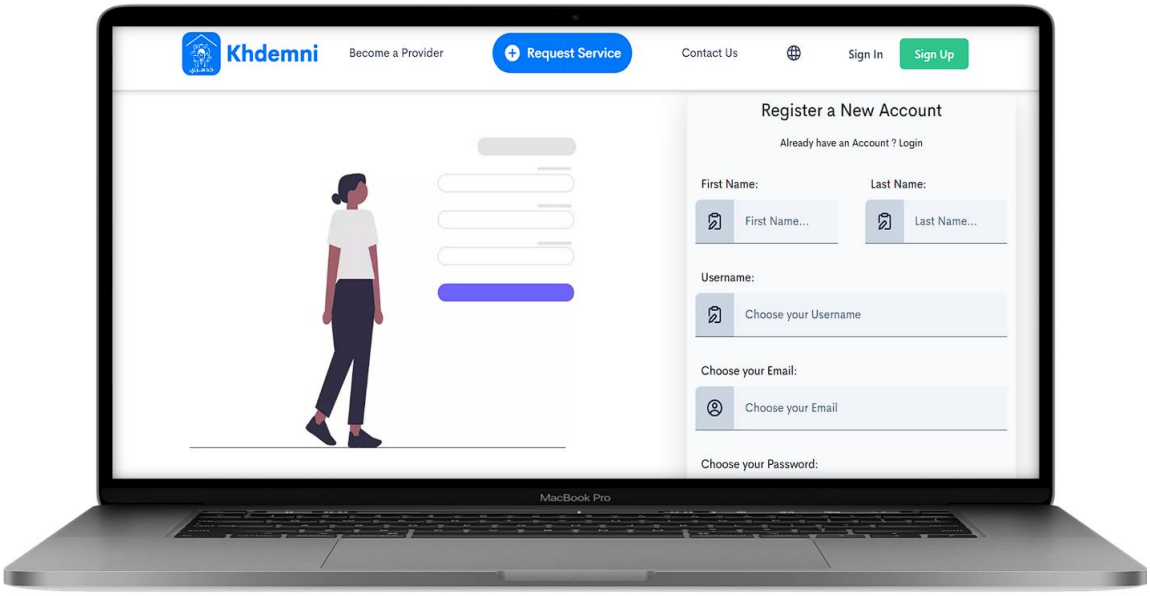


Figure 4.18: Sign Up Platform

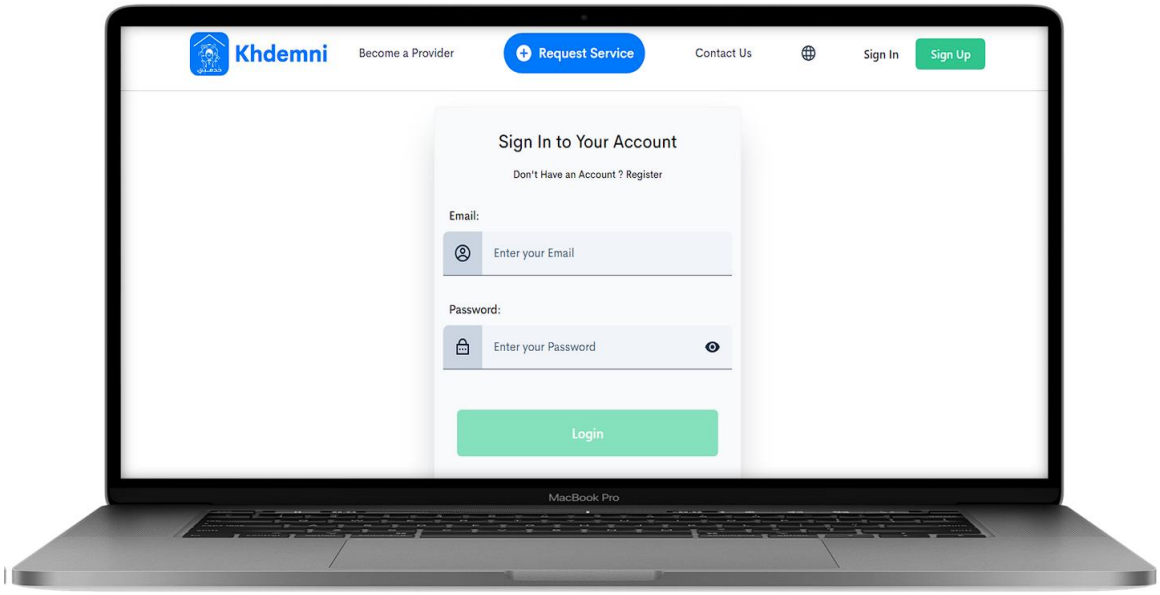


Figure 4.19: Sign In Platform

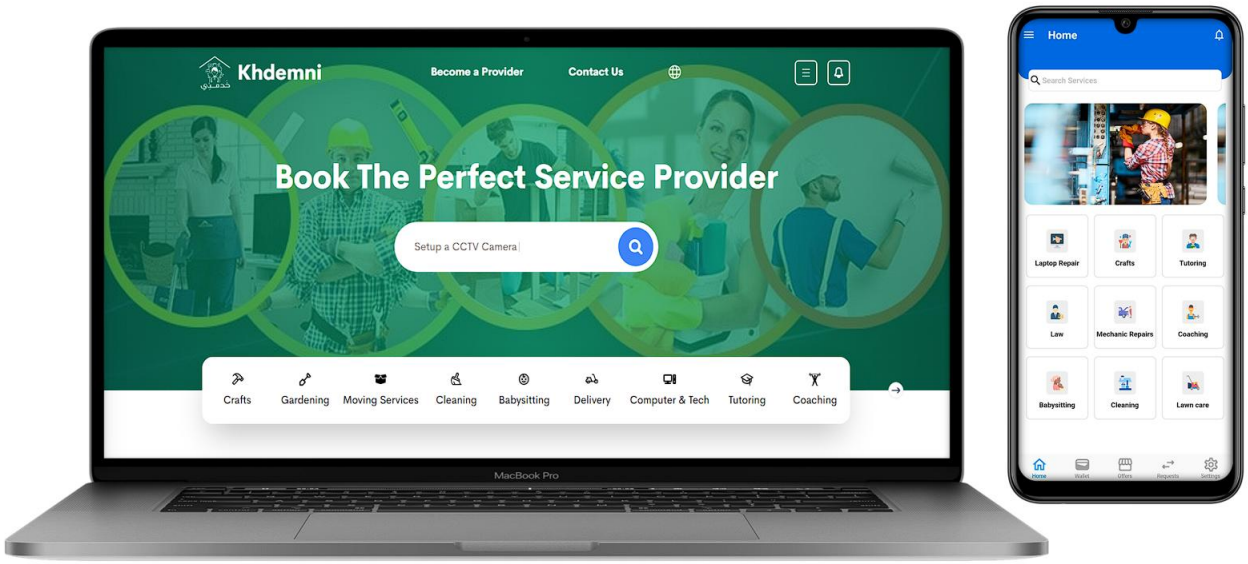


Figure 4.20: Home Page Platform

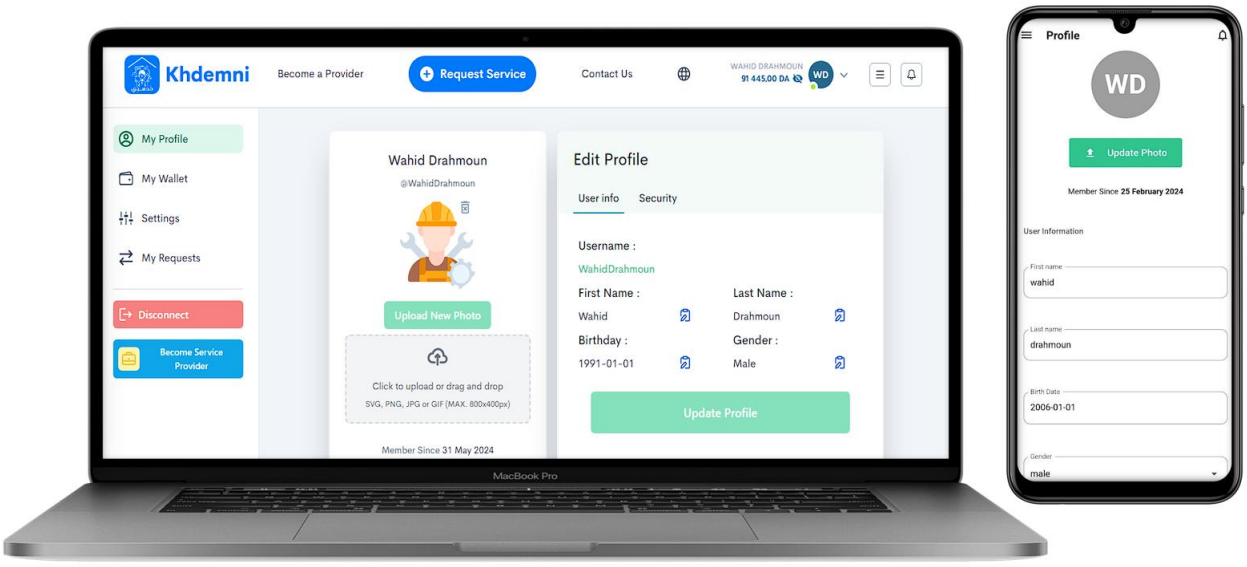


Figure 4.21: Profile In Platform

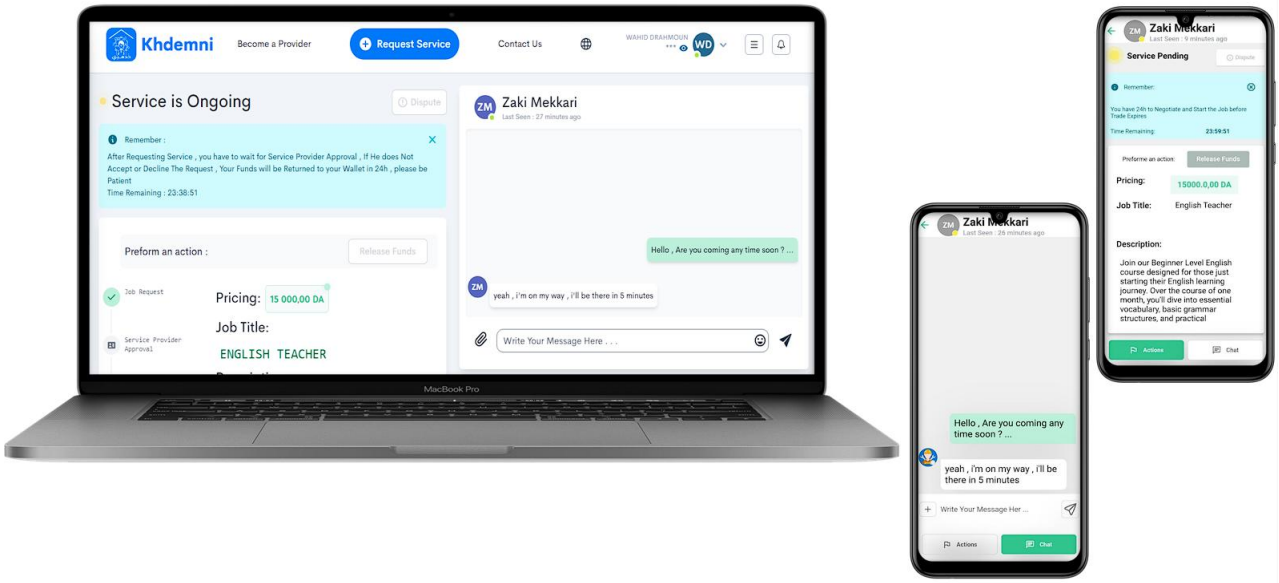


Figure 4.22: Platform

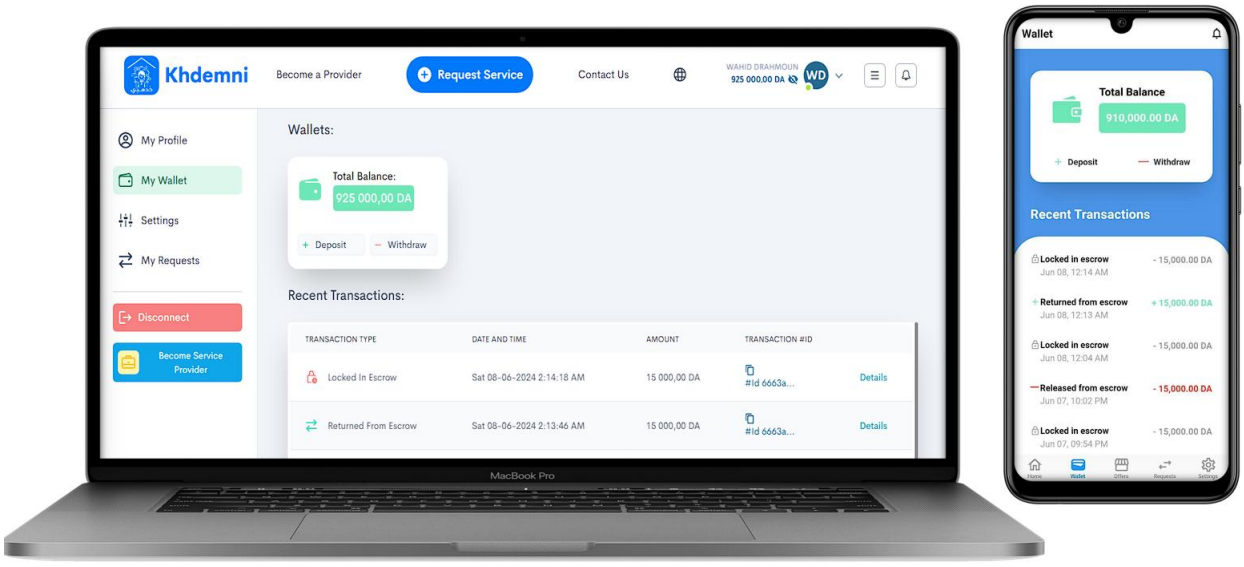


Figure 4.23: Wallet In Platform

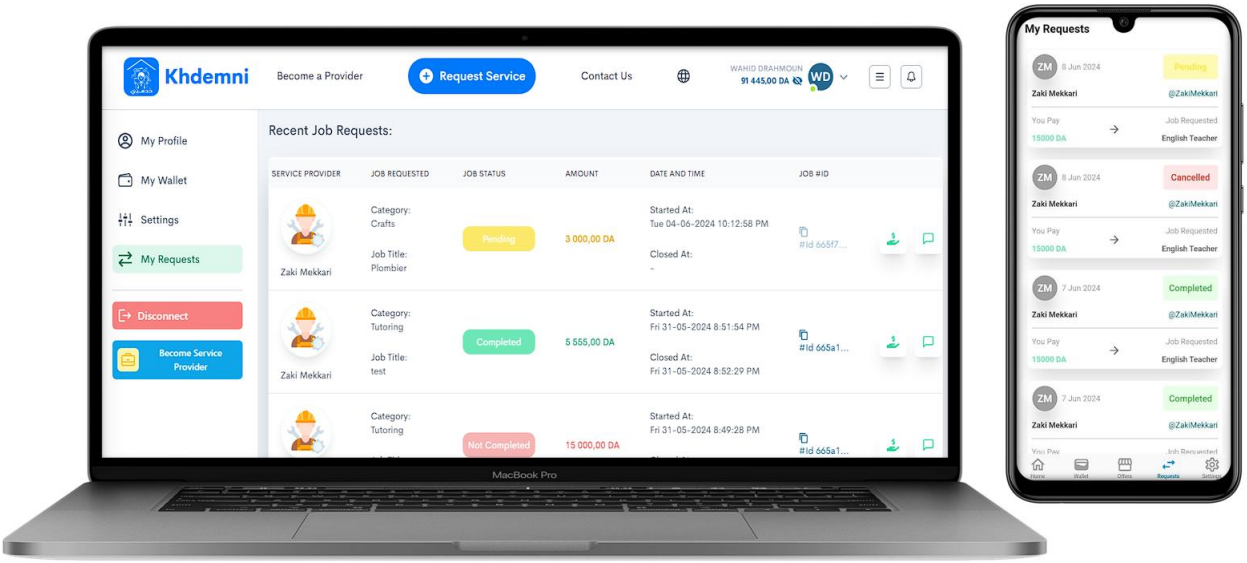


Figure 4.24: Requests in Platform

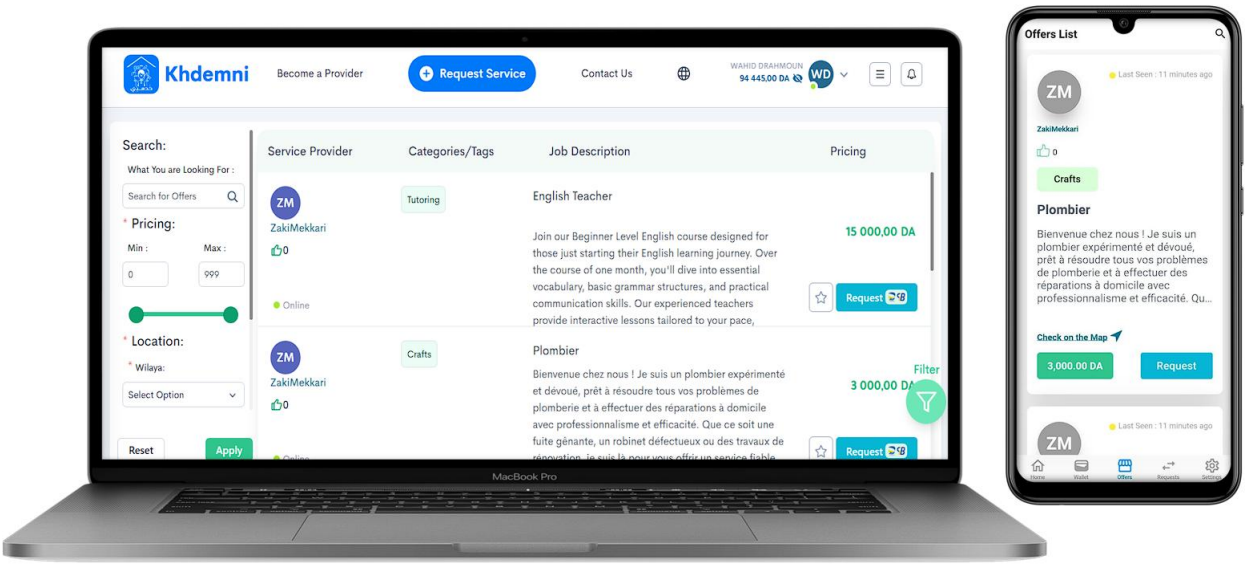


Figure 4.25: Offers In Platform

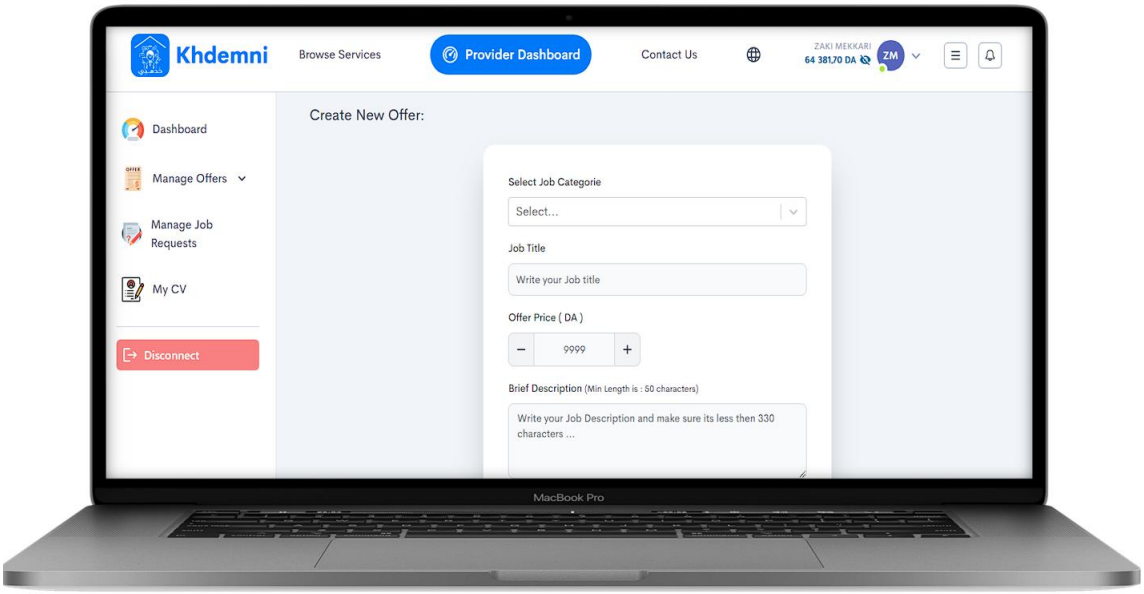


Figure 4.26: Create New Offer In Platform

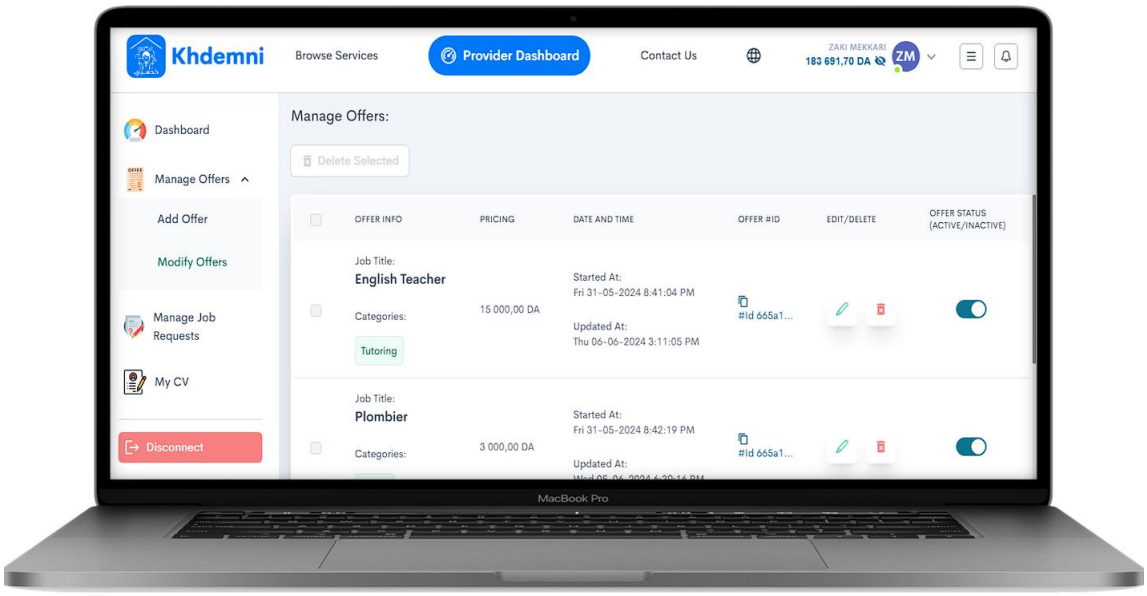
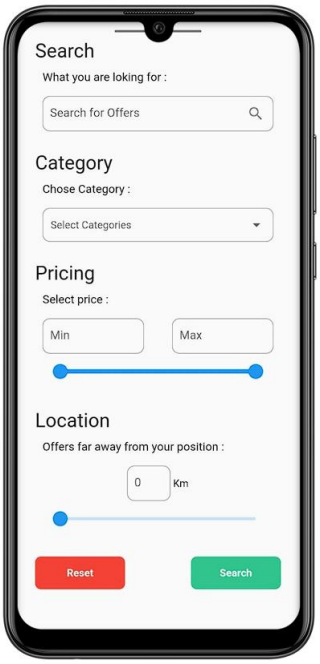


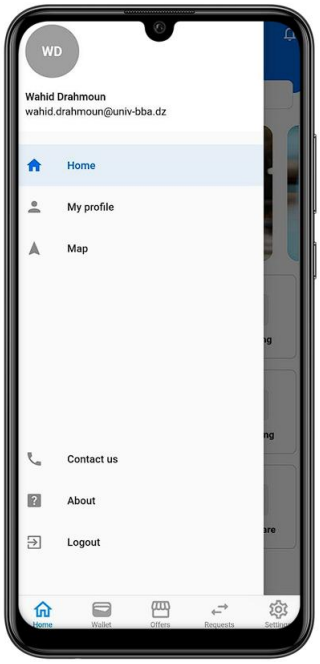
Figure 4.27: Modify Offer In Platform



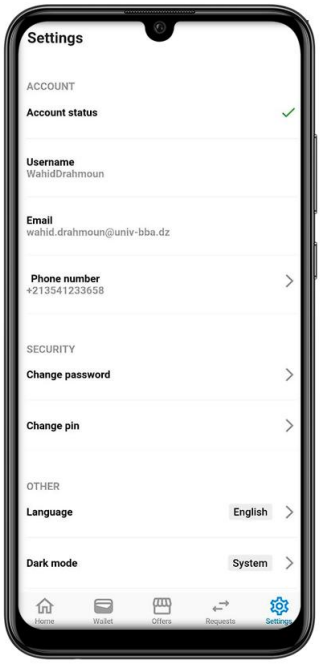
(a) Map In Mobile



(b) Search In Mobile



(c) Draw In Mobile



(d) Settings In Mobile

Figure 4.28: Mobile

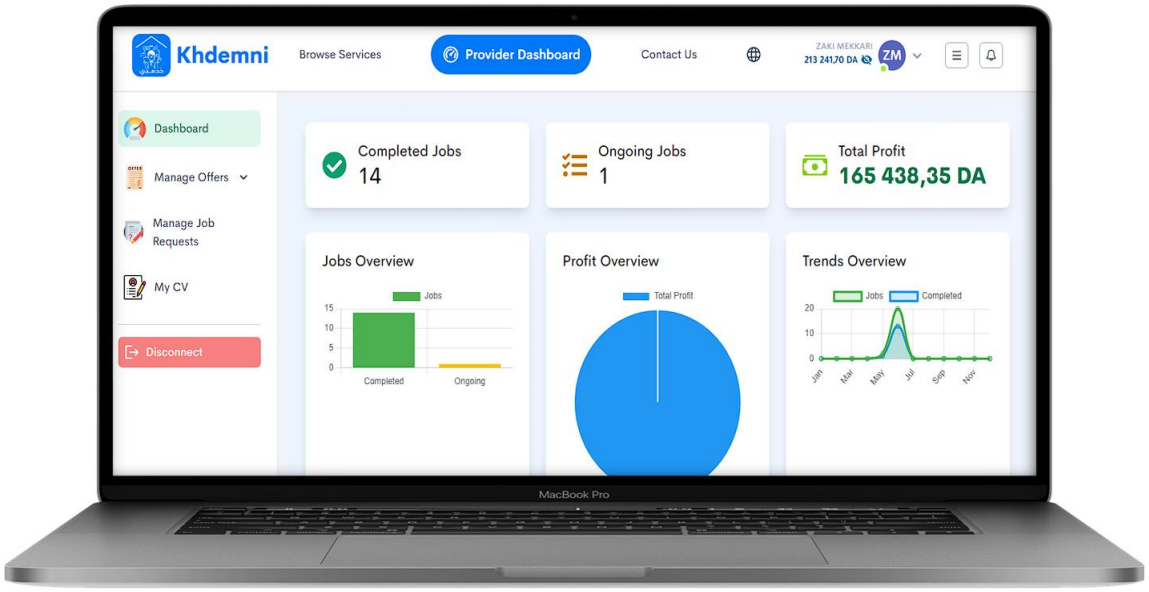


Figure 4.29: Dashboard In web application

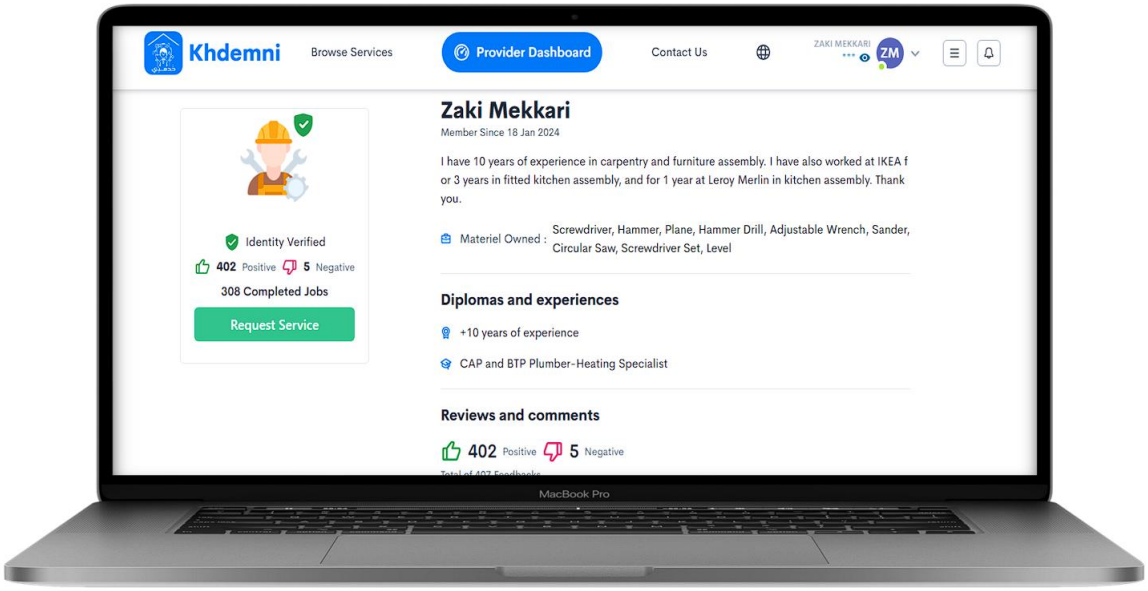


Figure 4.30: Service provider In application

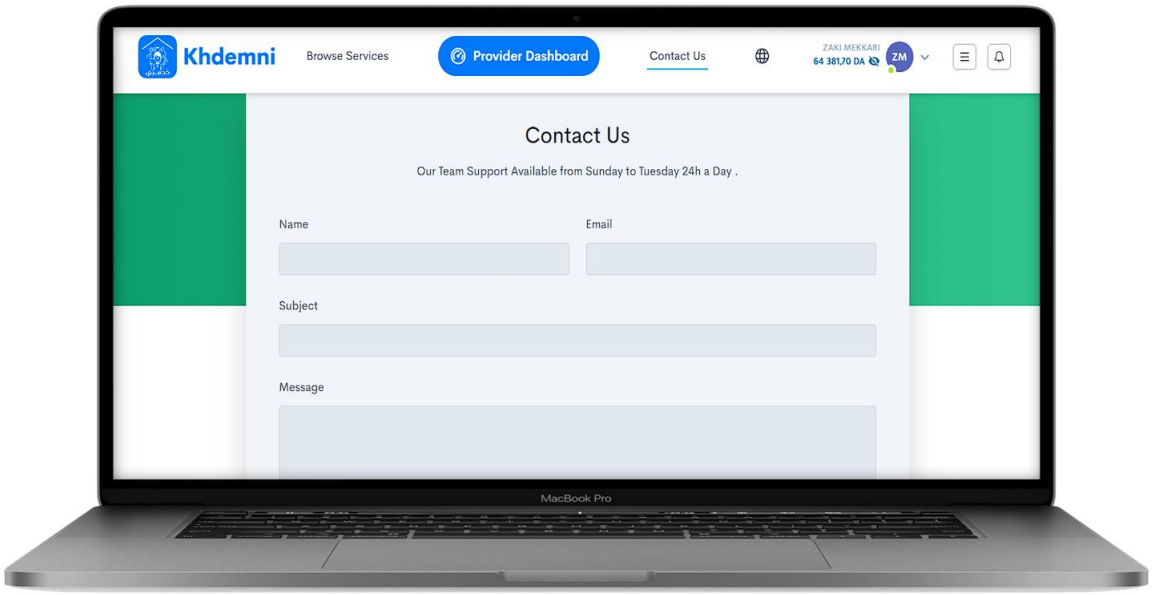


Figure 4.31: Contact Us In application

General Conclusion

In conclusion of this thesis, we seek to contribute to the development of an effective platform for providing services in a safe and innovative way, specifically designed for the Algerian market. Through a comprehensive literature review, we highlighted the guarantee and online payment system and its working mechanism. In addition, we conducted a comparison and study of similar platforms, which allowed us to uncover the challenges that users face, such as payment issues and intense competition.

Based on the research results, we defined and designed the platform requirements, focusing on functional and non-functional aspects to ensure user satisfaction. The database is also designed and modeled to store and retrieve data efficiently. We used UML diagrams to model the system explicitly. The implementation and testing phase followed the agile methodology, using programming languages and frameworks such as MongoDB, Node.js, and React.js. We ran tests using Jest to verify the functionality and stability of the system.

This thesis provided insights on how to integrate assurance and service provisioning to create a secure service delivery platform. The platform aims to empower service providers and provide high-quality services while ensuring a financial system that protects funds until the service is completed, and provides new and fast search methods. Through these efforts, we seek to create a startup that meets the needs of service providers and customers. Further and continuous improvements based on user feedback will be necessary to achieve the objectives of this memorandum and ensure the startup's long-term success.

References

[1] S. Price, Escrow Principles and Practice. Ashley Crown Systems, Incorporated, 2005. [Online]. Available: https://books.google.dz/books?id=jgbCPAAACAAJ

[2] Escrow documentation:. Retrieved from https://www.investopedia.com/terms/i/in-escrow.asp.

[3] Z. Pólkowski and I. C. Deaconescu, "Online payment systems," Zeszyty Naukowe Dolnośląskiej Wyższej Szkoły Przedsiębiorczości i Techniki. Studia z Nauk Technicznych, 2013.

[4] Who are we (n.d), accessed on:17-02-2024. [Online]. Available: https://www.freehali.com/qui-sommes-nous/

[5] Frequently Asked Questions (n.d), accessed on:17-02-2024. [Online]. Available: https://www.freehali.com/faq/

[6] farenow Documentation (n.d), accessed on:17-02-2024. [Online]. Available: https://www.farenow.com/faq

[7] How Does Upwork Work? (n.d), accessed on:17-02-2024. [Online]. Available: https://www.upwork.com/resources/is-upwork-worth-it#what-is-upwork

[8] How Does Upwork Work? (n.d), accessed on:17-02-2024. [Online]. Available: https://www.upwork.com/infographics/how-does-upwork-work

[9] Upwork Revenue, User, and Growth Statistics (2022), April 18, 2024, accessed on:17-02-2024. [Online]. Available: https://www.luisazhou.com/blog/upwork-revenue/

[10] Le Guile du Freehaleur (n.d), accessed on:17-02-2024. [Online]. Available: https:

//www.freehali.com/freehali-assets/uploads/2017/11/Guide-Freehaleurs-V0.1-1.pdf

[11] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries et al., "Manifesto for agile software development," 2001.

[12] K. Schwaber and J. Sutherland, "The scrum guide. the definitive guide to scrum: The rules of the game. scrum. org," October, vol. 2, p. 17, 2017.

[13] S. Lean, "Agile software development," 2013.

[14] L. T. Heeager, P. Svejvig, and B. R. Schlichter, "How agile methods inspire project management-the half double initiative," 2016.

[15] A. Moniruzzaman and D. S. A. Hossain, "Comparative study on agile software development methodologies," arXiv preprint arXiv:1307.3356, 2013.

[16] StarUML (n.d), accessed on:2-03-2024. [Online]. Available: https://staruml.io/

[17] R. Elmasri and S. B. Navathe, "Fundamentals of database systems 7th ed." 2016.

[18] Mozilla developer network (n.d). Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTML.

[19] Mozilla developer network (n.d). Retrieved from https://developer.mozilla.org/en-US/docs/Web/CSS.

[20] dart (n.d). Retrieved from https://dart.dev/.

[21] Tailwind official documentation (n.d). Retrieved from https://tailwindcss.com/docs/browser-support.

[22] Flutter official documentation (n.d). Retrieved from https://docs.flutter.dev/resources/faq.

[23] V. Subramanian, Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node, 1st ed. Apress, 2017.

[24] Why redux toolkit is how to use redux today (n.d). Retrieved from https://redux-toolkit.js.org/introduction/why-rtk-is-redux-today#what-is-redux-toolkit.

[25] socket.io (n.d). Retrieved from https://socket.io/.

[26] What is axios (n.d). Retrieved from https://axios-http.com/docs/intro.

[27] Mozilla developer network (n.d). Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[28] Introduction to json web tokens (n.d). Retrieved from https://jwt.io/introduction.

[29] Visual studio code (n.d). Retrieved from https://code.visualstudio.com/.

[30] Postman (n.d). Retrieved from https://www.postman.com/.

[31] Vite (n.d). Retrieved from https://vitejs.dev/.

[32] Git (n.d). Retrieved from https://git-scm.com/.

[33] Github (n.d). Retrieved from https://github.com/.

[34] Docker builds: Now lightning fast (n.d). Retrieved from https://www.docker.com/.

[35] A. Leff and J. T. Rayfield, "Web-application development using the model/view/controller design pattern," in Proceedings fifth ieee international enterprise distributed object computing conference. IEEE, 2001, pp. 118–127.

[36] MERN Stack Development For Your Project, March 19, 2024, accessed on:17-04-2024. [Online]. Available: https://markovate.com/blog/mern-stack-development/

الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي و البحث العلمي

جامعة محمد البشير الابراهيمي برج بوعريريج

عنوان المشروع:

Peer to peer secure escrow service plateform

مشروع لنيل شهادة مؤسسة ناشئة في اطار القرار الوزاري 1275

صورة العلامة التجارية

الاسم التجاري

خدمني Khdemni

السنة الجامعية

2024_ 2023

بطاقة معلومات:

حول فريق الاشراف وفريق العمل

1- فريق الاشراف:

فريق الاشراف	
المشرف الرئيسي (01): عتيقة بداوي	التخصص: إعلام آلي
المشرف الرئيسي (01):	التخصص: .....
المشرف المساعد:	التخصص: .....

2- فريق العمل:

فريق المشروع	التخصص	الكلية
الطالب: رحاحلة أسامة	تكنولوجيا المعلومات والإتصال	الرياضيات و الاعلام الآلي
الطالب: درحمون وحيد	الشبكة والوسائط المتعددة	الرياضيات و الاعلام الآلي
الطالب: مكاري زكارياء	الشبكة والوسائط المتعددة	الرياضيات و الاعلام الآلي
الطالب: بونابي إلياس	الشبكة والوسائط المتعددة	الرياضيات و الاعلام الآلي

فهرس المحتويات

المحور الأول: تقديم المشروع 81
فكرة المشروع (الحل المقترح) 81
القيم المقترحة 81
فريق العمل 82
أهداف المشروع 83
المحور الثاني: الجوانب الابتكارية 84
المحور الثالث: التحليل الاستراتيجي للسوق 85
المحور الرابع: خطة الإنتاج والتنظيم 87
خطة الإنتاج 87
خطة التنظيم 87
المحور الخامس: الخطة المالية PLAN FINANCIER 87
التكاليف الثابتة: 87
التكاليف المتغيرة: 88
رقم الأعمال : 88
المحور السادس : النموذج الاولي التجريبي 89
الجداول والأشكال 97

LISTE OF FIGURES قائمة الصور

FIGURE1 SIGN IN PLATFORM *89*
FIGURE2 SIGN UP PLATFORM *89*
FIGURE 3 HOME PAGE PLATFORM *90*
FIGURE 4 PROFILE IN PLATFORM *90*
FIGURE 5 SERVICE TRACKING IN PLATFORM *91*
FIGURE 6 WALLET IN PLATFORM *91*
FIGURE 7 REQUESTS IN PLATFORM *92*
FIGURE 8 OFFERS IN PLATFORM *92*
FIGURE 9 CREATE NEW OFFER IN PLATFORM *93*
FIGURE10 MODIFY OFFER IN PLATFORM ..... ***ERREUR ! SIGNET NON DEFINI.***
FIGURE11 MODIFY OFFERS IN PLATFORM *93*
FIGURE 12 DRAWER IN MOBILE *94*
FIGURE 13 SEARCH IN MOBILE *94*
FIGURE 14 SETTINGS IN MOBILE *94*
FIGURE 15 MAP IN MOBILE *94*
FIGURE 16 DASHBOARD IN WEB APPLICATION *95*
FIGURE 17 SERVICE PROVIDER IN APPLICATION *96*
FIGURE 18 CONTACT US IN WEB APPLICATION *96*

المحور الأول: تقديم المشروع

فكرة المشروع (الحل المقترح)

في عصرنا الحالي المليء بالحركة، تعتبر الراحة أمرًا لا غنى عنه. ومع ذلك، قد يكون من الصعب في بعض الأحيان العثور على الخدمة المناسبة في الوقت المناسب. هذا هو المكان الذي يأتي فيه دورنا.

تخيل مكانًا حيث يمكنك الوصول إلى مجموعة كبيرة من الخدمات المختلفة، تم تصميمها خصيصًا لتلبية احتياجاتك، وكل ذلك دون الحاجة لمغادرة منزلك. سواء كنت تبحث عن خدمة حرفية لتجديد منزلك، أو رعاية طبية لأحد أفراد عائلتك، أو حتى دروسًا خصوصية أو المساعدة في تسهيل حياتك اليومية، فإن "خدمني" هنا لتلبية جميع هذه الاحتياجات، وأكثر من ذلك.

القيم المقترحة

1. **إمكانية الوصول:** توفير وصول سهل ومريح إلى مجموعة متنوعة من الخدمات المنزلية لتلبية الاحتياجات المتنوعة للمستخدمين.
2. **الموثوقية:** ضمان جودة وموثوقية مقدمي الخدمات المتوفرين على المنصة لضمان ثقة المستخدم.
3. **سهولة الاستخدام:** تبسيط عملية البحث والحجز وتقديم الخدمات المنزلية، وبالتالي توفير تجربة خالية من المتاعب.
4. **التنوع:** تقديم مجموعة واسعة من الخدمات المنزلية المؤقتة لتلبية الاحتياجات المختلفة، بدءًا من الأعمال اليدوية وحتى الرعاية الطبية والتدريس وغير ذلك الكثير.
5. **التخصيص:** السماح للمستخدمين بتخصيص طلبات الخدمة الخاصة بهم بناءً على تفضيلاتهم واحتياجاتهم المحددة.
6. **الشفافية:** توفير معلومات واضحة حول مقدمي الخدمات والأسعار والمراجعات لمساعدة المستخدمين على اتخاذ قرارات مستنيرة.
7. **الأمان:** ضمان أمن المعاملات المالية والبيانات الشخصية والتفاعلات بين المستخدمين ومقدمي الخدمات.
8. **المجتمع:** تشجيع الشعور بالانتماء للمجتمع من خلال تعزيز العلاقات الإيجابية بين المستخدمين ومقدمي الخدمة.

9. **الابتكار:** اعتماد ودمج التقنيات والميزات الجديدة باستمرار لتحسين تجربة المستخدم.

10. **المسؤولية الاجتماعية:** تعزيز الممارسات الأخلاقية والمستدامة، ودعم مقدمي الخدمات المحليين في أنشطتهم.

ومن خلال تسليط الضوء على هذه القيم، يمكن لمنصتنا أن تضع نفسها كمساحة تكون فيها الجودة والتنوع والموثوقية والراحة في الخدمات المنزلية في قلب الاهتمامات. يمكن إيصال هذه القيم في إستراتيجيتنا التسويقية وميثاقنا التحريري وفي أي تبادل مع مستخدمينا، وبالتالي تعزيز الثقة والولاء تجاه منصتنا.

فريق العمل

- **الطالب 1 :** رحاحلة أسامة **التخصص :** تكنولوجيا الإعلام والاتصال

يتمثل دور الطالب 1 مدير المشروع هو المسؤول عن الإدارة العامة للمشروع، من التخطيط إلى الانتهاء، وضمان تنسيق والاشراف على تنفيذ المشروع مع فريق العمل.

- **الطالب 2 :** درحمون وحيد **التخصص :** شبكات ووسائط متعددة

يتمثل دور الطالب 2 مطور الويب في تصميم وإنشاء مواقع الويب باستخدام لغات البرمجة وتقنيات الويب، مع التأكد من أنها وظيفية وسهلة الاستخدام وتلبي احتياجات العملاء.

- **الطالب 3 :** مكاري زكرياء **التخصص :** شبكات ووسائط متعددة

يتمثل دور الطالب 3 مسؤول قاعدة البيانات في إدارة قواعد البيانات وصيانتها، وضمان أدائها وأمانها وتوافرها. تصميم وتطوير هياكل قاعدة البيانات لتلبية الاحتياجات المحددة للمشروع، مما يضمن كفاءتها وقابلية التوسع,

- **الطالب 4 :** بونابي الياس **التخصص :** شبكات ووسائط متعددة

يتمثل دور الطالب 4 مطور تطبيقات iOS/Android في تصميم وتطوير تطبيقات الهاتف المحمول باستخدام اللغات والأدوات الخاصة بالنظام الأساسي، مما يضمن أنها عملية وبديهية وتلبي احتياجات المستخدم

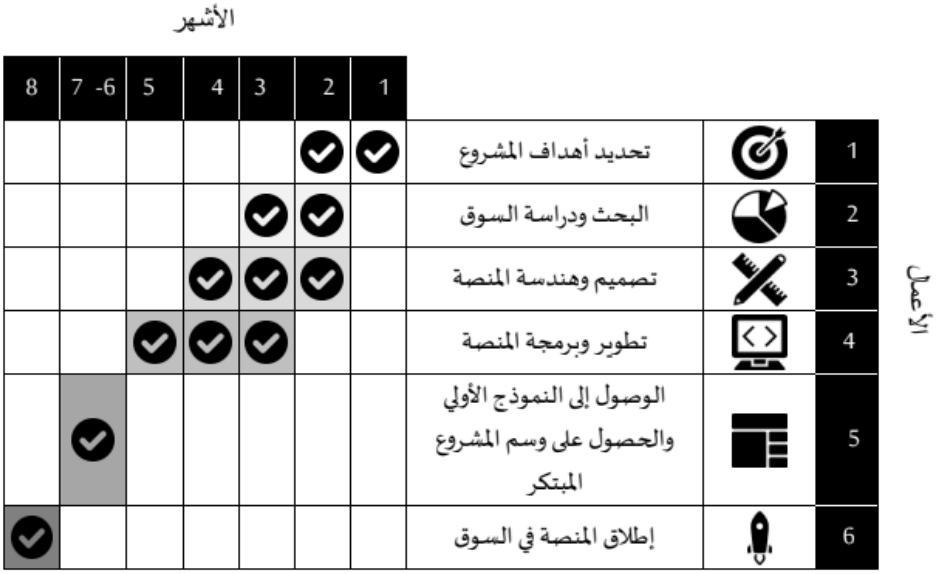
أهداف المشروع

أهداف المشروع هي الأهداف العامة وطويلة المدى التي نسعى لتحقيقها من خلال الانتهاء من المشروع. غالبًا ما تتماشى هذه الأهداف مع رؤية المنظمة ورسالتها، ويمكن أن تشمل جوانب مثل خلق القيمة، وتحسين الكفاءة، ورضا العملاء، وما إلى ذلك. فيما يلي بعض الأمثلة على أهداف المشروع لمنصة الخدمات المنزلية:

- ✓ **إنشاء فرص عمل:** توفير منصة تسمح لمقدمي الخدمات بالعثور على وظائف مؤقتة، مما يمكن أن56 يساعد في تقليل البطالة وتعزيز الاقتصاد المحلي.
- ✓ **تسهيل الوصول على الخدمات:** إنشاء منصة سهلة الاستخدام ويمكن الوصول إليها للسماح للعملاء بالعثور على الخدمات التي يحتاجون إليها وحجزها بسهولة وبالتالي تحسين نوعية حياتهم.
- ✓ **تقديم خدمات عالية الجودة:** الالتزام بتقديم خدمات عالية الجودة من خلال العمل مع مقدمي الخدمات المؤهلين ومراقبة رضا العملاء باستمرار.
- ✓ **تعزيز الأمن والثقة:** تنفيذ إجراءات أمنية قوية لحماية العملاء وأصولهم، وإقامة علاقة ثقة معهم.
- ✓ **دعم الاقتصاد المحلي:** التعاون مع مقدمي الخدمات المحليين لدعم الاقتصاد المحلي وتقديم الخدمات التي تلبي الاحتياجات المحددة للمجتمع.
- ✓ **الابتكار والتطور:** الاستمرار في الابتكار والتطور للبقاء في طليعة صناعة الخدمات المنزلية وتلبية احتياجات العملاء المتغيرة.
- ✓ **تعزيز الاستدامة:** اعتماد ممارسات مستدامة في تقديم الخدمات للحد من الأثر البيئي والمساهمة في مستقبل أكثر استدامة.
- ✓ **تعزيز رضا العملاء:** نهدف إلى تحقيق مستوى عالٍ من رضا العملاء من خلال تقديم خدمات تتجاوز توقعاتهم والاستجابة بفعالية لاحتياجاتهم واهتماماتهم.

ومن خلال اعتماد نهج يركز على المجتمع، يمكن للمنصة أن تصبح لاعبا إيجابيا في الحياة اليومية للجزائريين من خلال تلبية احتياجاتهم مع المساهمة في التنمية الاجتماعية والاقتصادية للمجتمع المحلي.

1. جدول زمني لتحقيق المشروع :



المحور الثاني: الجوانب الابتكارية

لجعل منصة الخدمات المنزلية الخاصة بنا مبتكرة في الجزائر، هنالك بعض الجوانب التي يجب مراعاتها:

1. **نظام التحقق من مقدمي الخدمات :** دمج عملية صارمة للتحقق من مقدمي الخدمات لضمان كفاءتهم وموثوقيتهم وشرعيتهم، مما يوفر ضمانًا إضافيًا للمستخدمين.
2. **تخصيص الخدمات:** توفير نظام حيث يمكن للمستخدمين تخصيص طلبات الخدمة الخاصة بهم بناءً على احتياجاتهم المحددة والفترات الزمنية وتفضيلات اللغة وما إلى ذلك.
3. **تقنية تحديد الموقع الجغرافي:** دمج تحديد الموقع الجغرافي للسماح للمستخدمين بالعثور بسرعة على مقدمي الخدمة القريبين، مما يسهل الاستجابة لطلبات الطوارئ أو الاحتياجات بضغطة واحدة.
4. **التقييم الشفاف:** إنشاء نظام تصنيف شفاف وموثوق، حيث يمكن للمستخدمين تقديم تعليقات صادقة على مقدمي الخدمات، وبالتالي بناء الثقة وجودة الخدمات.
5. **الدردشة في الوقت الفعلي:** توفير وظيفة الدردشة في الوقت الفعلي للتواصل المباشر بين المستخدمين ومقدمي الخدمات، مما يسهل التخطيط المسبق والتوضيح.

6. **خيارات دفع مرنة:** تقديم طرق دفع متنوعة وآمنة، تتكيف مع السياق المحلي، لتوفير المزيد من المرونة للمستخدمين.
7. **حلول متعددة اللغات:** تأكد من أن منصتنا تدعم لغات محلية متعددة لاستيعاب مجموعة سكانية متنوعة لغويًا في الجزائر.
8. **دعم العملاء المتقدم:** إنشاء خدمة عملاء سريعة الاستجابة وواسعة المعرفة يمكنها حل المشكلات بسرعة وتلبية احتياجات المستخدم.
9. **الشراكات المحلية:** إقامة شراكات مع أصحاب المصلحة المحليين، مثل الجمعيات أو الشركات، لتقديم عروض أو خدمات خاصة مصممة خصيصًا لتلبية الاحتياجات الخاصة.
10. **المسؤولية الاجتماعية:** دمج الممارسات المسؤولة اجتماعيًا وبيئيًا وأخلاقيًا لتعزيز صورة منصتنا.

يمكن لهذه الجوانب المبتكرة أن تساعد في تمييز منصتنا وتوفير تجربة مستخدم محسنة وتلبية الاحتياجات المحددة لسوق الخدمات المنزلية الجزائري.

المحور الثالث: التحليل الاستراتيجي للسوق

قد يشمل السوق والعملاء المستهدفون للمنصة الخاصة بنا التي تقدم مجموعة متنوعة من الخدمات المنزلية المؤقتة ما يلي:

1. **الأفراد والأسر:**

يمثل هذا القطاع الأفراد الذين يبحثون عن خدمات مؤقتة لاحتياجاتهم الشخصية في المنزل، مثل الإصلاحات والرعاية الطبية المنزلية والدروس الخصوصية، وما إلى ذلك.

2. **العائلات المزدحمة:**

• قد تبحث العائلات ذات الجداول الزمنية المزدحمة والالتزامات العائلية عن حلول عملية لإدارة منازلهم أو الحصول على الرعاية لأحبائهم أو تلقي خدمات توفر الوقت.

3. **كبار السن أو الأشخاص ذوي القدرة المحدودة على الحركة:**

• يمكن لكبار السن أو أولئك الذين يعانون من صعوبات في الحركة الاستفادة من الخدمات المنزلية، مثل الرعاية التمريضية أو المساعدة المنزلية أو إصلاحات محددة.

4. **المهنيين المستقلين:**

• قد يجد العاملون لحسابهم الخاص، مثل الحرفيين، ومتخصصي الصحة المنزلية، والمدربين الشخصيين، وما إلى ذلك، أن منصتنا وسيلة فعالة للترويج لخدماتهم والوصول إلى قاعدة عملاء أوسع.

5. **الطلاب والمتعلمون:**

• قد يكون الطلاب أو الأشخاص الذين يتطلعون إلى تعلم مهارات جديدة مهتمين بالدروس الخصوصية أو التدريب المنزلي المقدم على منصتك.

لذا فإن السوق المستهدف واسع جدًا، ويصل إلى مجموعة واسعة من الأشخاص ذوي الاحتياجات المنزلية المتنوعة. إن استهداف هذه القطاعات المختلفة من خلال تسليط الضوء على تنوع خدماتنا وعمليتها وموثوقيتها يمكن أن يساعد في جذب العملاء والاحتفاظ بهم.

بالطبع، فيما يلي نظرة عامة على السوق المستهدف لمقدمي خدمات النظام الأساسي الخاص بنا:

1. **الحرفيين والعمال المهرة:** تستهدف الحرفيين المهرة مثل السباكين والكهربائيين والنجارين والرسامين الذين يسعون إلى تقديم خدماتهم للعملاء بشكل مباشر.

2. **أخصائيو الصحة المنزلية:** الممرضون ومقدمو الرعاية والمعالجون الذين يرغبون في تقديم خدمات الرعاية المنزلية الخاصة بهم، والتي تستهدف كبار السن أو أولئك الذين يحتاجون إلى رعاية خاصة في المنزل.

3. **المعلمون والمدربون الشخصيون:** المعلمون والمدربون الرياضيون والمدربون الذين يرغبون في تقديم دروس أو دورات تدريبية في المنزل للطلاب أو العملاء المهتمين.

4. **مقدمو الخدمات المتنوعة:** مقدمو الخدمات الذين يقدمون خدمات خاصة مثل التصوير الفوتوغرافي والطبخ المنزلي والموسيقى، بهدف الوصول إلى العملاء الذين يبحثون عن تجارب فريدة في المنزل.

5. **رواد الأعمال الشباب والعاملون لحسابهم الخاص:** رواد الأعمال الشباب الذين يسعون لبدء أو تطوير أنشطتهم من خلال تقديم مهاراتهم في المنزل، وبالتالي الاستفادة من الرؤية التي توفرها المنصة.

وبالتالي فإن السوق المستهدف لمقدمي الخدمات يمتد إلى المهنيين في مختلف القطاعات الذين يتطلعون إلى تقديم خدمات مؤقتة في المنزل. من خلال الترحيب بهم في منصتنا، فإنك تخلق فرصة لهم لتوسيع قاعدة عملائهم وجعل خدماتهم أكثر سهولة ووضوحًا.

المحور الرابع: خطة الإنتاج والتنظيم

خطة الإنتاج

تبدأ خطة الإنتاج بمرحلة البحث والتحليل، حيث يتم دراسة السوق واحتياجات العملاء المستهدفين، بالإضافة إلى تحليل المنافسين وتحديد الفرص المتاحة. ثم يتم الانتقال إلى مرحلة تصميم وتخطيط المنصة، حيث يتم تصميم واجهة وتجربة المستخدم ووضع خطة لتطوير البرمجيات والبنية التحتية ,

في مرحلة تطوير البرمجيات، يتم بناء الميزات الأساسية للمنصة وتطوير تطبيقات الهواتف المحمولة. بعد ذلك، تأتي مرحلة اختبار البرمجيات لضمان وظائف وأداء المنصة وضمان الأمان والسيولة في الاستخدام.

عقب الانتهاء من التطوير والاختبار، يتم الإطلاق التجريبي للمنصة لجمع الملاحظات من مجموعة محدودة من المستخدمين، ومن ثم يتم الإطلاق الرسمي مع تنفيذ حملات تسويقية لجذب المستخدمين ومقدمي الخدمات. تستمر عملية الإنتاج مع التطوير المستمر والصيانة الدورية، حيث يتم إطلاق تحديثات دورية وتحسينات وصيانة دورية لمعالجة المشكلات الأمنية والتقنية.

خطة التنظيم

يتكون الهيكل التنظيمي للمنصة من عدة أقسام، تبدأ بالإدارة العليا التي تشمل الرئيس التنفيذي (CEO) المسؤول عن الاستراتيجية العامة، والرئيس التنفيذي للتكنولوجيا (CTO) المسؤول عن الجوانب التقنية. يلي ذلك الفريق التقني، الذي يضم مهندسي البرمجيات لتطوير وصيانة المنصة، ومهندسي الشبكات لإدارة البنية التحتية، وخبراء الأمن السيبراني لحماية المنصة,

يضم فريق التسويق مدير التسويق الذي يقود الاستراتيجيات التسويقية، ومختصين في التسويق الرقمي وتحسين محركات البحث (SEO) لضمان الوصول الأمثل إلى الجمهور المستهدف. أما فريق الدعم وخدمة العملاء فيتكون من مدير الدعم الذي يدير فريق خدمة العملاء، وممثلين لدعم العملاء يقدمون المساعدة للمستخدمين.

الفريق الإداري يشمل المحاسبين الذين يديرون الشؤون المالية، والمستشارين القانونيين الذين يقدمون الاستشارات القانونية اللازمة لضمان الامتثال للقوانين.

المحور الخامس: الخطة المالية PLAN FINANCIER

التكاليف الثابتة:

- **تطوير موقع الويب:** التكلفة الأولية لتطوير موقع الويب الخاص بالمنصة، بما في ذلك التصميم والترميز والاختبار.

- **البنية التحتية:** الاستثمارات في الخوادم والاستضافة والبنية التحتية التكنولوجية الأخرى لضمان التشغيل السلس.

- **الامتثال القانوني والتنظيمي:** النفقات المتعلقة بالحصول على التراخيص والتصاريح اللازمة والامتثال للقوانين واللوائح الجزائرية.

- **التسويق والترويج:** الأموال المخصصة للحملة التسويقية والإعلان الرقمي واكتساب المستخدمين لجذب العملاء إلى المنصة.

- **الموظفون والرواتب:** التكاليف المرتبطة بتعيين فريق لصيانة النظام الأساسي ودعم العملاء وتطوير الأعمال.

التكاليف المتغيرة:

- **رسوم بوابة الدفع:** الرسوم التي يتم تكبدها عند تسهيل معاملات الدفع الآمنة بين المستقلين والعملاء.

- **دعم العملاء:** النفقات المتغيرة لتقديم خدمات دعم العملاء الفعالة لكل من العاملين لحسابهم

الخاص والعملاء.

- **ترقيات النظام الأساسي ي وصيانته:** التكاليف المستمرة للحفاظ على وظائف النظام الأساسي والأمان وتجربة المستخدم وتحسينها.

رقم الأعمال :

- ✓ سيتم إنشاء معدل رقم أعمال المنصة من خلال الرسوم المفروضة على المعاملات لناجحة بين مقدمي الخدمات والعملاء (من 8% إلى 25%). سيتم احتساب هذه الرسوم كنسبة مئوية من القيمة الإجمالية للخدمات المقدمة من خلال المنصة.
- ✓ لتقدير حجم الأعمال المحتمل ، نحتاج إلى النظر في عوامل مثل حجم سوق العمل الخدماتي في الجزائر ،

والعدد المتوقع من العاملين الأحرار والمقاولين الذاتيين المسجلين والعملاء على المنصة ، ومتوسط قيمة المعاملات. سيتم إجراء بحث وتحليل شامل للسوق للتنبؤ بدقة حجم الأعمال.

✓ من المهم ملاحظة أن نجاح النظام الأساسي وقدرته على جذب قاعدة مستخدمين كبيرة سيكون أمرا حاسما في تحديد رقم الأعمال الإجمالي. ستكون استراتيجيات التسويق الفعالة والواجهات سهلة الاستخدام ونظام الدفع الموثوق أمرا ضروريا ا لتعزيز الثقة وتشجيع حجم كبير من المعاملات.

المحور السادس : النموذج الاولي التجريبي

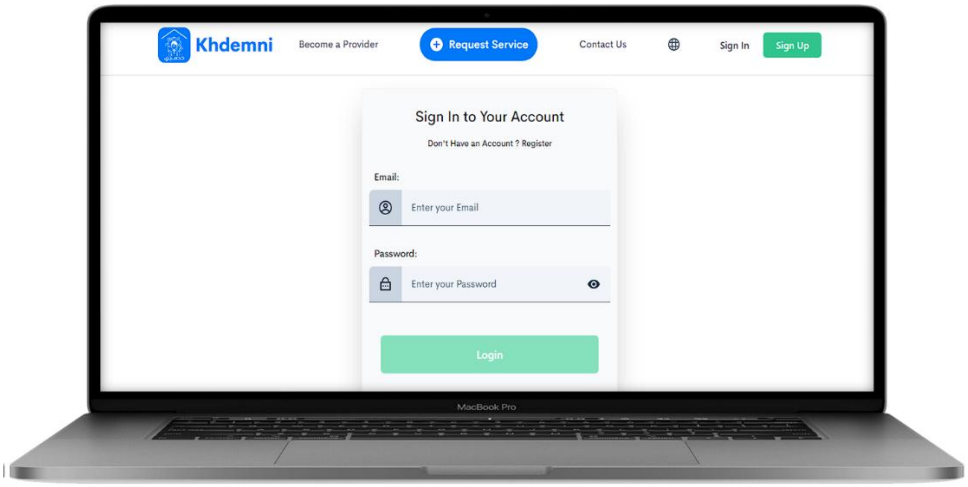


Figure1 Sign In Platform

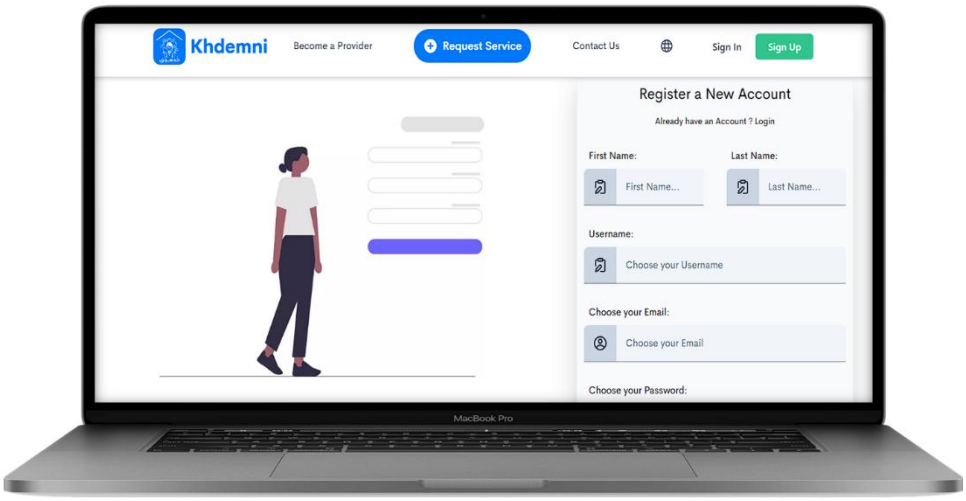


Figure2 Sign Up Platform

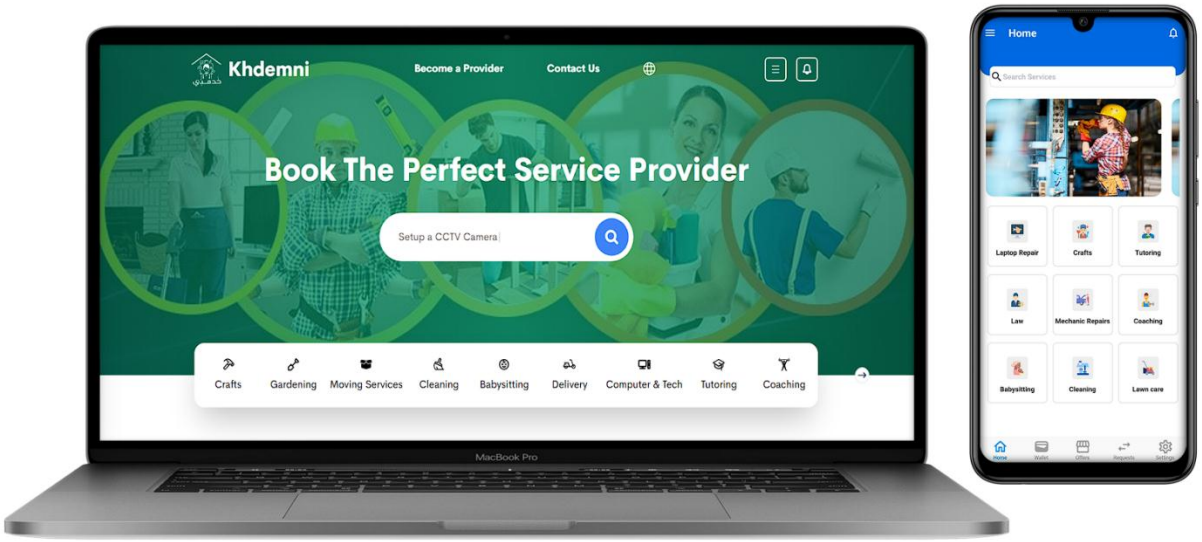


Figure 3 Home Page Platform

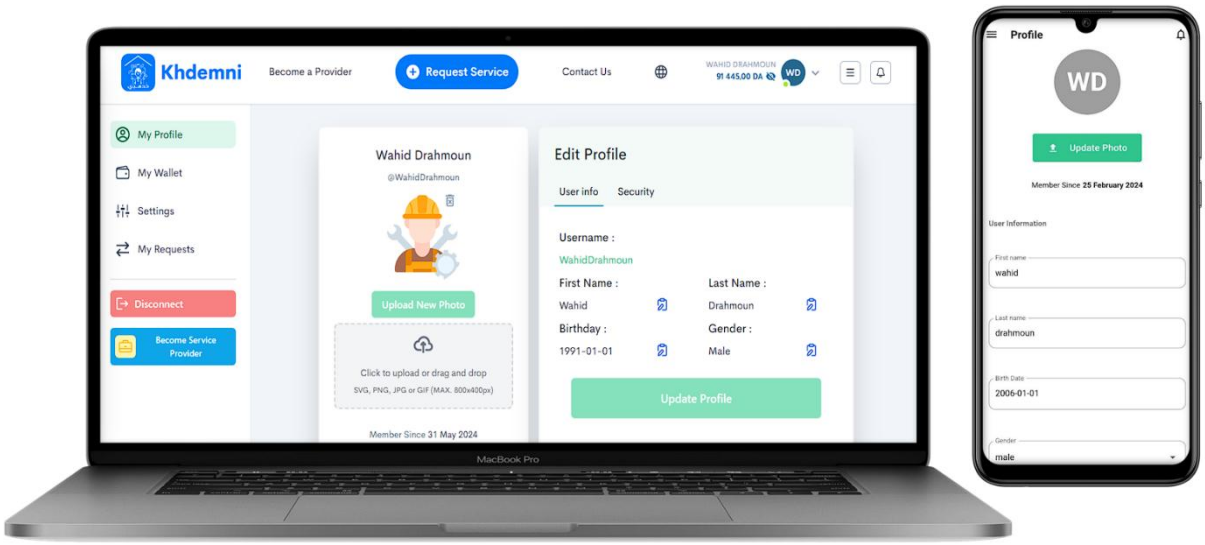


Figure 4 Profile In Platform

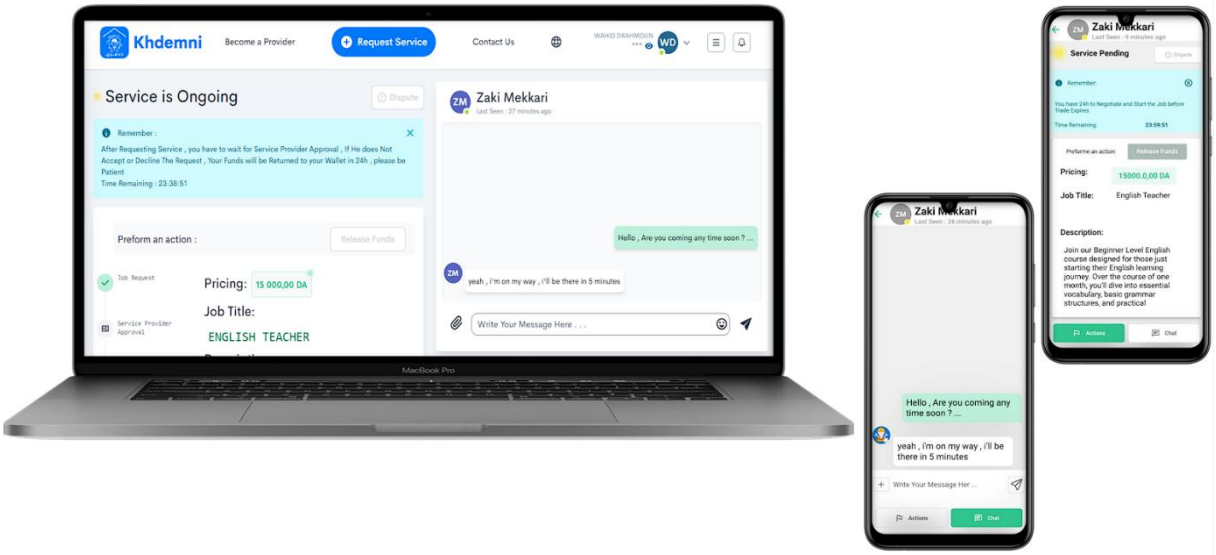


Figure 5 Service Tracking In Platform

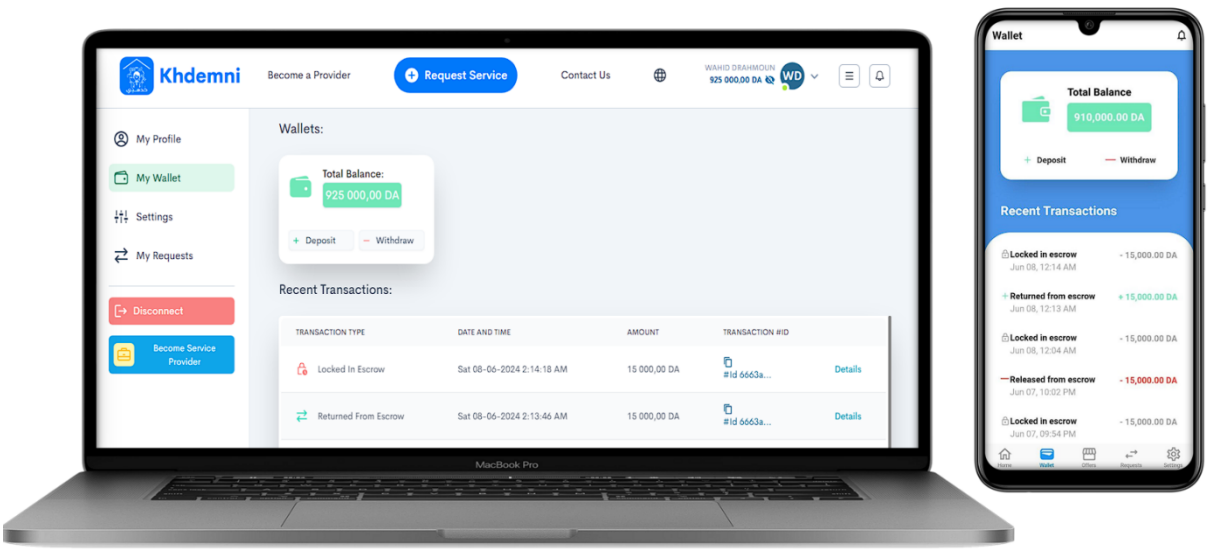


Figure 6 Wallet In Platform

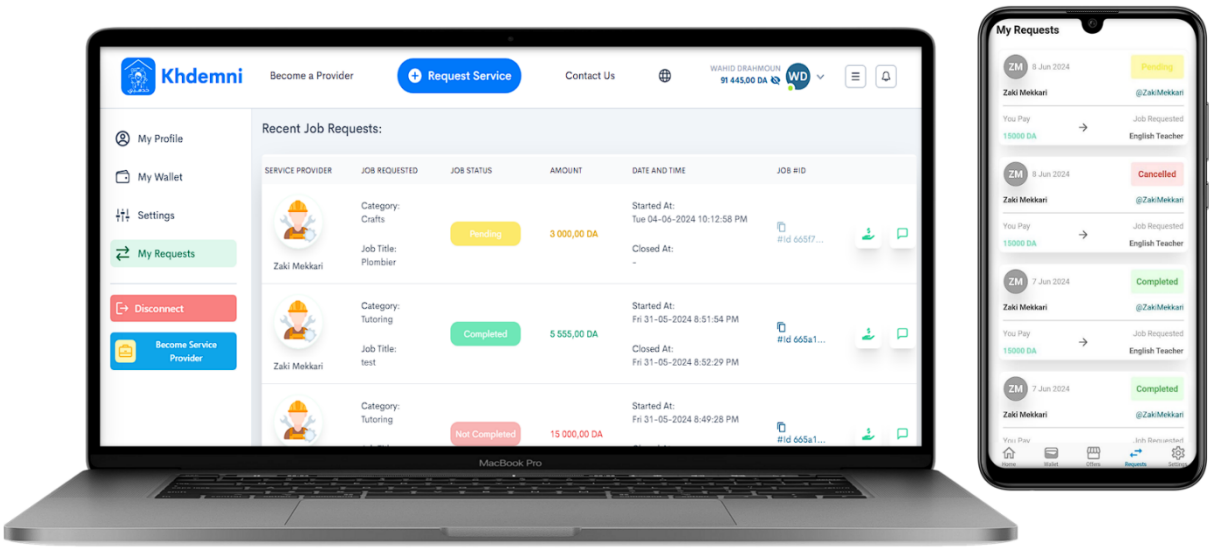


Figure 7 Requests in Platform

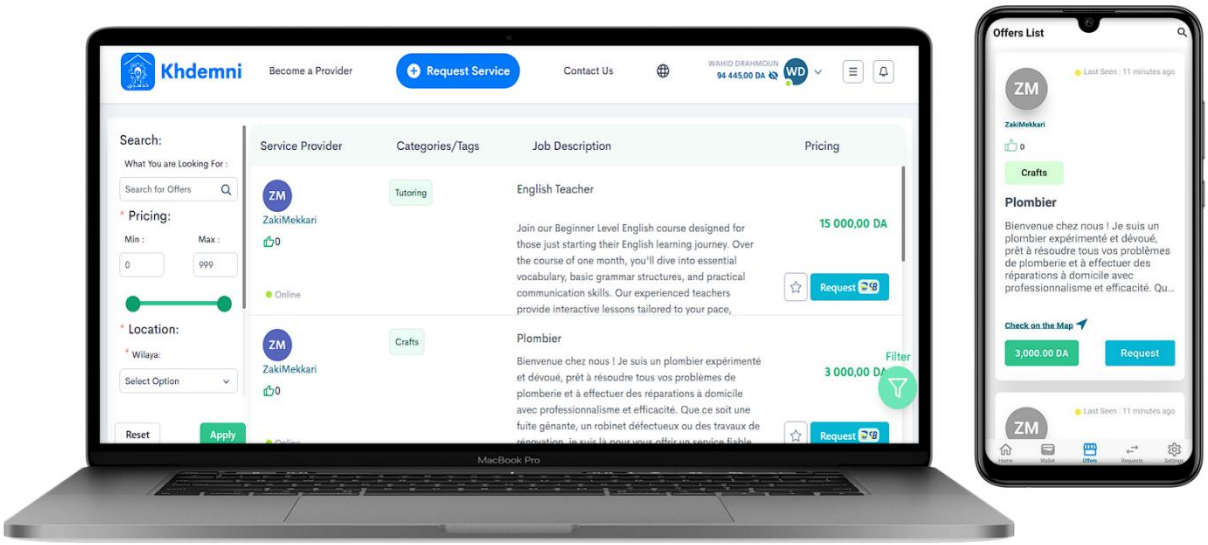


Figure 8 Offers In Platform

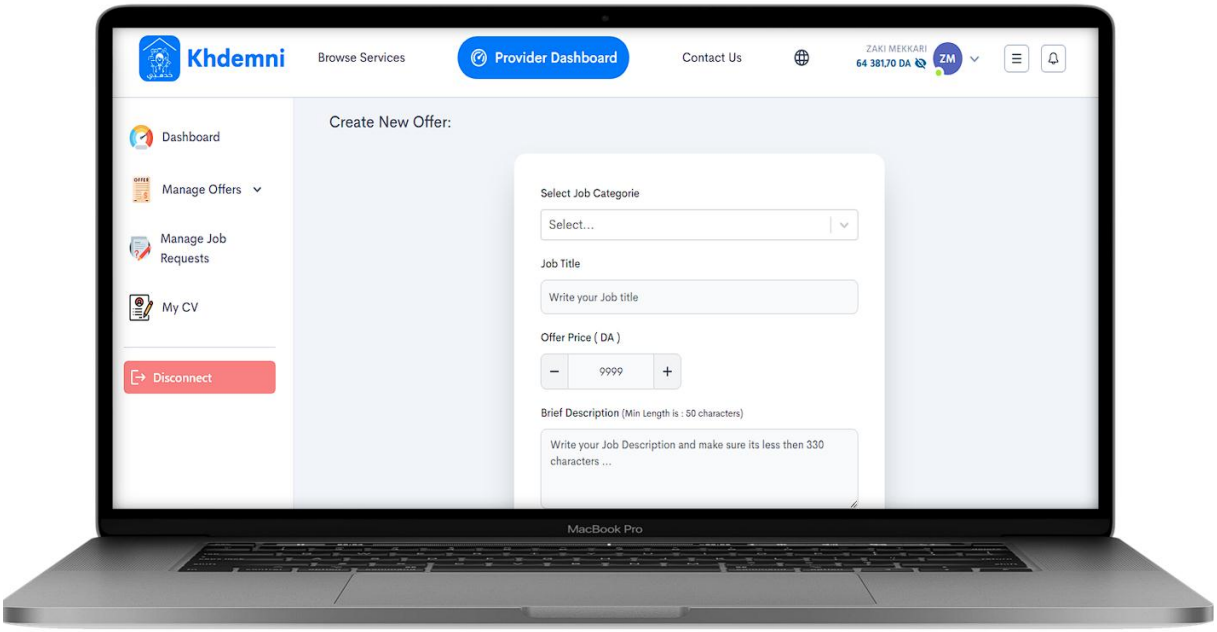


Figure 9 Create New Offer In Platform

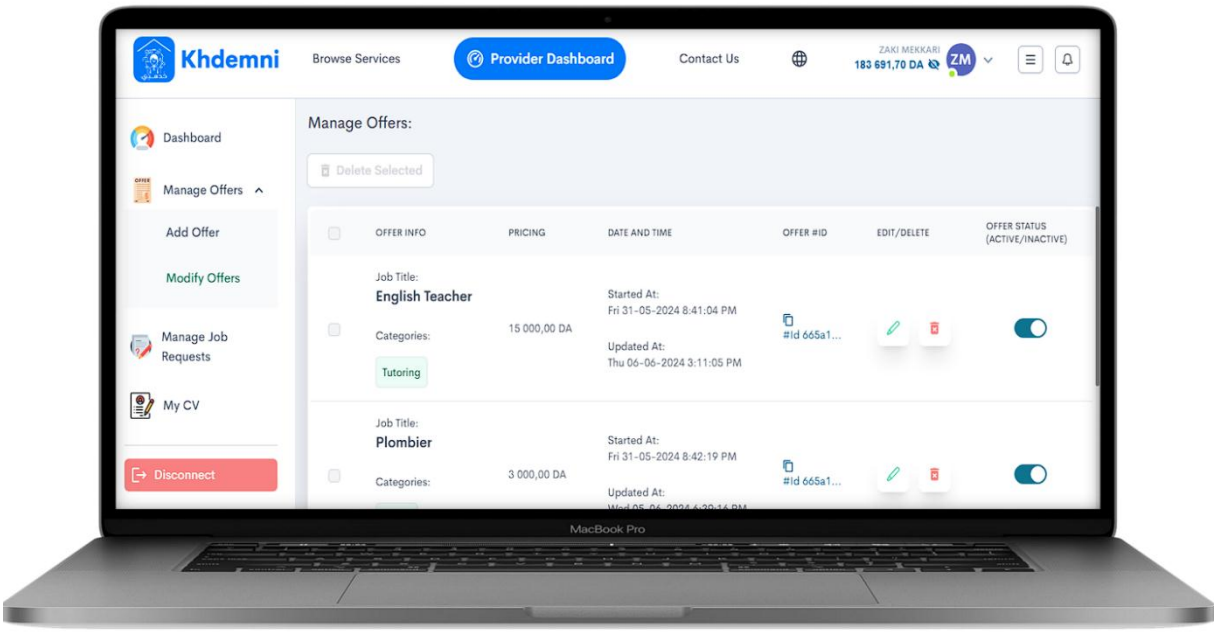


Figure10 Modify Offers In Platform



Figure 14 Map In Mobile

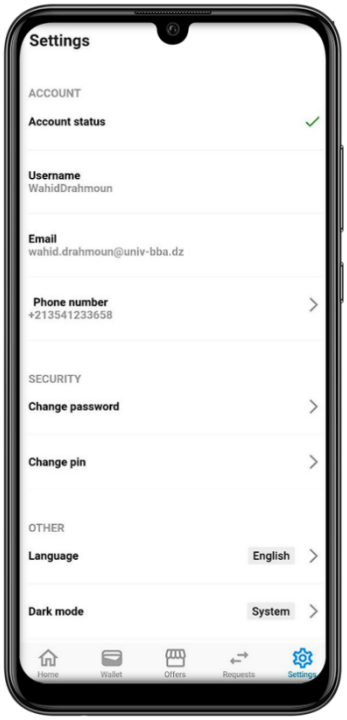


Figure 13 Settings In Mobile

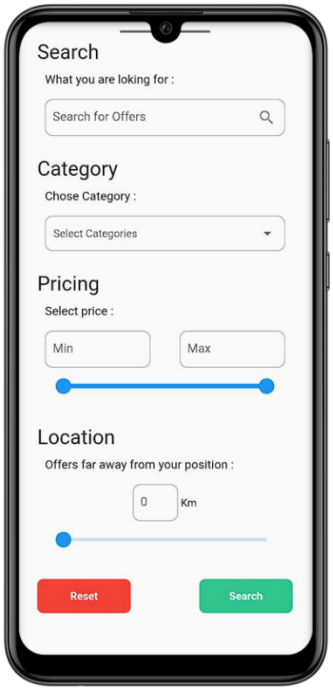


Figure 12 Search In Mobile

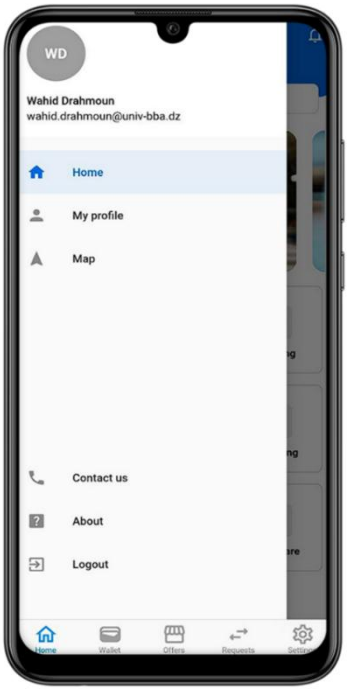


Figure 11 Drawer In Mobile

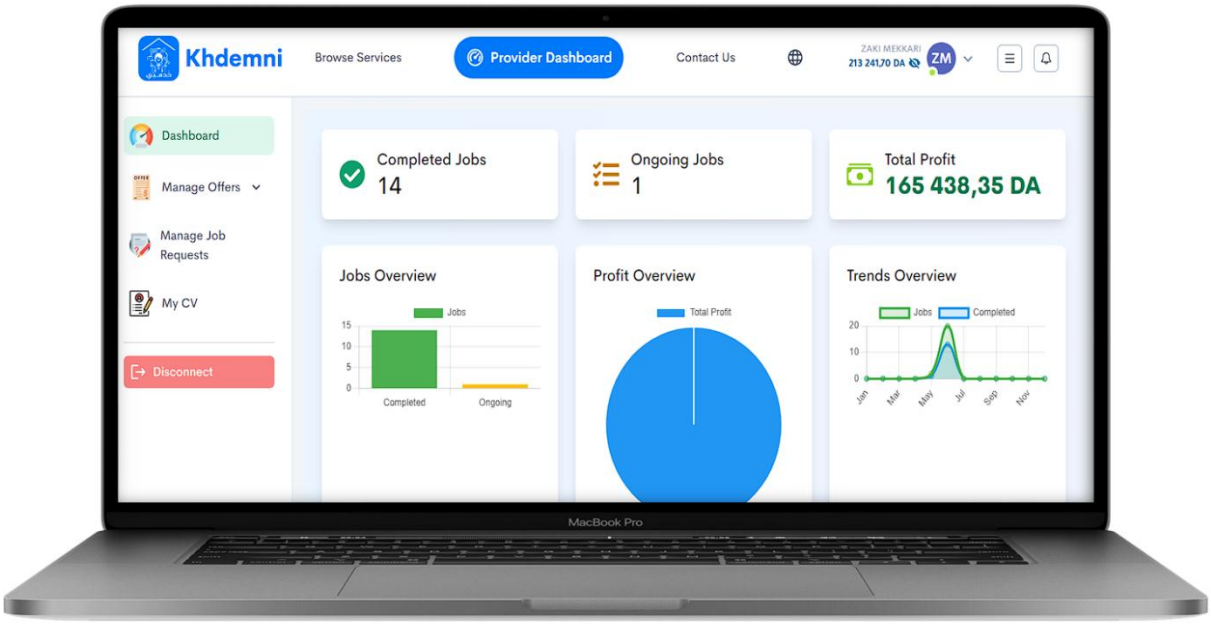


Figure 15 Dashboard In web application

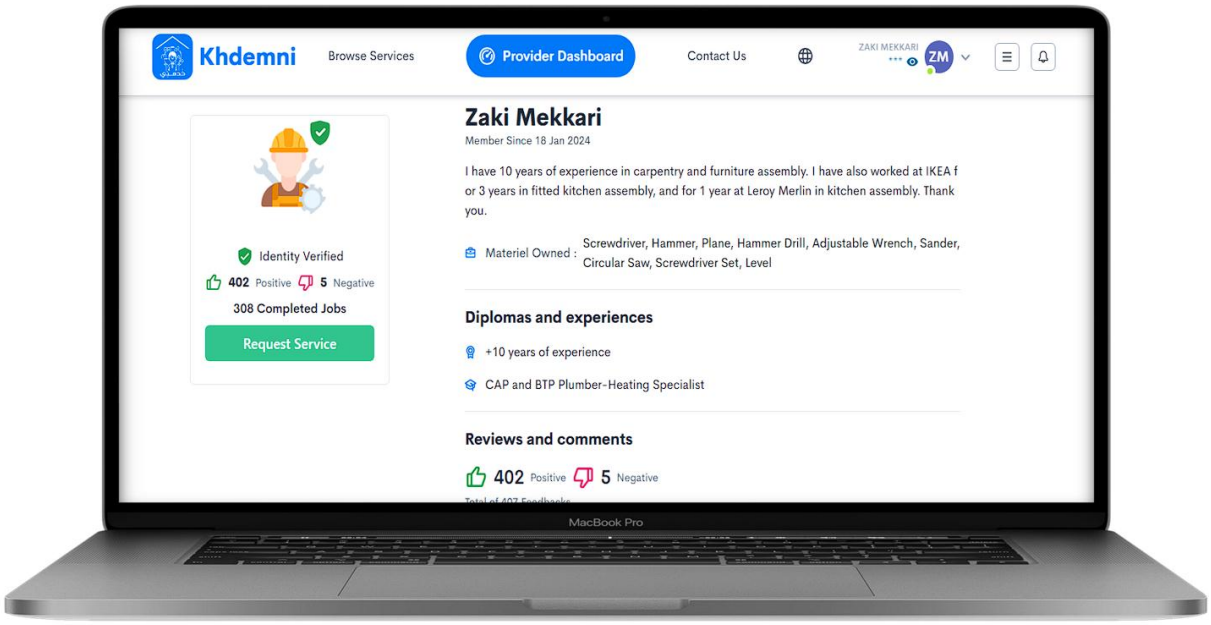


Figure 16 Service provider In application

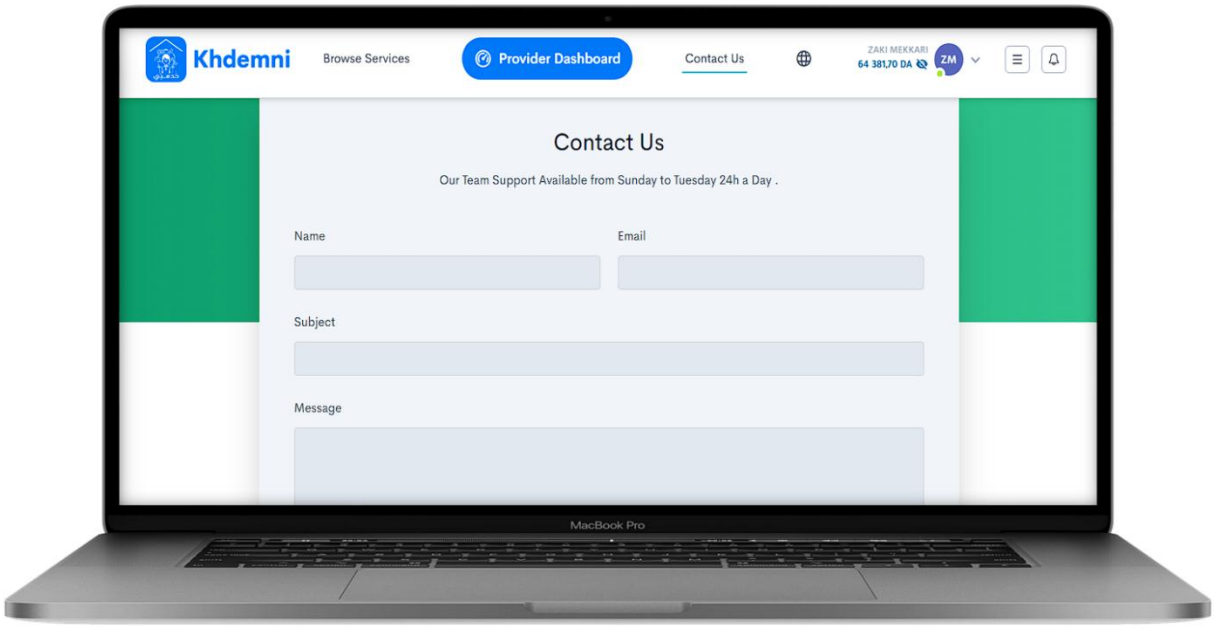


Figure 17 Contact Us In Web Application

الجداول والأشكال

الجدول(01): التكاليف الأولية

	التكلفة التقديرية (د.ج)
تكلفة وتطوير المنصة	600 000.00
تصميم واجهة المستخدم	150 000.00
البنية التحتية للخوادم	200 000.00
تراخيص البرمجيات	50 000.00
رواتب الفريق التقني والإدارة	3 200 000.00
خدمة العملاء والدعم الفني	720 000.00
التكاليف الكلية للحملات الإعلانية والتسويقية	1 500 000.00
معدات التصوير الفوتوغرافي	300 000.00
المعدات المكتبية	700 000.00
الامتثال القانوني والتنظيمي	300 000.00
تكلفة استئجار المكتب الخاص بالمؤسسة	360 000.00
رسوم إضافية	200 000.00
المجموع	**8 280 000.00**

.

الجدول(02): ميزانية المؤسسة الناشئة

KHDEMNI

ETS DE PRESTATION
ADRESSE: B,B,ARRERIDJ

B I L A N P R E V I S I O N N E L

R U B R I Q U E S	B I L A N A C T I F					
	Note	2024	2025	2026	2027	2028
ACTIF IMMOBILISE (NOM COUR)						
Ecarts d'aquis,(goodwill positif)						
Immobilisation incoroorelles						
IMMOBILISATION XORPORELLES						
Terrains						
Batiments industriels						
Autres immobilisation corporelles		1 100 000	1 100 000	1 100 000	2 500 000	4 000 000
Immobilisation Concession						
Immobilisation en cours						
Amortissements		220 000	440 000	660 000	1 160 000	1 960 000
IMMOBILISATION FINANCIERES						
Titres mis en équvalence						

Autres particip,et créances ratt,						
Autres titres immobilisés						
Prets et Autres actifs fin,non cour,		0	0	0	0	0
Impot Differés d'actif						
TOTAL ACTIF NOM COURANT		**880 000**	**660 000**	**440 000**	**1 340 000**	**2 040 000**
ACTIF COURANT						
Stocks et encours		0	0	0	0	0
CREANCE ET EMPLOIS ASSIMILES						
Clients		200 000	320 000	450 000	700 000	2 000 000
Autres débiteurs		0	0	0	0	0
Impots et assimilés		0	0	0	0	0
Autres créances et Emploi assimilés		0	0	0	0	0
DISPONIBILITES ET ASSIMILES						
plac,et autres actifs fin,couran						
Tresorerie		1 102 000	2 600 000	5 411 000	16 653 000	20 823 000
TOTAL ACTIF COURANT		**1 302 000**	**2 920 000**	**5 861 000**	**17 353 000**	**22 823 000**
TOTAL GENERAL ACTIF		**2 182 000**	**3 580 000**	**6 301 000**	**18 693 000**	**24 863 000**

RUBRIQUES	BILAN PASSIF					
	Note	2024	2025	2026	2027	2028
CAPITAUX PROPRES						
Capital émis		1 500 000	1 500 000	1 500 000	9 000 000	9 000 000
Capital non appelé						
Primes et réserves		0	0	0	0	0
Ecarts de réévaluation						
Ecarts d'équvalence						
Resultat net		522 000	1 318 000	2 571 000	4 622 000	5 900 000
Autres cap,propres/Rep,à nouv,		0	522 000	1 840 000	4 411 000	9 033 000
TOTAL CAPITAUX PROPRES (I)		2 022 000	3 340 000	5 911 000	18 033 000	23 933 000
PASSIF NON-COURANTS						
Emprunt et dettes financieres		0	0	0	0	0
Impots Différes et provisionnés						
Autres dettes non courantes						
Prov,et prod,constatés d'avance						
TOTAL PASSIFS NON-COURANTS (II)		0	0	0	0	0
PASSIF COURANTS						
Fournisseurs et compte rattachés		150 000	230 000	380 000	650 000	920 000
Impots		10 000	10 000	10 000	10 000	10 000
Autres dettes		0	0	0	0	0
Tresorerie passive						

TOTAL PASSIFS COURANTS	**160 000**	**240 000**	**390 000**	**660 000**	**930 000**
TOTAL GENERAL PASSIFCTIF	**2 182 000**	**3 580 000**	**6 301 000**	**18 693 000**	**24 863 000**

الجدو ل(03): جدول حسابات النتائج المتوقعة

R U B R I Q U E S	**TABLEAU DES COMPTES DE RESULTAT**					
	Note	**2024**	**2025**	**2026**	**2027**	**2028**
Vente de Marchandises						
Production Vendue,Vente de travaux						
Prestation de servises		2 500 000	3 750 000	7 500 000	12 500 000	20 000 000
Production immobilsée						
Subventions d'exploitation						
I - Production de l'exercice		**2 500 000**	**3 750 000**	**7 500 000**	**12 500 000**	**20 000 000**
-						
ACHATS DE MARCHANDISES VENDUES						
MATIERES PREMIERES						
AUTRES APPROVISIONNEMENTS		300 000	500 000	800 000	1 100 000	1 400 000
AUTRES CONSOMATIONS						
SOUS TRAITANCE		0	0	0	0	0
LOCATION		360 000	360 000	360 000	400 000	500 000
ENTRETIEN ET REPARATION						
PRIME D'ASSURANCE		10 000	20 000	30 000	50 000	120 000
RENUMERATION D'INERMIDIAIRES ET HONORAIRES						
PUBLICITES		8 000	23 000	45 000	78 000	140 000
DEPLACEMENT MISSION ET RECEPTION		0	0	200 000	400 000	500 000
AUTRES SERVICES		10 000	27 000	84 000	90 000	130 000
II- Consommation de, l'exercice		**688 000**	**930 000**	**1 519 000**	**2 118 000**	**2 790 000**
III VALEUR AJOUTEE D'EXPLOITAION (I-II)		**1 812 000**	**2 820 000**	**5 981 000**	**10 382 000**	**17 210 000**

Charges Personnel		960 000	1 152 000	2 880 000	4 800 000	9 600 000
Impots,taxes et vers,assimilés		0	0	0	0	0
IV EXCEDENT BRUT DEXPLOITATION		**852 000**	**1 668 000**	**3 101 000**	**5 582 000**	**7 610 000**
Autres Produits operationnels		0	0	0	0	0
Autres Charges opérationnelles		100 000	100 000	250 000	300 000	500 000
Dotation aus Amort,et aux prov,		220 000	220 000	220 000	500 000	800 000
Reprise sur pertes de val,et ptov,						
V RESULTAT OPERATIONEL		**532 000**	**1 348 000**	**2 631 000**	**4 782 000**	**6 310 000**
Produits Financiers			0	0	0	0
Charges Financiéres		0	20 000	50 000	150 000	400 000
VI RESULTAT FINANCIER		**0**	**-20 000**	**-50 000**	**-150 000**	**-400 000**
VII RESULTAT AVANT IMPOTS (V + VI)		**532 000**	**1 328 000**	**2 581 000**	**4 632 000**	**5 910 000**
Elements ext,(Produits à preciser)						
Elements ext,(Charges à preciser)						
VIII-RESULTAT EXTRORDINAIRE						
Impots exigibles sur resultats ord,		10 000	10 000	10 000	10 000	10 000
Impots defféres (Var,) sur rés,ord,						
XI RESULTAT NET		**522 000**	**1 318 000**	**2 571 000**	**4 622 000**	**5 900 000**

الجدول(04): الخزينة الافتتاحية

KHDEMNI

ADRESSE:WE DE B,B,ARRERIDJ

EXERCICE 2024

DATE	PIECE	LIBELLE OPERATION	REF	DEBIT	CREDIT
1/1/2024	0	**PC BUREAU X3**	OUVERT	450,000.00	
1/1/2024	0	**IMPREMANTE COULEUR CANON X3**	OUVERT	150,000.00	
1/1/2024	0	**BUREAU X3**	OUVERT	120,000.00	
1/1/2024	0	**ARMOIRE**	OUVERT	80,000.00	
1/1/2024	0	**SERVEUR**	OUVERT	300,000.00	
1/1/2024	0	**CAISSE**	OUVERT	400,000.00	
1/1/2024	0	**CAPITAL**	OUVERT		1,500,000.00
TOTAL				**1,500,000.00**	**1,500,000.00**

مخطط نموذج العمل التجاري لمنصة خدمني

الشريحة المستهدفة

- الأفراد الباحثين عن خدمات متنوعة (التعليم الخاص ، الرعاية الصحية ، التنظيف النقل الحرفية)
- الأشخاص البسطاء الباحثين عن الاستعانة بمصادر خارجية في العمل (أصحاب المهارات ، الحرفيين)
- المقاولون الذاتيون (الباحثون عن فرص عمل جديدة ، الراغبون في إدارة مشاريعهم بطريقة مبسطة)

العلاقات مع العملاء

- علاقة شخصية مباشرة مع العملاء
- منصة الكترونية
- مواقع التواصل الاجتماعي
- الخدمات الآلية
- نظام مراجعات وتقييمات

قنوات التواصل

- ·الإعلان عبر وسائل التواصل الاجتماعي
- ·استراتيجيات تحسين المحتوى
- ·برنامج إحالة
- الوسائط

القيمة المضافة

توفر المنصة بيئة متكاملة تسهل للأفراد العثور على المقاولين المستقلين وأصحاب الأعمال الحرة وتوظيفهم بسهولة، مع الامتثال للقانون الرئاسي رقم 23-22 المتعلق بالقانون الأساسي للمقاول الذاتي.

تتميز المنصة بما يلي :

- خاصيات متطورة لضمان الأموال وحل النزاعات
- توفير أدوات تواصل مباشرة بين العملاء والعاملين.
- نظام تتبع الوقت لضمان العمل الفعال.
- إدارة الفواتير لضمان شفافية العمليات المالية.
- بيئة آمنة للمستخدمين تضمن حماية البيانات والمعاملات.
- تجربة مستخدم سلسة تدعم نجاح الأعمال وتعزيز الثقة.

الأنشطة الرئيسية

- تطوير وتصميم وصيانة المنصة
- تصميم واجهة مستخدم بسيطة وسهلة التنقل.
- تحسين تجربة المستخدم لتكون سلسة وفعالة.
- دعم فني متواصل للمساعدة في حل المشاكل
- الاستثمار في التسويق والإعلان
- تعزيز صورة المنصة كمكان موثوق وآمن للتعاقد مع المقاولين الذاتيين وأصحاب الأعمال الحرة.

الموارد الرئيسية

1. **موارد بشرية :**
   - فريق المطورين والمصممين ، موظفو خدمة العملاء
2. **موارد مادية**
   - الخوادم والأجهزة المعلوماتية
   - موارد التسويق والإعلان

الشركاء الرئيسيون

بوابات الدفع الإلكتروني

- شراكة مع بوابات الدفع الإلكتروني لضمان مدفوعات آمنة وفعالة.

شركات التأمين

- شراكة مع شركات التأمين لتأمين وضمان حماية بيانات المنصة.

صندوق تمويل المؤسسات

- شراكة مع صندوق تمويل المؤسسات لتقديم دعم مالي.

مصادر الإيرادات

- تحقيق الإيرادات من خلال نموذج قائم على العمولة (تتراوح العمولة عادةً بين 5% و20% من قيمة الخدمة)
- مصادر إيرادات إضافية (توفير مساحات إعلانية على المنصة ، تقديم خيارات إعلانية للأفراد والشركات لزيادة ظهور خدماتهم)
- إتاحة فرصة للعاملين لدفع رسوم إضافية لجعل خدماتهم تظهر في مقدمة نتائج البحث.

هيكل التكاليف

- تكاليف تطوير البرمجيات وصيانتها
- تكاليف شراء الأجهزة وتجديدها
- الالتزامات الضريبية والتأمين
- التسويق والإعلان

Abstract

In this dissertation, we focus on designing and implementing a platform that offers a unique experience for service providers, freelance contractors, and customers.

Our platform, named "Khedemni," has been created to streamline service operations between providers and clients. Customers can search for the right professional for their required services, while service providers can showcase their offerings and interact directly with clients.

Keywords : Mobile applications , Service , UML , Money transfer , Startup .

Dans ce Mémoire, nous concentrons sur la conception et la mise en œuvre d'une plateforme offrant une expérience unique aux prestataires de services, aux travailleurs indépendants et aux clients.

Notre plateforme, nommée "Khedemni", a été créée pour rationaliser les opérations de service entre les prestataires et les clients. Les clients peuvent rechercher le professionnel approprié pour leurs services requis, tandis que les prestataires de services peuvent présenter leurs offres et interagir directement avec les clients.

Mots clé : Applications mobiles , Service , UML , Transfert d'argent , Startup .

نَعْمَل فِي هَذِه الْمُذَكِّرَة عَلَى تَصْمِيم وَتَنْفِيذ مَنْصَّة تُقَدِّم تَجْرِبَة فَرِيدَة لأَصْحَاب الْخَدَمَات وَالْمُقَاوِلِين الذَّاتِيِّين وَكَذَلِك الزَّبَائِن.

تَم إِنْشَاء مَنْصَّتِنَا الْمُسَمَّاة " خَدِّمْنِي " لِتَسْهِيل الْعَمَلِيَّات الْخِدْمِيَّة بَيْن أَصْحَاب الْخَدَمَات وَالْعُمَلَاء، حَيْث يُمْكِن لِلزَّبَائِن الْبَحْث عَن الْعَامِل الْمُنَاسِب لِخَدَمَاتِهِم الْمَطْلُوبَة، وَبِالنِّسْبَة لِلْعَامِلِين، يُمْكِنُهُم عَرْض خَدَمَاتِهِم وَالتَّفَاعُل الْمُبَاشِر مَع الْعُمَلَاء.

الكلمات المفتاحية : تطبيقات الهواتف المحمولة، خدمة، لغة النمذجة الموحدة، تحويل الأموال، شركة ناشئة.