People’s Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M’Hamed BOUGARA – Boumerdès

Institute of Electrical and Electronic Engineering

Department of Electronics

Project Report Presented in Partial Fulfilment of

the Requirements of the Degree of

‘MASTER’

In Electrical and Electronic Engineering

Title:

Freelancing Platform for Tradesmen Profession “TRADEHUB”

Presented By:

- **BEROUAKEN Mohamed**

Supervisor:

Dr.ZITOUNI Abdelakder

Dedication

To begin, I express my gratitude to the Almighty God for giving me the strength and knowledge to live my life every day, and for providing my family and I with good health and protection.

I would like to dedicate this work to my mother, who serves as my role model and support system, as her selfless care inspires me to persevere. Additionally, I extend my appreciation to my father, who consistently encourages me to succeed.

I also want to acknowledge my siblings Imad and manel who always believed in my abilities and demonstrated eternal love. Furthermore, I owe a debt of gratitude to my closest friends Sami, Yahia and Nacer who made my university experience more enjoyable by growing alongside me.

Lastly, I would like to extend my heartfelt appreciation to all my family members, friends, and those I have met along the way, regardless of their proximity. Thank you all!

Acknowledgements

First and foremost, I express my gratitude to ALLAH for enabling me to complete this modest work. My unwavering faith in him has enabled me to persevere during difficult times when it seemed impossible to continue.

I would like to extend my sincere appreciation to my supervisor, Dr. Zitouni Abdelkader for his constant guidance and encouragement. Without his unwavering support, this work would not have been possible, and I am truly grateful to him.

I would also like to express my deepest appreciation to Dr. OTMANI for his valuable lectures and feedback, particularly his guidance on the Advanced Programming course.

I am also grateful to all the teachers at the Institute of Electrical and Electronic Engineering, particularly those in the Computer Engineering option, who provided me with a solid foundation to develop my project.

Abstract

This project aims to develop a freelancing platform specifically designed for tradesmen (Electricians, Plumber, Handyman...), providing them with a user-friendly and efficient way to connect with potential clients and offer their services. The platform “Tradehub” will mainly have three pages: the home page in which clients can see some services that other freelancers posted them, the account page for both the client and the freelancer in which they can access the jobs they posted and finally a search page so that clients can search for specific service.

This platform is built using HTML, CSS and JavaScript for the Front-End and PHP for Back-End, with a focus on providing a seamless user experience. Its core features include user registration, job posting and bidding, and dispute resolution, all tailored to the needs of tradesmen and their clients. This work delves into the design and implementation of the platform, shedding light on the challenges faced during the development process and the corresponding solutions employed.

Overall, this project demonstrates the feasibility of creating an intuitive freelancing platform for tradesmen using modern web technologies. With this platform, tradesmen can effectively expand their customer base and increase their revenue while providing their clients with a convenient and reliable way to access their services. The resulting platform serves as a testament to the successful realization of the project's objective

Table of Contents

List of Figures:

List of Tables:

General Introduction:

I Introduction:

The internet has changed the way we conduct business and access services. With the rise of gig economy, freelancing and job-listing platforms became a very popular way for individuals to hire professionals to complete their tasks. However, most of these platforms are geared towards graphic designers and programmers; very few of them are for tradesmen professions specifically.

Tradesmen, such as plumbers, electrician and painters provide essential services that helps in maintaining our homes and businesses. But finding a reliable tradesmen can be a daunting task especially in Algeria where there is no centralized platform connecting clients and tradesmen. This resulted in many consumers being unable to find a skilled tradesman and resulting to hiring unlicensed or unskilled workers.

The motive behind this project is to provide a reliable platform that connects between the consumers and skilled tradesmen in Algeria. Currently many consumers in Algeria resort to word-of-mouth recommendation and directory listings, which can be tiring and time consuming. By building a platform that connects consumers with qualified tradesmen, this project aims to provide a convenient way to finding and hiring tradesmen.

In addition, this platform can benefit tradesmen also by providing them with a platform to showcase their work and attract more customers. Many skilled tradesmen in Algeria struggle to find work due to the lack of exposure, and this platform may help them build reputation and credibility leading to more job offers and increase in earnings.

In conclusion, this project aims to build a web-based platform to help connect consumers with skilled tradesmen to facilitate the task of finding and hiring tradesmen in Algeria. By filling this gap in the market, the platform will help to create more job opportunities for skilled tradesmen while also providing a valuable service to consumers.

II Problem Statement:

The research problem addressed in this study is the lack of a centralized platform for accessing the services of tradesmen professionals in Algeria. The absence of such a platform makes it difficult for individuals to find reliable and trustworthy tradesmen for various services.

The lack of transparency in pricing and service quality also hinders the growth of the tradesmen profession in Algeria. Additionally, the traditional methods of finding tradesmen, such as word of mouth or newspaper ads, are outdated and unreliable. Therefore, the research problem is to develop a platform that provides a centralized and reliable source for accessing tradesmen services, thereby addressing the issues of transparency, trustworthiness, and accessibility for both the customers and the tradesmen professionals.

III Objectives:

Building a centralized web-based platform that connects consumers with tradesmen and provide a reliable and efficient way to find and hire tradesmen in Algeria.

IV Overview:

There are five chapters in this report. After defining the project objectives and research problem, the first chapter will provide information about the theoretical background needed to complete this project from the concept of web and the programming languages used in this project. The second chapter is about the design of our website and how it was implemented using (HTML, CSS and JAVA SCRIPT). The third chapter will discuss in detail how the front and back-end of our platform were implemented. The fourth chapter will present the results we got with brief over view on how the platform looks online as well as discuss our results and limitations.

Chapter One: Theoretical Background

Chapter One: Theoretical Background

1.1 Introduction:

This chapter aims to define the project objectives and problem statment and then provide information about the theoretical background needed to complete this project.

1.2 Web applications:

1.2.1 Definition:

A web application is a software program accessible through a web browser that allows the user to interact and perform tasks over the internet. Unlike static websites, web application are dynamic, providing interactive functionality, data processing, and often involve user authentication and data base integration. Web applications commonly utilize a combination of front-end technologies (HTML, CSS, JavaScript) for user interface design and interactivity, along with back-end technologies (programming languages, frameworks, and databases) for server-side processing and data management [1]. Web applications can be further classified into two main types: single-page web applications and multi-page web applications which we will discuss next.

1.2.2 Single-Page Application:

Single-page web applications (SPAs) are characterized by having a single HTML page that dynamically updates its content as users interact with the application, providing a seamless and responsive user experience. SPAs often rely on JavaScript frameworks like AngularJS, React, or Vue.js to handle client-side rendering and data management.

Overall, single-page web applications are known for their ability to deliver a fluid user experience within a single page, without the need for frequent reloads [2].

1.2.3 Multi-Page Application:

Multi-page web applications consist of multiple interconnected HTML pages. Each page represents a separate functional component or section of the application. Users navigate between different pages to access different features or perform specific tasks. Unlike single-page web applications (SPAs), multi-page applications rely on traditional server-side rendering, where each interaction typically results in a page refresh. This approach is suitable for applications with distinct sections or complex navigational structures, providing clear separation between different functionalities and content. Multi-page web applications are built using server-side technologies, such as PHP, Python, or Ruby, to handle requests, process data, and render HTML pages dynamically [3].

1.2.4 Web client and Web server:

1.2.4.1 Web client:

A web client is a software application, typically a web browser, that users use to access websites and web applications. It serves as a gateway between users and the web, providing a user-friendly interface to navigate and interact with web content. The web client plays a crucial role in rendering web pages by interpreting the HTML, CSS, and executing JavaScript code. It ensures that the visual and interactive aspects of web pages are displayed accurately and enables users to interact with various elements, such as forms, buttons, and media. [4].

1.2.4.2 Web server:

A web server can be either a hardware or software or both component working together to facilitate the hosting, processing, and delivery of web content.

On the Hardware side, a web server refers to the computer system that is used to store and serve the web server software and the files that comprise a website (HTML documents, CSS stylesheets, and JavaScript files). It connects to the internet, enabling physical data interchange with other devices.

On Software side, a web server includes components such as an HTTP server, which is responsible for understanding URLs and HTTP protocol. The HTTP server software allows users to access websites via domain names and ensures the delivery of websites content [5].

1.3 HTTP:

HTTP (Hypertext Transfer Protocol) is an application protocol that serves as the foundation for the communication between web clients and web servers. It defines the set of rules and conventions for the format and transmission of messages between the clients and server, ensuring reliable and standardized data exchange. The HTTP protocol enables clients to request specific resources, such as web pages or files, from servers and receive responses containing the requested content. By establishing a standardized protocol, HTTP simplifies the process of accessing and delivering web content, contributing to the interoperability and scalability of the World Wide Web[6].

1.4 URL:

URL (Uniform Resource Locator) is a web address that identifies the location of a specific resource on the internet. It consists of several components, each serving a specific purpose in defining the location and accessing the resource [7]. The most important parts are highlighted on the URL parts on Figure 1.1:

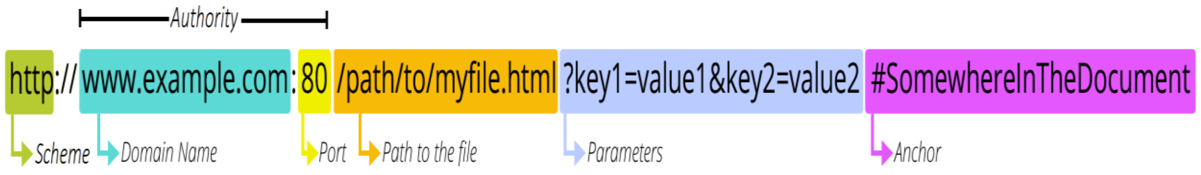

Authority
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

Figure 1.1: URL Parts [7]

1.4.1 Scheme/Protocol:

The scheme or protocol specifies the communication protocol used to access the resource. Common schemes include "http://" for Hypertext Transfer Protocol and "https://" for Hypertext Transfer Protocol Secure [7].

1.4.2 Subdomain and Domain:

The subdomain and domain identify the specific website or server hosting the resource. The subdomain is an optional part that precedes the domain. For example, in "www.example.com," "www" is the subdomain and "example.com" is the domain [8].

1.4.3 Top-Level Domain:

The top-level domain is the last part of the domain and represents the category or type of organization associated with the website. Examples include ".com" for commercial websites, ".org" for non-profit organizations, and ".edu" for educational institutions [8].

1.4.4 Path:

The path represents the specific location or file within the website's directory structure. It helps navigate to a specific resource or page on the website [7].

1.4.5 Query Parameters:

Query parameters are optional and appear after a question mark "?" in the URL. They provide additional information to the server, such as search terms or parameters for data retrieval. Query parameters are typically in the form of key-value pairs, separated by ampersands "&" [7].

1.4.6 Fragment Identifier/ Anchor:

The fragment identifier is an optional part of the URL that appears after a hash "#" symbol. It specifies a specific section or element within the resource itself, such as an anchor tag on a webpage [7].

1.5 Programming Languages:

Programming languages are formal languages designed for instructing computers to perform specific tasks. They provide a set of syntax and semantics that enable developers to write code and create software applications. Programming languages are used to define the behavior and functionality of a program, allowing developers to build web applications and other software systems [9].

1.5.1 HTML:

HTML is a markup language used for structuring the content of web pages. It provides a standardized way to describe the elements and layout of a webpage, including text, images, links, and other media. HTML uses tags to mark up different elements and create a hierarchical structure. Web browsers interpret HTML code to render web pages on users' devices [10].

1.5.2 CSS:

CSS is a style sheet language used to describe the presentation and visual layout of HTML documents. It enables developers to define the appearance of web pages, including colors, fonts, spacing, and positioning of elements. CSS works in conjunction with HTML to separate the content and presentation aspects of a webpage, allowing for consistent and reusable styling across multiple pages [11].

1.5.3 JavaScript:

JavaScript is a versatile programming language primarily used for creating interactive and dynamic web content. It runs on the client-side (in the browser) and enables developers to add behavior, interactivity, and real-time updates to web pages. JavaScript can manipulate HTML elements, handle user events, make asynchronous requests, and interact with server-side components. It plays a crucial role in enhancing the user experience and adding functionality to web applications [12].

1.5.4 Python:

Python is a high-level programming language known for its readability and simplicity. It is widely used for web development, among other domains. In the Tradehub project, Python was used in conjunction with the Django framework. Django is a powerful web framework that follows the Model-View-Controller (MVC) architectural pattern, allowing developers to build scalable and maintainable web applications. Python, along with Django, facilitated server-side processing, database interactions, and the overall business logic of the Tradehub web application [13].

1.6 Tools and Frameworks:

The Tradehub project incorporates a range of tools and frameworks that enhance the development process and provide additional functionalities. These tools streamline the workflow and help create a robust and efficient web application. The following are the key tools and frameworks utilized in the Tradehub project:

1.6.1 Visual Studio Code (VScode):

VScode is a powerful source code editor that offers a wide range of features such as syntax highlighting, debugging, code completion, and version control integration. It was the primary editor used throughout the development of Tradehub due to its flexibility, extensive plugin ecosystem, and user-friendly interface [14].

1.6.2 StarUML:

StarUML is a UML modeling tool that aids in designing and visualizing the system architecture. It allows for the creation of UML diagrams such as class diagrams, sequence diagrams, and use case diagrams [15]. StarUML was utilized to create comprehensive UML diagrams for Tradehub, providing a visual representation of the system's structure and interactions.

1.6.3 Django:

Django is a high-level Python web framework that simplifies the development of complex web applications. It follows the Model-View-Controller (MVC) architectural pattern, providing a clear separation between the data models, views, and templates. Django was chosen for its robustness, scalability, and built-in features such as authentication, database management, and URL routing [16].

1.6.4 SendGrid:

SendGrid is an email delivery service that offers reliable email sending capabilities. It provides APIs and libraries for integrating email functionality into web applications. In Tradehub, SendGrid was used to facilitate customer service email sending, ensuring that important notifications and communication were effectively delivered to users [17].

1.7 Introduction to The Database:

1.7.1 Definition:

A database is a structured collection of data that allows for efficient storage, organization, and retrieval of information. It serves as a central repository for data used in applications, enabling operations such as inserting, updating, querying, and deleting data. Databases can be relational or non-relational, offering different approaches to data storage and management [20].

1.7.2 Relational Model:

The relational model is a widely used data model that represents the database as a collection of tables with predefined relationships between them. In Tradehub, the relational model is employed to organize and structure data in a logical manner. Tables such as "Users," "Jobs," and "Comments" are created with appropriate fields to store related information. The relational model ensures data consistency, integrity, and allows for efficient querying through SQL operations [21].

1.7.3 Non-Relational Model:

The non-relational model, also known as the NoSQL model, is an alternative to the relational model for database management. NoSQL databases, such as MongoDB or other document-oriented databases, offer flexibility in data storage and retrieval [21]. In Tradehub, certain components of the system, such as user comments or customer inquiries, may be stored in a non-relational database. This model allows for scalability, fast data retrieval, and accommodates unstructured or semi-structured data.

1.7.4 SQLite:

SQLite is a popular embedded database engine that provides a self-contained, serverless, and file-based relational database management system (RDBMS). It is lightweight, fast, and widely used in various applications and programming languages. SQLite implements a transactional SQL database engine with support for standard SQL syntax, data integrity, and ACID (Atomicity, Consistency, Isolation, Durability) properties. It is designed to be embedded within applications, requiring minimal configuration and administration. SQLite is known for its simplicity, efficiency, and compatibility, making it a popular choice for small to medium-sized projects [22].

1.8 Modeling Language:

Modeling languages are essential tools used in software development to represent and communicate different aspects of a system's design. They provide a standardized way to visually depict the structure, behavior, and interactions of software components. In the context of the Tradehub project, we utilized modeling languages to create clear and concise representations of the system's architecture and design [18].

1.8.1 Unified Modeling Language (UML):

The Unified Modeling Language (UML) is a widely adopted modeling language in software engineering. It provides a standardized notation to visualize, specify, construct, and document the artifacts of a system's design. UML offers a comprehensive set of diagrams that cover different aspects of a software system, allowing developers to effectively communicate and understand the system's structure, behavior, and interactions [19].

1.8.2 Use Case Model:

A Use Case Model provides a high-level view of the system's functionality from the perspective of its users, known as actors. It helps identify the various interactions between actors and the system, represented as use cases. Each use case represents a specific goal or task that an actor can perform within the system.

1.9 Conclusion

In this chapter, we have explored the theoretical foundations that form the basis of our project. We discussed web applications and their types, highlighting Tradehub as an example, and examined important concepts such as HTTP, URLs, web servers, and clients. We also defined the various programming languages, frameworks and tools that were used to design and build our website.

By understanding these concepts and employing the necessary tools and frameworks, we have gained insights into the fundamental principles behind web development. In the next chapter, we will delve deeper into the architectural design and implementation details of the Tradehub web application, further enhancing our understanding of its technical aspects.

Chapter Two: Design

Chapter Two: Design

2.1 Introduction:

In this chapter, we explore the design of Tradehub, our web application connecting tradesmen and clients. Using UML, we present a comprehensive model illustrating system components and interactions. Additionally, we discuss the database structure, highlighting the use of Django ORM. These design elements form the foundation of Tradehub, enhancing job searches, communication, and user experience for tradesmen and clients.

2.2 Modelling:

In the Tradehub project, we employed UML to create various diagrams that capture different perspectives of the application's design. These diagrams helped us analyze and communicate the system's architecture, relationships between components, and the flow of information and control.

By utilizing UML, we were able to visualize and document the Tradehub application's design in a clear and concise manner. In the following sections, we will delve into the specific UML diagrams used in the Tradehub project, highlighting their purpose and how they contribute to the overall design of the application..

2.2.1 Actors:

Actors in the Tradehub system represent different entities that interact with the application. Based on our project, we can identify the following actors:

1. Tradesmen: Represents registered users who post their services in the website.
2. Customer: Represent users who come to website just to view its contents
3. Admin: Represents an administrator or superuser with privileged access and additional functionalities.

In our website we have three types of actors:

2.2.1.1 Administrator:

Responsible for managing the whole website, he has the authority to manage all the users accounts and the jobs they posted on the platform.

2.2.1.2 Tradesmen:

They can create accounts and post their services on the platform.

2.2.1.3 Customer:

A user role in Tradehub who can browse job listings, post comments, and contact customer support.

2.2.3 Use Case Description:

The role of each actor in our web-application is defined in Table 1

Table 1 Roles of actors in the use case diagram

Actors	Use Case
Admin	• **Login**: access his account through an email and password • **User management**: The admin has the ability to add, remove users (Tradesmen) and post new jobs for them or modify old ones • **Authentication and Authorization**: The admin has the ability to set permissions of a certain user or group
Tradesmen	• **Login**: access his account with an email and password • **Job Posting**: Post new jobs or edit previous ones • **Job Deletion**: Tradesmen can delete the jobs they posted • **Comment Posting:** Tradesmen can post comments on the website. • **Customer Service Contact:** Tradesmen can fill out a form to contact customer service.
Customer	• **Comment Posting:** Customers can post comments on the website. • **Customer Service Contact:** Tradesmen can fill out a form to contact customer service.

2.2.4 Use Case Identifications:

a) **Login:** Allow the users to access their accounts

b) **User management:** The admin ability to add or remove any user, and to create new job postings for them or edit old ones

c) **Authentication and Authorization:** The admin ability to decide which kind of permissions a certain group or user can have

d) **Job Posting:** registered Tradesmen have the ability to post their services in the form of a job post, or edit, remove a previously posted job

e) **Job Deletion:** Tradesmen can delete the jobs they posted.

f) **Comment Posting:** Tradesmen can post comments on the website.

g) **Customer Service:** Tradesmen can fill out a form to contact customer service.

2.2.5 Use Case Diagrams:

A use case diagram is a visual representation of the interactions between users (known as actors) and a system. It provides a high-level view of the functionality and behavior of a system from the perspective of its users. Use case diagrams are widely used in software development to capture and communicate the requirements of a system.

Actors in a use case diagram represent the different entities that interact with the system, they each have certain tasks to perform within the system known as use cases

Use cases in a use case diagram represent specific interactions or tasks performed by actors within the system. They capture the functional requirements of the system and describe the desired behavior or outcomes. Use cases are connected to actors through associations, indicating which actors are involved in each use case.

Before show casing the diferent Use Case Diagrams of our system, let's briefly define the meaning of each arrow symbol used which represent the relationship between actors and use cases in the diagram in Table 2

Table 2 Description of used arrows

Symbol	Description
Association $\longrightarrow$	The association arrow in a use case diagram represents a relationship between an actor and a use case. It indicates that the actor interacts with or participates in the use case.
Include $\dashrightarrow$	The include arrow in a use case diagram represents a relationship between two use cases, indicating that one use case includes the functionality of another use case. It signifies that the base use case incorporates the behavior defined in the included use case. This relationship allows for modularization and reuse of common functionality across multiple use cases.
Extend $\dashleftarrow$	The extend arrow in a use case diagram indicates an optional or alternative behavior that can be added to a base use case. It signifies that the extending use case may be invoked under certain conditions, which are defined as extension points in the base use case. The extend relationship allows for the specification of additional functionality that is not always present in the base use case.

Now that we have covered the basics of the use case diagram, we will present the System Diagram in Figures 2.2, 2.3 and 2.4. Highlighting the key elements and their connections, which will provide a comprehensive visual representation of the Tradehub system's architecture.

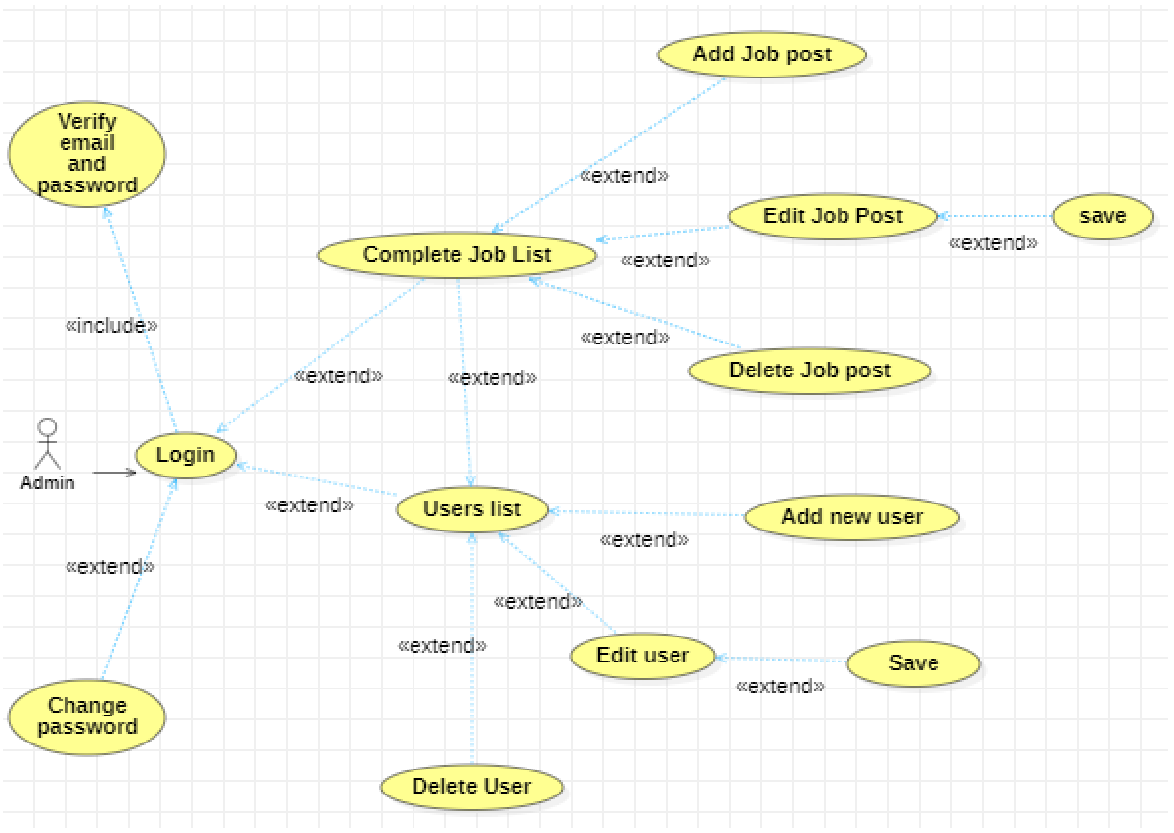


Figure 2.2 Admin Use case diagrams

From the description above (Figure 2.2), we can outline the interactions of an admin with our Tradehub platform. The admin's journey begins by logging in using their dedicated admin account. Upon successful authentication of the provided email and password, they are granted access to the admin interface, which serves as their central control hub.

Within the admin interface, the admin is presented with comprehensive options and functionalities. They can navigate through the complete list of jobs posted on the platform, gaining insights into the various opportunities available. Additionally, the admin has access to the list of registered users, enabling them to manage and monitor the user base effectively.

The admin's authority extends beyond mere observation. They possess the ability to make modifications as needed. This includes the capability to add or delete job posts or registered users, granting them a high degree of flexibility and control over the platform's content. Furthermore, the admin is empowered to edit user information or update details of previously posted jobs, ensuring the accuracy and relevance of the platform's data.

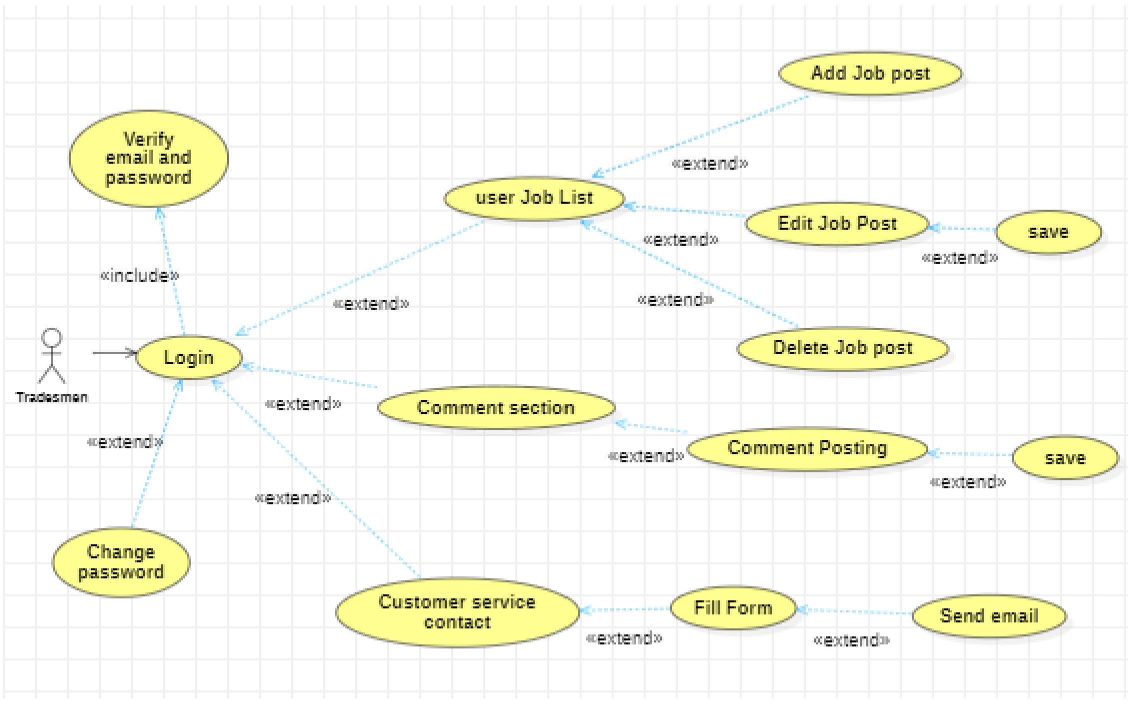


Figure 2.3 Tradesmen Use case diagram

From the description above (Figure 2.3), we can outline the interactions of tradesmen with our Tradehub platform. The process begins with tradesmen logging in and verifying their email and password. Once authenticated, they are redirected to a dedicated interface where they can access their previously posted jobs. In this interface, tradesmen have the ability to edit or delete their existing job posts and add new ones, providing them with flexibility and control over their listings. Moreover, tradesmen can actively engage with the platform by posting comments or contact customer service for any inquiries or assistance they may need during their platform usage.

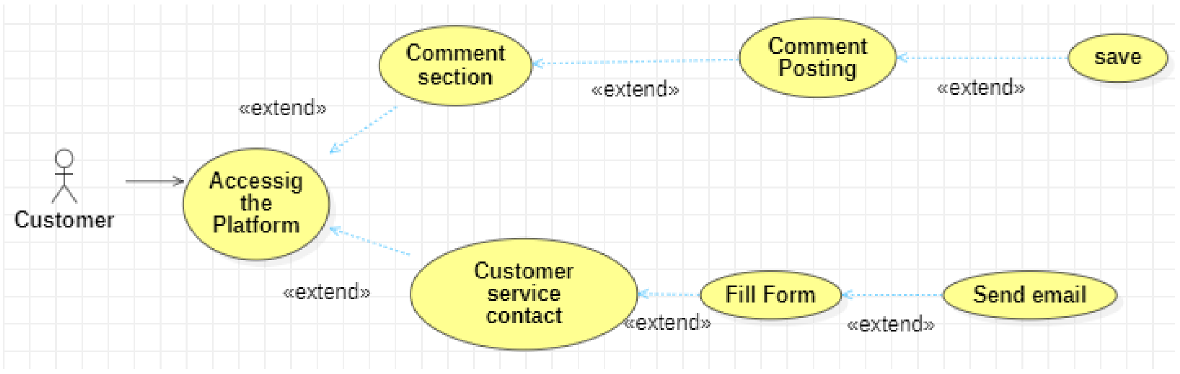


Figure 2.4 Customer Use case diagram

Figure 2.4 showcase the customer interaction with the platform. For the customers they don't have to login they can see the job list on website directly and look for their suitable tradesmen, they also have the ability to post comments and contact customer support.

2.2.6 Class Diagram:

In this section, we present the Class Diagram for our project, "Tradehub," which serves as a central hub for tradesmen and clients to connect and collaborate effectively. The Class Diagram illustrates the entities and their associations, helping us gain a deeper understanding of the system's design and organization.

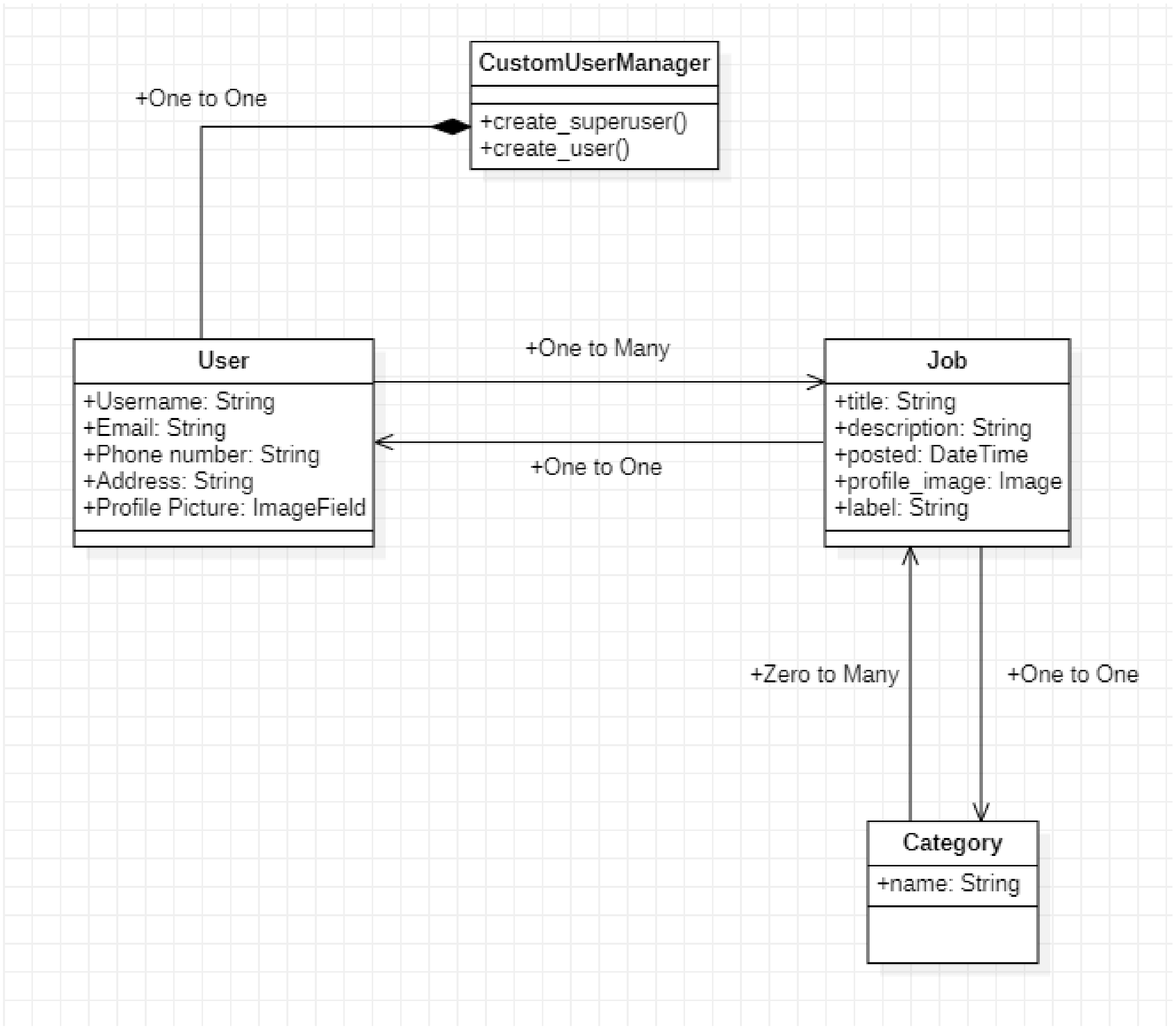


Figure 2.5 Class Diagram

The Figure 2.5 provides a comprehensive overview of the system's structure, highlighting the essential classes and their relationships within the "Tradehub" platform. The diagram showcases the following classes:

User: Represents the users of the platform, including tradesmen and clients. The User class contains attributes such as username, phone number, address, and profile picture. It has a one-to-many relationship with the Job class, allowing users to post and manage their jobs.

Job: Represents the job postings made by tradesmen. The Job class includes attributes such as title, description, posted date, and label. It has one-to-one relationships with both the Category and User classes, associating each job with a specific category and the corresponding user who posted it.

Category: Represents the categories or types of jobs available on the platform. The Category class contains attributes such as name and establishes a zero-to-many relationship with the Job class, allowing multiple jobs to be associated with a specific category.

Additionally, the diagram includes the **CustomUserManager** class, responsible for managing user creation and authentication. It is compositionally associated with the User class, ensuring each User object is associated with a **CustomUserManager** object.

The Class Diagram offers a visual representation of the system's architecture, aiding in the understanding of the relationships between entities and their roles in the platform. It serves as a valuable reference for developers, enabling them to implement the system's functionality effectively.

By analyzing the Class Diagram, we gain insights into the organization of classes, their attributes, and the associations between them. This understanding forms the foundation for the successful implementation of the "Tradehub" platform, facilitating efficient tradesmen-client connections and collaborations.

2.3 Database:

2.3.1 Importance of a Centralized Data Repository:

Our platform relies on a robust and efficient database as a central repository for storing and managing the data necessary to power the web application's functionality and facilitate user interactions. The database serves as a fundamental component, ensuring data integrity, scalability, and accessibility.

2.3.2 Efficient Data Retrieval and Storage:

Tradehub's well-structured database enables efficient data retrieval, storage, and manipulation. Users can quickly search for specific jobs, access their profiles, and retrieve relevant details, resulting in reduced response times and an enhanced user experience. The database's storage capabilities efficiently handle large amounts of data, ensuring scalability and robustness

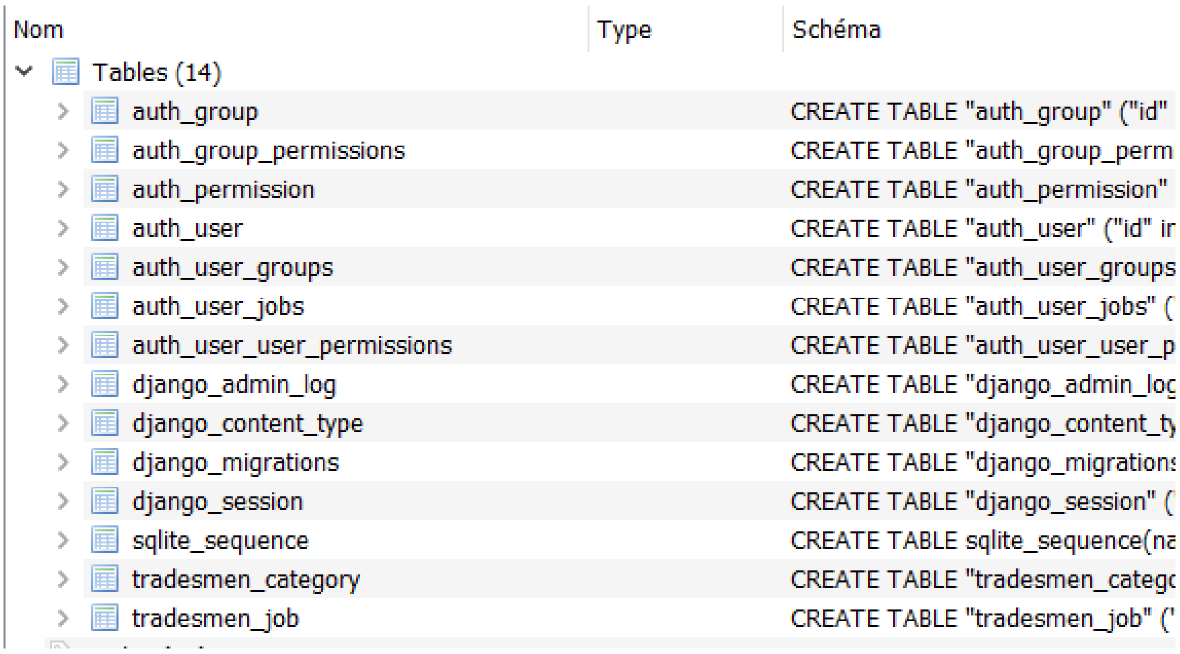
2.3.3 Enhancing User Experience:

By leveraging the power of a well-designed database, Tradehub ensures a smooth and engaging user experience. Efficient data retrieval, storage, and manipulation contribute to seamless interactions, allowing users to find information quickly and perform actions smoothly.

2.3.4 Description of the Database used in our project:

Our web application utilizes a SQLite database, which is a lightweight, serverless, and file-based database management system. The database schema consists of several tables that store different types of data related to the system's functionality.

Since we used Django commands to create the Database, there are many tables that were created by default to facilitate various Django functionalities (As you will see in Figure 2.5). However, for the purpose of our discussion, we will focus on the essential tables that were specifically defined through Django models. These tables capture the core functionality and data storage requirements of the Tradehub web application.



Nom	Type	Schéma
Tables (14)		
auth_group		CREATE TABLE "auth_group" ("id"
auth_group_permissions		CREATE TABLE "auth_group_perm
auth_permission		CREATE TABLE "auth_permission"
auth_user		CREATE TABLE "auth_user" ("id" ir
auth_user_groups		CREATE TABLE "auth_user_groups
auth_user_jobs		CREATE TABLE "auth_user_jobs" (
auth_user_user_permissions		CREATE TABLE "auth_user_user_p
django_admin_log		CREATE TABLE "django_admin_loc
django_content_type		CREATE TABLE "django_content_ty
django_migrations		CREATE TABLE "django_migrations
django_session		CREATE TABLE "django_session" (
sqlite_sequence		CREATE TABLE sqlite_sequence(na
tradesmen_category		CREATE TABLE "tradesmen_catego
tradesmen_job		CREATE TABLE "tradesmen_job" ('

Figure 2.6 Database Structure

2.3.4.1 Django_admin_log table:

This table contains eight columns shown in Figure 2.6:

- Id: An identification number that is automatically generated by the database.
- Object_id: stores the ID of the specific object that was modified or affected by the administrative action.
- Object_repr: stores the name of the user or the title of a post that an action was performed on.
- Action_flag: stores numerical values that correspond to different types of actions. These values are predefined constants in Django's source code, such as (1=Adittion 2=Change, 3=Deletion).
- Change_message: stores an optional descriptive message associated with a specific action performed in the Django admin interface
- Content_type_id: A foreign key to the table “Django_content_type”
- User_id: Stores the users_id (which in our case are tradesmen)
- Action_time: Record the time that the specific action happened at.

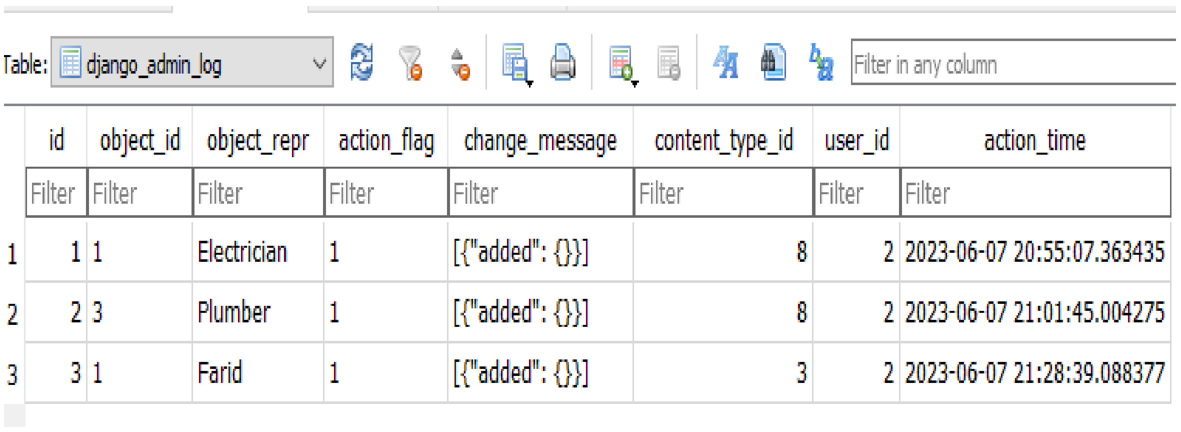


Table: django_admin_log

Filter in any column

	id	object_id	object_repr	action_flag	change_message	content_type_id	user_id	action_time
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	Electrician	1	[{"added": {}}]	8	2	2023-06-07 20:55:07.363435
2	2	3	Plumber	1	[{"added": {}}]	8	2	2023-06-07 21:01:45.004275
3	3	1	Farid	1	[{"added": {}}]	3	2	2023-06-07 21:28:39.088377

Figure 2.7 Django_admin_log table

2.3.4.2 Django_content_type:

This table contains three columns shown in table 2.7:

- Id: An identification number that is automatically generated by the database.
- App_label: represents the label or name of the Django application associated with the content type
- Model: represents the name of the model associated with the content type. It is used by Django's content type framework to manage and retrieve content types dynamically at runtime.

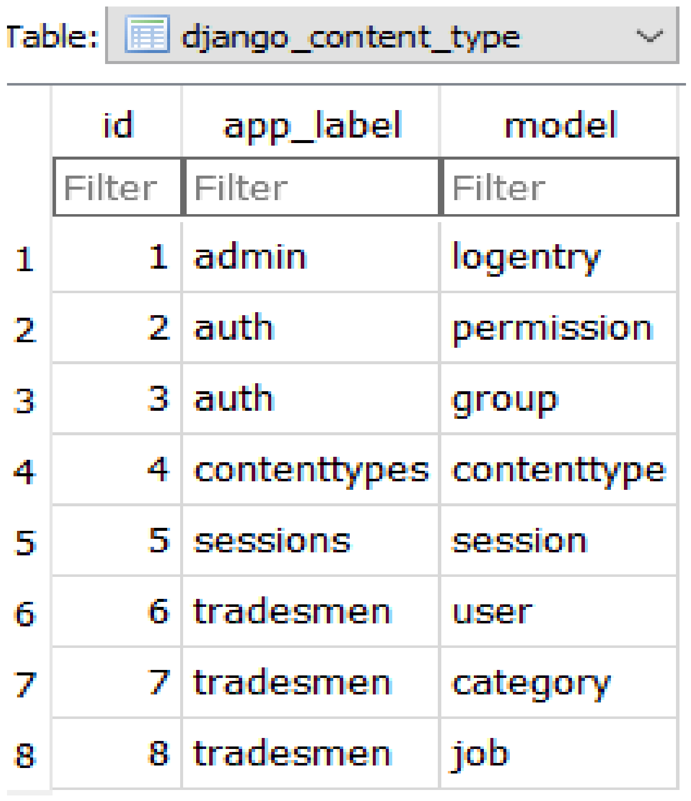


Table: django_content_type

	id	app_label	model
	Filter	Filter	Filter
1	1	admin	logentry
2	2	auth	permission
3	3	auth	group
4	4	contenttypes	contenttype
5	5	sessions	session
6	6	tradesmen	user
7	7	tradesmen	category
8	8	tradesmen	job

Figure 2.8 Django_content_type table

2.3.4.3 Tradesmen_category:

It contains 2 columns shown in Figure 2.8:

- Id: An identification number that is automatically generated by the database.
- Name: The names of each category

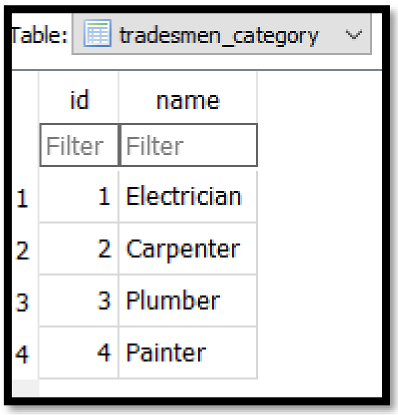


Table: tradesmen_category

	id	name
	Filter	Filter
1	1	Electrician
2	2	Carpenter
3	3	Plumber
4	4	Painter

Figure 2.9 tradesmen_category table

2.3.4.4 Tradesmen_job:

It contains 7 columns shown in Figure 2.9:

- Id: An identification number that is automatically generated by the database.
- Title: the title of the job post

- Description: the main part of the post where the user can describe the job he is offering and his contact information
- Posted: the date that the post was added to website
- Profile_image: an image attached with the job post
- Label: location of operation of the user (tradesmen)
- User_id: an identification number unique to each registered user

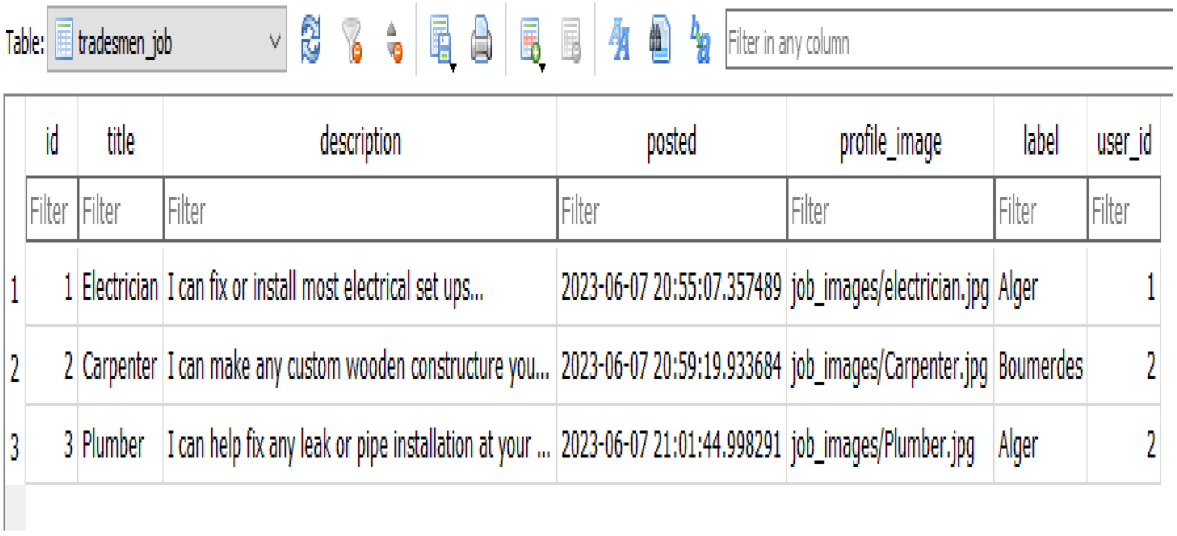


Table: tradesmen_job

	id	title	description	posted	profile_image	label	user_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Electrician	I can fix or install most electrical set ups...	2023-06-07 20:55:07.357489	job_images/electrician.jpg	Alger	1
2	2	Carpenter	I can make any custom wooden constructure you...	2023-06-07 20:59:19.933684	job_images/Carpenter.jpg	Boumerdes	2
3	3	Plumber	I can help fix any leak or pipe installation at your ...	2023-06-07 21:01:44.998291	job_images/Plumber.jpg	Alger	2

Figure 2.10 tradesmen_job table

2.3.4.5 Users_comments:

It contains 4 columns shown in Figure 2.10:

- Id: The unique identifier for each comment.
- user_id: The foreign key referencing the user who made the comment.
- Content: The actual content of the comment.
- created_at: The timestamp indicating when the comment was created.

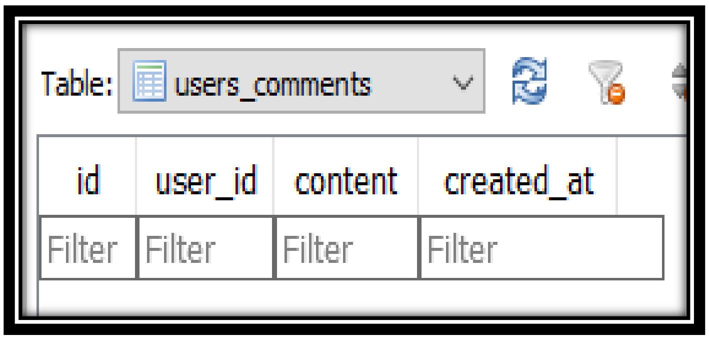


Table: users_comments

id	user_id	content	created_at
Filter	Filter	Filter	Filter

Figure 2.11 users_comments table

2.4 Conclusion:

In this design chapter, we presented a comprehensive use case model that captured the interactions between actors and the Tradehub system's functionalities. We emphasized the importance of a well-designed database as a central repository for efficient data retrieval, storage, and manipulation, contributing to a seamless user experience.

We explored the database structure, focusing on the essential tables defined through Django models. Each table and its relevant columns were described, providing insights into how the system organizes and stores data.

In the next chapter, we will delve into the implementation phase, discussing the development process, data migration, version control, and the integration of various technologies. By examining the implementation details, we will gain a comprehensive understanding of how the design concepts discussed in this chapter were translated into a fully functional Tradehub web application.

Chapter Three: Implementation & Results

Chapter 3: Implementation & Results

3.1 Introduction:

In this chapter, we will delve into the implementation phase of the Tradehub project, where we bring the designed concepts to life and present the tangible results. Building upon the tools and roles discussed in previous chapters, we will explore the functionalities of the web application and showcase the user interfaces that have been developed. Additionally, we will evaluate the performance, functionality of the Tradehub platform, discussing the achieved milestones and addressing any identified limitations.

3.2 Development Process:

3.2.1 Iterative Development:

The Tradehub application was developed iteratively, focusing on incremental progress and continuous improvement. I broke down the project into smaller tasks and milestones, setting achievable goals for each iteration. Regular self-evaluations and progress reviews were conducted to ensure the project was on track and aligned with the initial objectives.

3.2.2 Technology Stack:

The Tradehub web application leveraged the power of the Django framework for seamless back-end development. Django provided a comprehensive set of tools and features that simplified the implementation of essential functionalities. The built-in user authentication system allowed for secure user management, including registration, login, and password recovery. Django's ORM (Object-Relational Mapping) facilitated easy integration with the database, ensuring efficient data retrieval and manipulation. The framework's URL routing mechanism enabled clean and organized URL patterns, making it simple to define routes and connect them to the appropriate views. Moreover, Django models were employed to define the database structure and relationships between entities. Models such as User, Job, Category, Comment provided a structured representation of the data, enabling seamless data management and retrieval within the Tradehub application.

On the front-end, the Tradehub web application employed a combination of HTML, CSS, JavaScript to create an engaging user interface. HTML provided the backbone for structuring the content and defining the various elements of the application. CSS was utilized to style and customize the visual presentation, ensuring a visually appealing and cohesive design. JavaScript enhanced interactivity, allowing for dynamic functionality such as client-side form validation and real-time updates. Additionally, Django's built-in templating language was utilized to generate dynamic HTML content, enabling the inclusion of variables and logic in the presentation layer.

By leveraging the capabilities of Django for back-end development and utilizing HTML, CSS, JavaScript for the front-end, the Tradehub web application offered a powerful and user-friendly platform that streamlined user interactions, provided robust data management, and delivered an aesthetically pleasing user experience.

3.2 Code Implementation:

3.2.1 Models.py:

The **models.py** file plays a crucial role in the Tradehub project as it defines the data models used for storing and managing various entities. These models represent the structure of the application's database and facilitate the organization and retrieval of data.
Now let's dive into the implementation details of each key model in the Tradehub project

3.2.1.1 Category Model:

```
class Category(models.Model):
    name = models.CharField(max_length=100)
    # Add any other fields as needed

    def __str__(self):
        return self.name
```

- Title: Category Model
- Description: The Category model represents a category of tradesmen available in the Tradehub application. It serves as a way to classify tradesmen based on their specialization. The model includes a single field, **name**, which stores the name of the category.
- Use: The Category model allows tradesmen to be grouped into specific categories, enabling users to search and filter tradesmen based on their desired category or specialization.

3.2.1.2 CustomUserManager Model:

```
class CustomUserManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        # Create and save a new user
        if not email:
            raise ValueError('The Email field must be set')
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, password=None, **extra_fields):
        # Create and save a new superuser
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)
        return self.create_user(email, password, **extra_fields)
```

- Title: Custom User Manager
- Description: The CustomUserManager class extends the BaseUserManager provided by Django and is responsible for creating and managing user accounts. It includes methods for creating regular users and superusers with enhanced privileges.
- Use: The CustomUserManager handles the creation and management of user accounts in the Tradehub application. It ensures that user data is properly validated, encrypted, and stored in the database.

3.2.1.3 User Model:

```
class User(AbstractUser):
    phone_number = models.CharField(max_length=20)
    address = models.CharField(max_length=255)
    profile_picture = models.ImageField(upload_to='profile_pictures/',
blank=True, null=True)
    # Add any other fields you need
    objects = CustomUserManager()
    class Meta:
        # Add the related_query_name attribute to avoid clashes with the
default User model
        verbose_name = 'User'
        db_table = 'auth_user'
        swappable = 'AUTH_USER_MODEL'

    def __str__(self):
        return self.username

    # Relationship with Job model
    jobs = models.ManyToManyField('tradesmen.Job',
related_name='tradesmen_user', related_query_name='tradesmen_user',
blank=True)

    def delete_job(self, job_id):
        try:
            job = self.jobs.get(id=job_id)
            job.delete()
        except Job.DoesNotExist:
            # Handle the case when the job does not exist
            pass
```

- Title: User Model
- Description: The User model extends Django's AbstractUser model to represent a user in the Tradehub application. It includes additional fields such as **phone_number**, **address**, and **profile_picture**. The model also establishes a many-to-many relationship with the Job model, allowing users to create and manage multiple jobs.
- Use: The User model serves as the foundation for user authentication, profile management, and job association. It stores user-specific information, such as contact details and profile pictures, and provides methods for job-related operations, including job creation and deletion.

3.2.1.4 Job Model:

```
class Job(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True,
related_name='tradesmen_job')
    title = models.CharField(max_length=100)
    description = models.TextField()
    #location = models.CharField(max_length=100)
    posted = models.DateTimeField(auto_now_add=True)
    #category = models.ForeignKey(Category, on_delete=models.CASCADE)
    profile_image = models.ImageField(upload_to='job_images/', blank=True)
    label = models.CharField(max_length=100, blank=True)

    def __str__(self):
        return self.title
```

- Title: Job Model
- Description: The Job model represents a job posted by a user in the Tradehub application. It contains fields such as **title**, **description**, **posted**, and **label**. The model maintains a foreign key relationship with the User model to associate jobs with the respective user who posted them.

- Use: The Job model serves as a central entity for storing job-related information. It enables users to create and manage their posted jobs, and it provides fields to store details like the job title, description, posting date, and labels for categorization.

Together, these models form the backbone of the Tradehub application's data structure, allowing for the efficient storage, retrieval, and manipulation of information related to categories, users, and jobs.

3.2.2 views.py:

The `views.py` file in the Tradehub project contains the implementation of various views, which handle the logic behind rendering templates, processing form submissions, and interacting with the models. Let's explore the key functionalities provided by these views.

3.2.2.1 Customer service function:

```python
def customer_service(request):
    if request.method == 'POST':
        form = CustomerServiceForm(request.POST)
        if form.is_valid():
            email = form.cleaned_data['email']
            message = form.cleaned_data['message']
            send_mail(
                'Customer Service',
                f'Email: {email}\nMessage: {message}',
                'tradehub.dz@gmail.com',
                ['tradehub.dz@gmail.com'],
                fail_silently=False,
            )
            return render(request, 'customer_service_success.html')
    else:
        form = CustomerServiceForm()

    return render(request, 'customer_service.html', {'form': form})
```

- Description: This view handles the submission of the customer service form. Upon receiving a POST request, the form data is validated, and an email is sent to the customer service email address containing the customer's email and message. Upon successful submission, the user is redirected to a success page.

3.2.2.2 Create-job function:

```python
@login_required
def create_job(request):
    if request.method == 'POST':
        form = JobForm(request.POST, request.FILES)
        if form.is_valid():
            job = form.save(commit=False)
            job.user = request.user  # Set the user for the job
            job.save()

            job_data = {
                'title': job.title,
                'description': job.description,
                'profile_image': job.profile_image.url if job.profile_image
else '',
                'label': job.label,
                'posted': job.posted.strftime('%Y-%m-%d')
            }

            return JsonResponse({'job': job_data})
        else:
            return JsonResponse({'error': 'Invalid form data'})

    return JsonResponse({'error': 'Invalid request method'})
```

- Description: This view handles the creation of a new job. Upon receiving a POST request with valid form data, a new job instance is created and associated with the currently authenticated user. The job details are then saved, and a JSON response is returned with the created job data.

3.2.2.3 Delete-job function:

```
@login_required
def delete_job(request):
    if request.method == 'POST':
        job_id = request.POST.get('job_id')
        job = get_object_or_404(Job, id=job_id)

        if job.user != request.user:
            raise PermissionDenied("You don't have permission to delete this
job.")

        job.delete()
        return JsonResponse({'message': 'Job deleted successfully'})
    else:
        return JsonResponse({'message': 'Invalid request method'})
```

- Description: This view handles the deletion of a job. When a POST request is received, the corresponding job is fetched based on the provided job ID. If the authenticated user is the owner of the job, the job is deleted, and a success message is returned as a JSON response.

3.2.2.4 Index function:

```
def index(request):
    jobs = Job.objects.all().order_by('-posted')
    return render(request, 'index.html', {'jobs': jobs})
```

- Description: This view retrieves all the jobs from the database and renders the `index.html` template, displaying the jobs in descending order of their posting dates.

3.2.2.5 Register function:

```
def register(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')  # Redirect to the login page after
successful registration
    else:
        form = UserRegistrationForm()

    return render(request, 'registration.html', {'form': form})
```

- Description: This view handles user registration. Upon receiving a POST request with valid form data, a new user account is created, and the user is redirected to the login page for authentication.

3.2.2.6 User_page function:

```
def user_page(request, username):
    user = User.objects.get(username=username)
    jobs = user.jobs.all()
    context = {'user': user, 'jobs': jobs}
    return render(request, 'tradesmen/user_page.html', context)
```

- Description: This view retrieves a specific user based on the provided username and renders the `user_page.html` template. It fetches all the jobs associated with the user and passes them to the template for display.

These views play a crucial role in handling user interactions, rendering templates, and processing data in the Tradehub application.

3.2.3 admin.py:

```
from django.contrib import admin
from .models import Job, Category, User

admin.site.register(Job)
admin.site.register(Category)
admin.site.register(User)
```

The code registers the **Job**, **Category**, and **User** models with the Django admin interface. This enables administrators to perform various administrative tasks related to these models, such as creating, editing, and deleting records directly through the admin interface. By registering these models, you provide an easy-to-use interface for managing and manipulating the data stored in these models.

3.3 Application Interfaces:

3.3.1 Login Interface:

The login interface in our web application is a vital component that enables users to access their accounts and perform various actions within the web application. It serves as a gateway for tradesmen, customers, and admins to securely log in and access personalized features and functionalities.

The login interface consists of a user-friendly form that prompts users to enter their login credentials. It includes two main fields: the username field and the password field. Users need to provide their registered username and password to authenticate their identity and gain access to their respective accounts.

In case a user forgets their password, the login interface offers a password recovery option. This feature allows users to initiate the password retrieval process, typically through an email or verification link, enabling them to reset their password and regain access to their account.

The login interface plays a pivotal role in Tradehub by controlling access to essential features such as creating and managing job posts, viewing previous job postings, and administering user accounts. It establishes a secure and reliable connection between users and the web application, providing a seamless and efficient user experience.

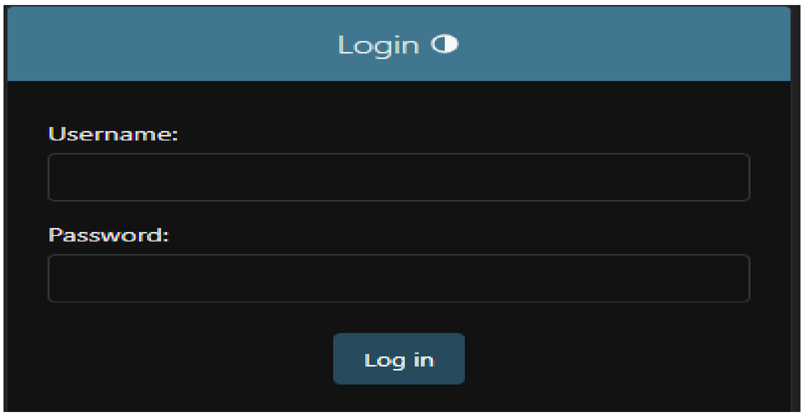


Figure 3.12 Login interface

3.3.2 Register Interface:

The register interface in Tradehub provides a straightforward and convenient way for new users to create an account and join the platform. It serves as an entry point for individuals who wish to access the features and services offered by Tradehub.

To begin the registration process, users are directed to the register page by clicking on the "Join Us" button located on the main index page. The register page presents a registration form that captures essential user information. The form includes the following entries:

1. Username: Users are prompted to enter a unique username that will serve as their identification within the Tradehub platform. The registration form provides clear instructions and guidelines regarding username restrictions, such as character limits or specific character requirements.

2. Email: Users are required to provide a valid email address, which will serve as a primary contact method and enable communication between Tradehub and the user.

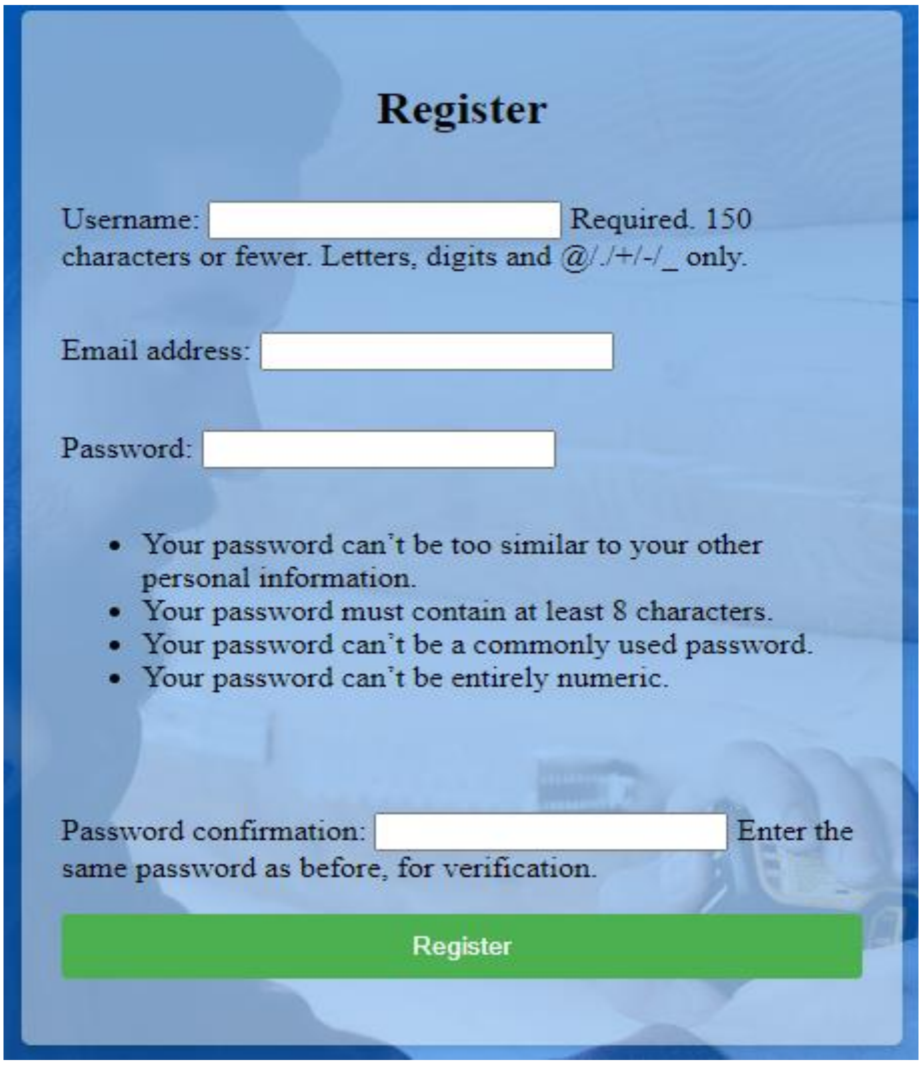
3. Password: Users must create a secure password that meets specific criteria, such as minimum length or the inclusion of special characters. The registration form includes comments and instructions to guide users in creating a strong and reliable password.

4. Confirm Password: To ensure accuracy and prevent input errors, users are prompted to re-enter their chosen password in a separate field for confirmation.

The registration form in Tradehub incorporates validation checks to ensure that all required fields are filled out before proceeding. If a user fails to provide the necessary information or misses a required field, an appropriate notification or error message is displayed, indicating the missing or incorrect input.

Once all the required information is provided and the form is successfully submitted, the user's registration data is processed, and an account is created. The register interface plays a crucial role in expanding the Tradehub community by facilitating user onboarding and providing a seamless registration experience.

By offering a user-friendly and intuitive registration process, Tradehub aims to encourage new users to join the platform and explore its various functionalities, including the ability to create job posts, access personalized features, and engage with the Tradehub community.


Register

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email address:

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

Figure 3.13 Register Interface

After registering, the user can now access his user page by logging in. There are two types of user pages in this project, admin interface and tradesmen interface. The interfaces of various users will be discussed in the following sections.

3.3.3 Admin Dashboard:

3.3.3.1 Admin home interface:

The admin interface in Tradehub provides an essential tool for managing and overseeing various aspects of the platform. It offers two main sections: the Tradesmen section and the Authentication and Authorization section.

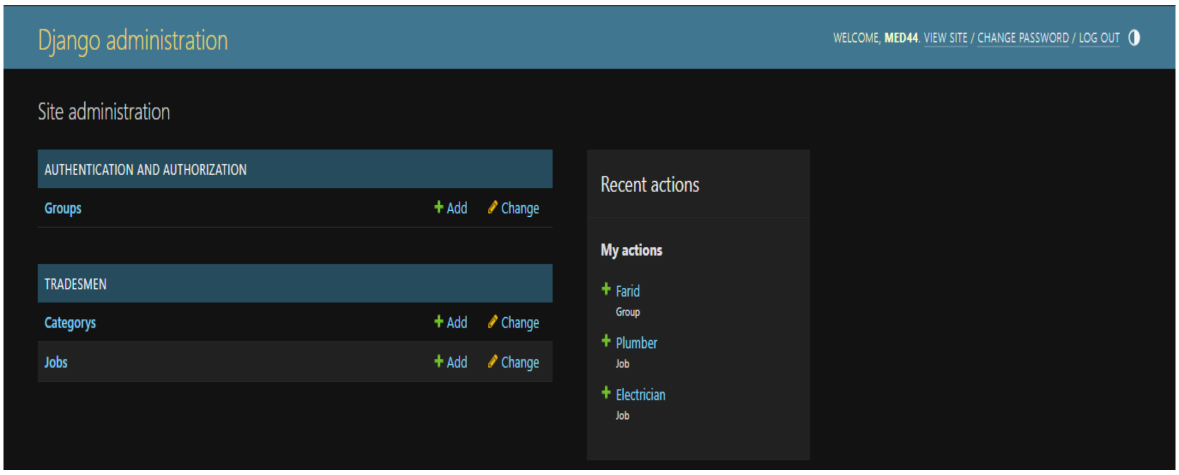


Figure 3.14 Admin home interface

Our admin home interface is made of header and a main area which is divided into two parts mainly "site administration" and "Recent actions".

From the header you can either redirect to the main page of the website or change the password of your account, and of course the option to log out of your account when you are done with your actions.

3.3.3.2 Admin User_management interface:

Now for the Authentication and Authorization section, when you click on the group you will be directed to another page as shown in Figure 3.14 where you will be able to see all the users and groups available and you will have the ability to set the permission for any user or you can add/delete users from the platform.

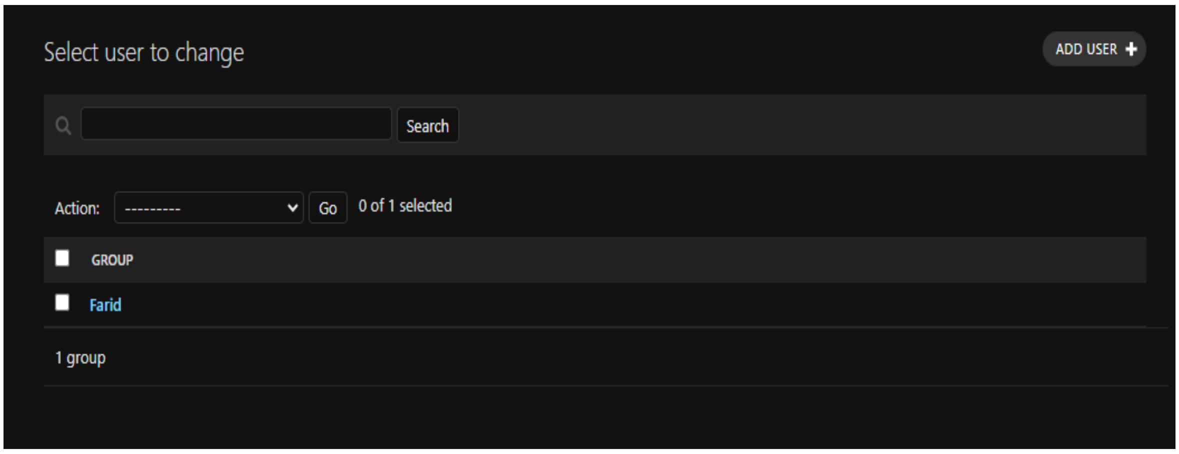
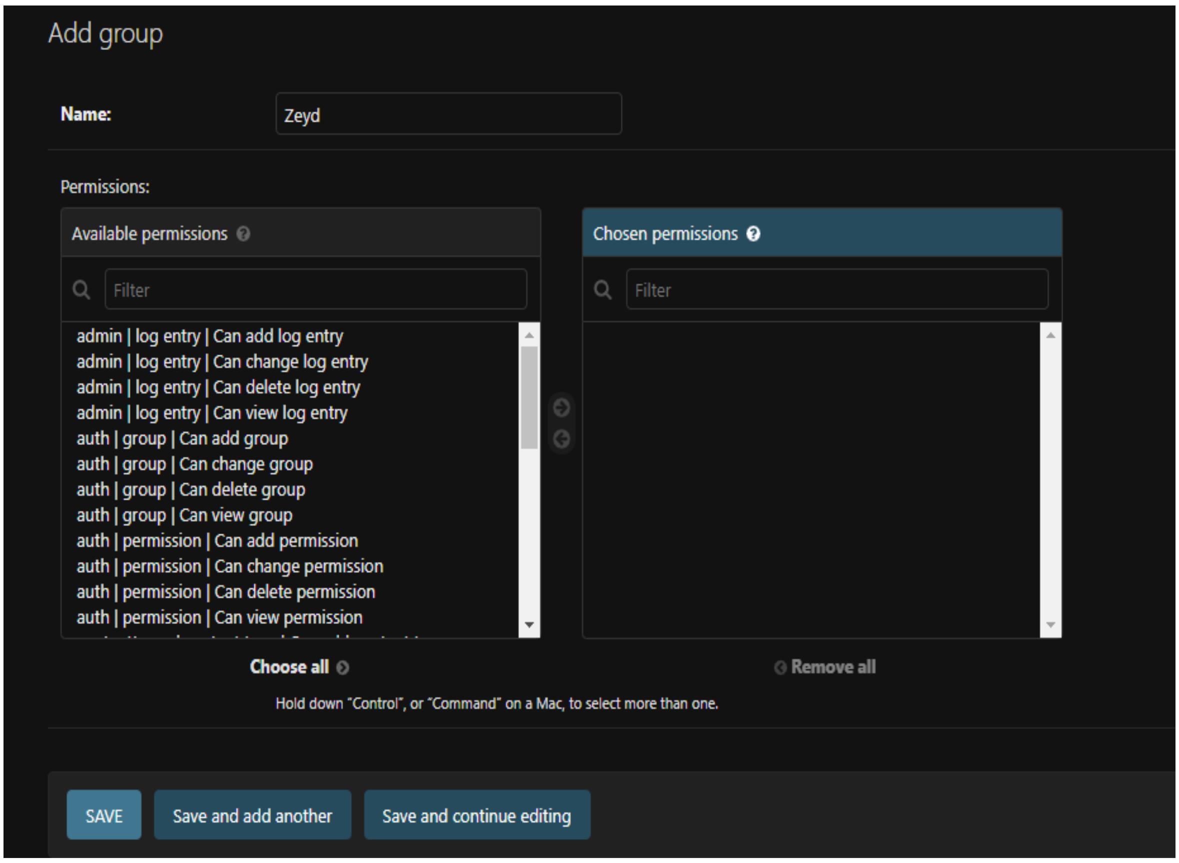


Figure 3.15 User management interface

As seen from the figure above, we were redirected to the list of users. Currently there is only one, as an admin we can add new users by cliking the button "ADD USER" and then setting his permission as shown in Figure 3.15:


Add group

Name: Zeyd

Permissions:

Available permissions

admin | log entry | Can add log entry
admin | log entry | Can change log entry
admin | log entry | Can delete log entry
admin | log entry | Can view log entry
auth | group | Can add group
auth | group | Can change group
auth | group | Can delete group
auth | group | Can view group
auth | permission | Can add permission
auth | permission | Can change permission
auth | permission | Can delete permission
auth | permission | Can view permission

Chosen permissions

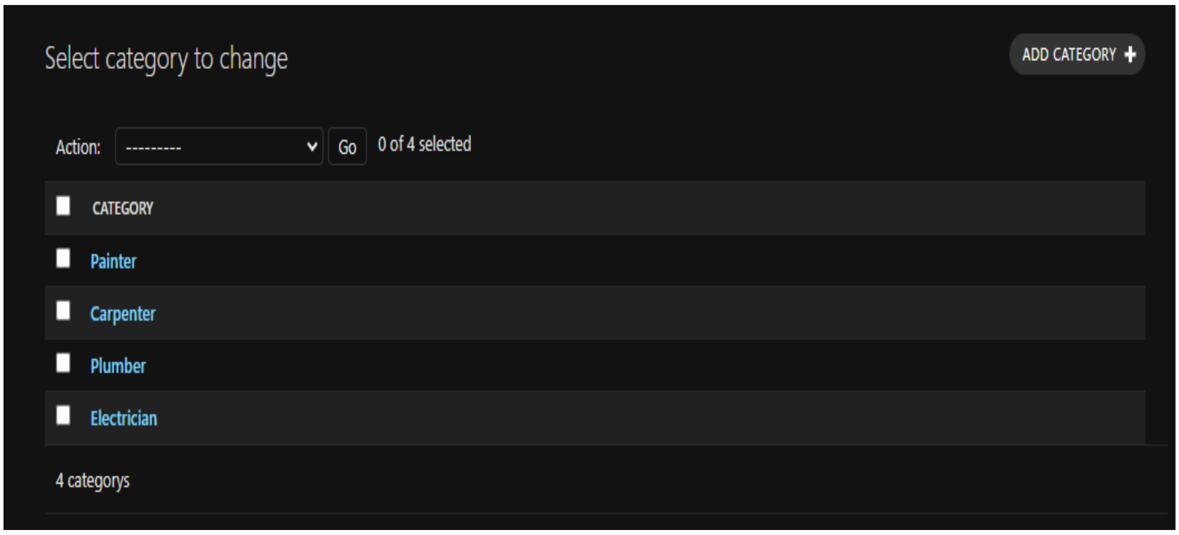
Hold down "Control", or "Command" on a Mac, to select more than one.

Figure 3.16 Adding new users interface

3.3.3.3 Admin Tradesmen interface:

Now lets go to the Tradesmen section which has two parts Categorys and Jobs.

By cliking on Categorys we will be redirected to another page where we can see all types of tradesmen there is on the platform, as shown in Figure 3.16:


Select category to change

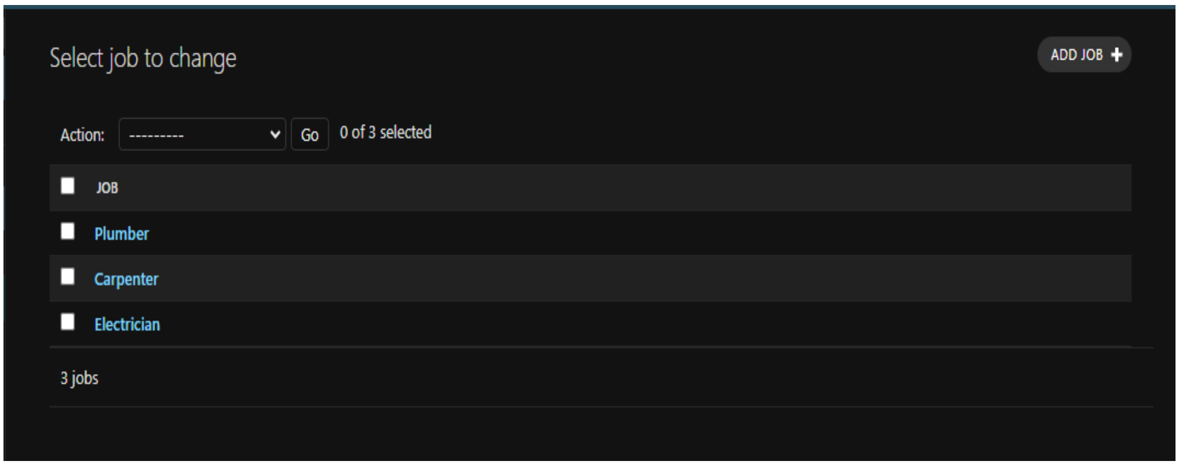
Action: --------- 0 of 4 selected

CATEGORY
Painter
Carpenter
Plumber
Electrician

4 categorys

Figure 3.17 Category interface

Now we go to the Jobs section, where we can view all the jobs listed on our platform, and have the ability to add new one in our name or in the name of another user, we can also delete or edit job posts, as shown in Figure 3.17 and 3.18:


Select job to change

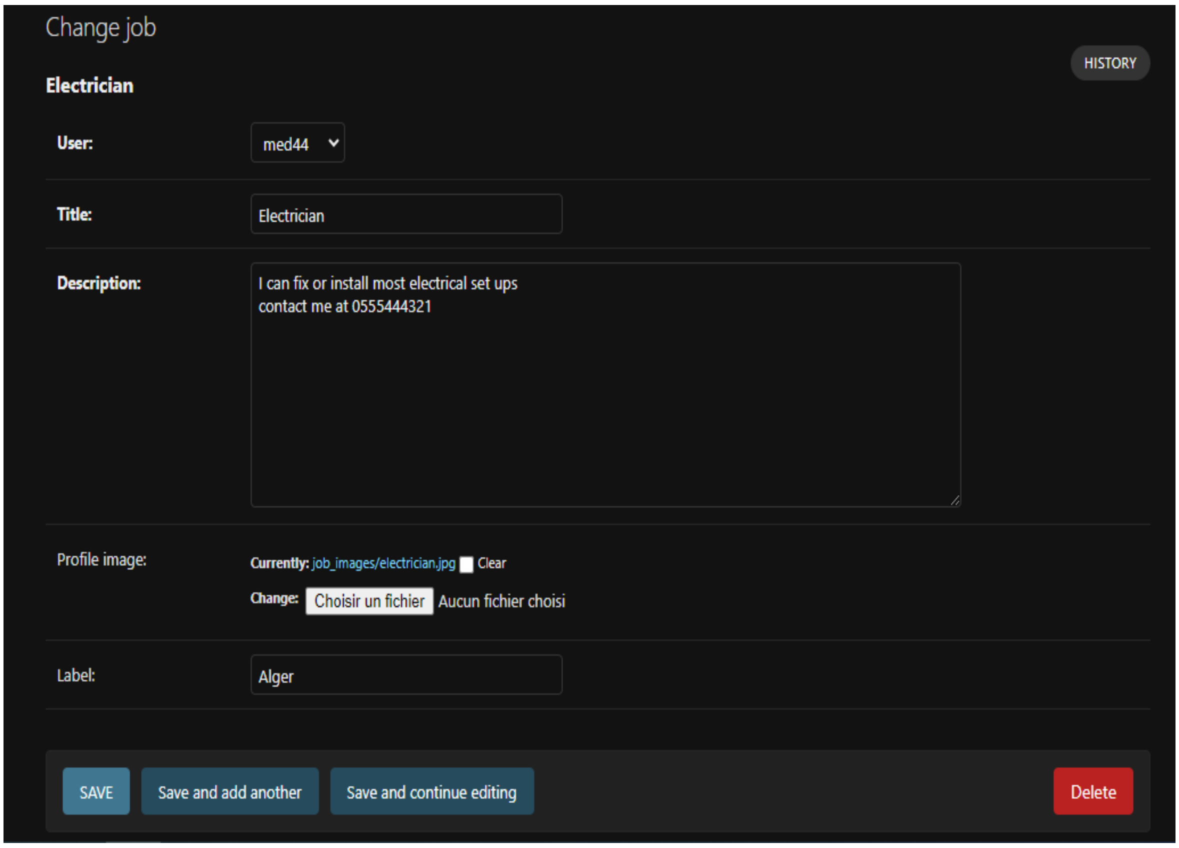
ADD JOB +

Action: --------- Go 0 of 3 selected

	JOB
☐	Plumber
☐	Carpenter
☐	Electrician

3 jobs

Figure 3.18 Job list Interface


Change job

HISTORY

Electrician

User:	med44
Title:	Electrician
Description:	I can fix or install most electrical set ups contact me at 0555444321
Profile image:	**Currently:** job_images/electrician.jpg ☐ Clear **Change:** Choisir un fichier Aucun fichier choisi
Label:	Alger

SAVE | Save and add another | Save and continue editing | Delete

Figure 3.19 Edit Posts Interface

3.3.4 Web application main interface:

In this section we will describe the main interface that any user will se when first accessing our platform, and highlight its different parts

3.3.4.1 Main page header:

The main page interface features a header section with various options to navigate through the Tradehub website as seen in Figure 3.19. These options include "Jobs," "Comments," "Blog," and a “footer”.

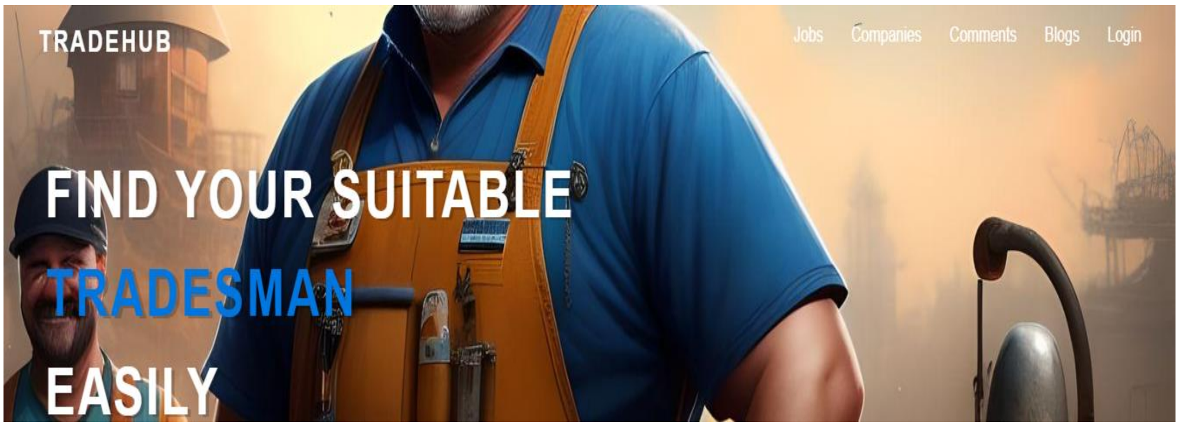
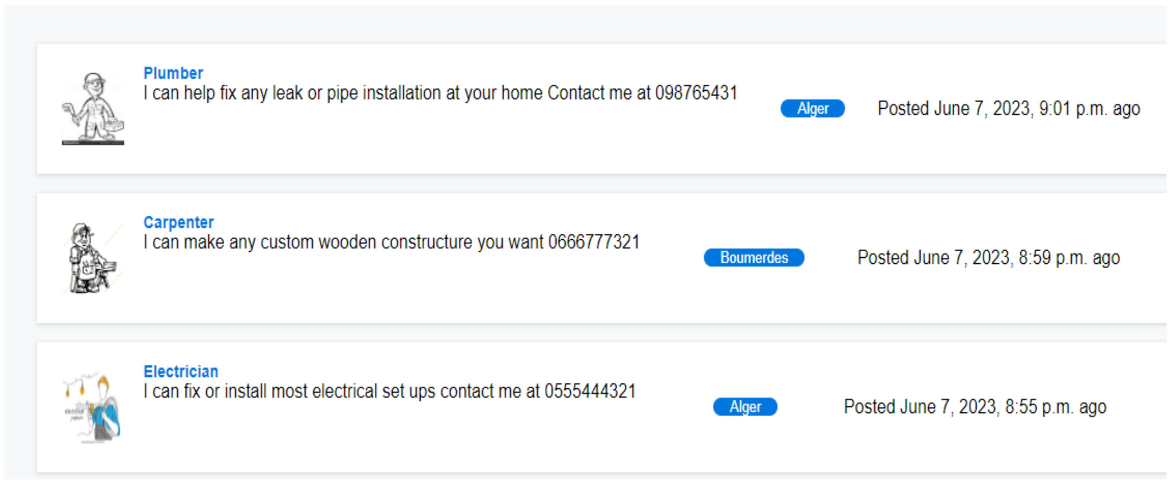


Figure 3.20 Main page Header

3.3.4.2 Jobs section:

Clicking on the "Jobs" option in the header directs users to the job list, where they can find previously posted jobs. Tradesmen who wish to add a new job are required to log in. After logging in, a job creation form appears above the job list. By filling out the form and clicking "Create," a new job is posted as highlighted in the Figures 3.20 and 3.21. This functionality allows tradesmen to easily add new job listings and manage their offerings.
As you can see from the Figure bellow, before login in the user can’t post a new job they can only observe the list.



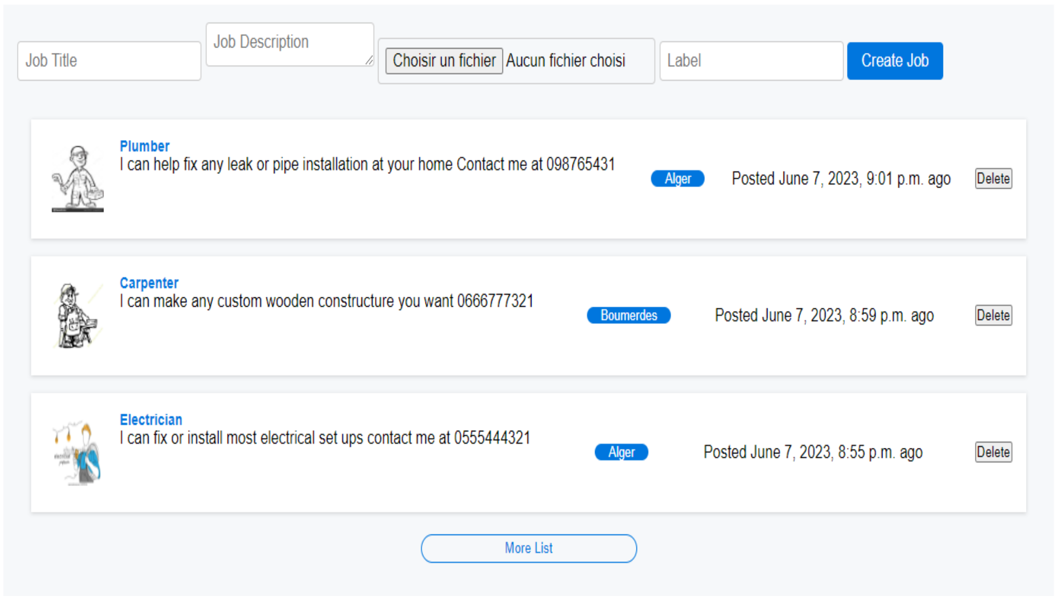
Plumber
I can help fix any leak or pipe installation at your home Contact me at 098765431
Alger
Posted June 7, 2023, 9:01 p.m. ago

Carpenter
I can make any custom wooden constructure you want 0666777321
Boumerdes
Posted June 7, 2023, 8:59 p.m. ago

Electrician
I can fix or install most electrical set ups contact me at 0555444321
Alger
Posted June 7, 2023, 8:55 p.m. ago

Figure 3.21 Jobs section before login

And now, after the user logged into their account they can see a form appearing above the list, by filling it they can create new job posts that would be displayed in the list.



Plumber
I can help fix any leak or pipe installation at your home Contact me at 098765431
Alger
Posted June 7, 2023, 9:01 p.m. ago
Delete

Carpenter
I can make any custom wooden constructure you want 0666777321
Boumerdes
Posted June 7, 2023, 8:59 p.m. ago
Delete

Electrician
I can fix or install most electrical set ups contact me at 0555444321
Alger
Posted June 7, 2023, 8:55 p.m. ago
Delete

Figure 3.22 Jobs section after login

3.3.4.3 Join us section:

Located below the job list, there is a section with a button labeled "Join Us" as seen in Figure 3.22. Clicking this button redirects users to the registration form we showed in Figure 3.12, where they can create a new account to access additional features and services of the platform such as posting their services.

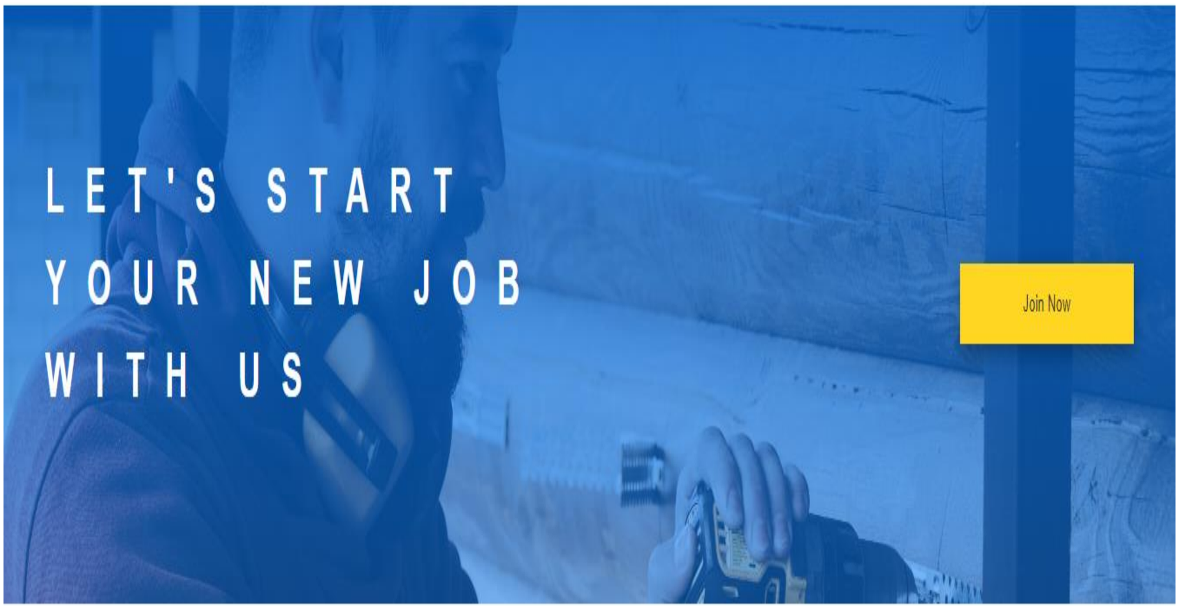
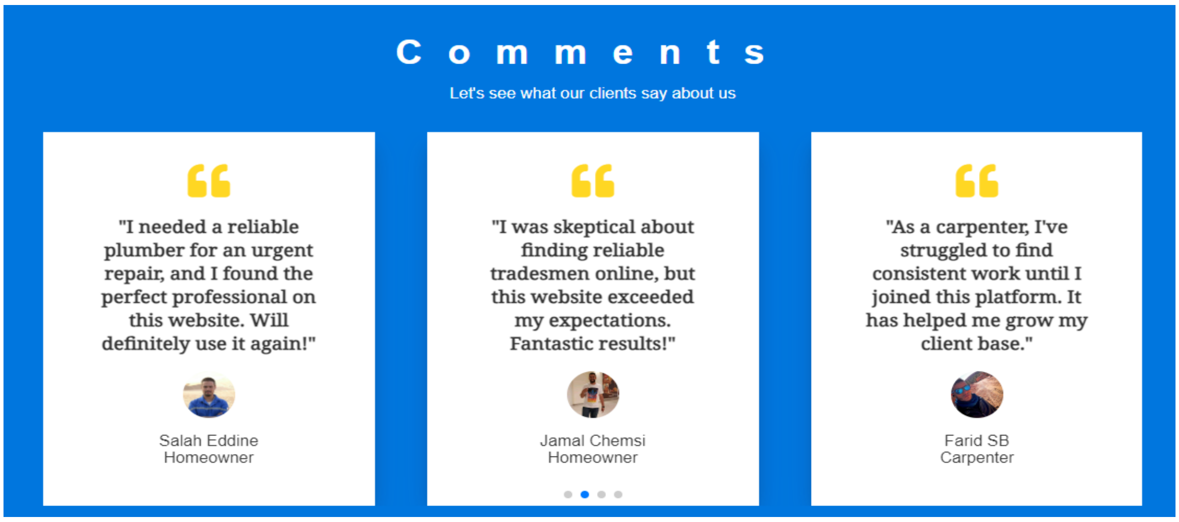


Figure 3.23 Join us section

3.3.4.4 Comments section:

When users select the "Comments" option from the header, they are presented with a rotating display of comments posted by other users. The comments are displayed in the form of cards, allowing users to view and engage with the feedback and experiences shared by the Tradehub community.



Comments

Let's see what our clients say about us

"I needed a reliable plumber for an urgent repair, and I found the perfect professional on this website. Will definitely use it again!"

Salah Eddine
Homeowner

"I was skeptical about finding reliable tradesmen online, but this website exceeded my expectations. Fantastic results!"

Jamal Chemsi
Homeowner

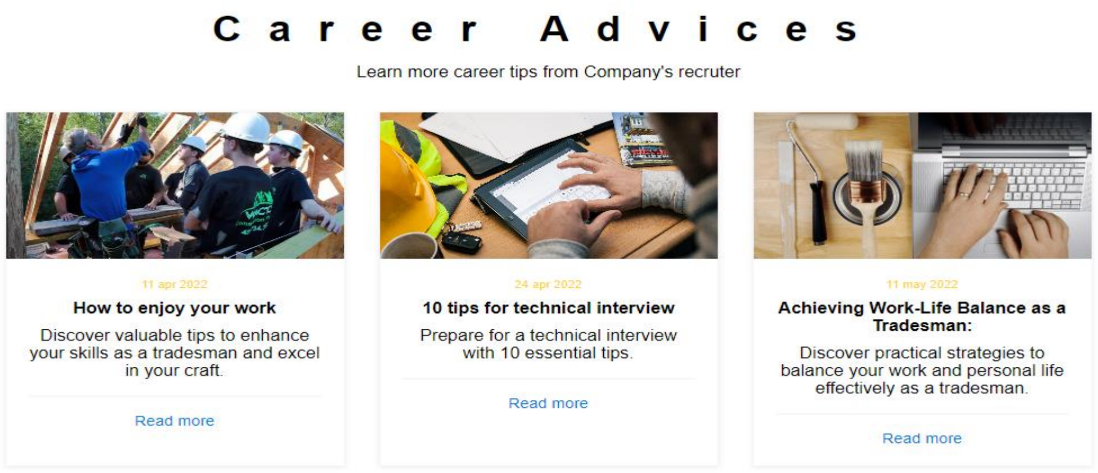
"As a carpenter, I've struggled to find consistent work until I joined this platform. It has helped me grow my client base."

Farid SB
Carpenter

Figure 3.24 Comment section

3.3.4.5 Blog section:

Selecting the "Blog" option scrolls the page further down to a dedicated section that contains three informative blog posts relevant to tradesmen as shown by Figure 3.23. Clicking on a specific blog post redirects users to a separate page where they can access detailed information and resources related to the topic of interest.



Career Advices

Learn more career tips from Company's recruter

11 apr 2022
How to enjoy your work
Discover valuable tips to enhance your skills as a tradesman and excel in your craft.
Read more

24 apr 2022
10 tips for technical interview
Prepare for a technical interview with 10 essential tips.
Read more

11 may 2022
Achieving Work-Life Balance as a Tradesman:
Discover practical strategies to balance your work and personal life effectively as a tradesman.
Read more

Figure 3.25 Blog section

3.3.4.6 Footer section:

The footer section appears at the bottom of the main page interface and includes various elements (Figure 3.25).

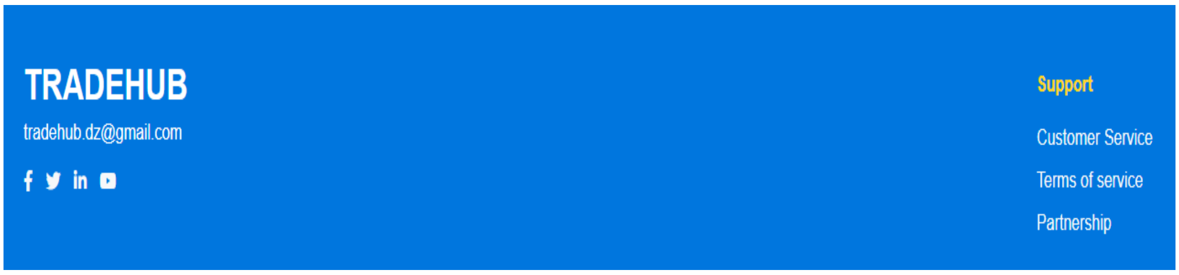


Figure 2.26 Footer section

One of the clickable texts within the footer is "Customer Service." Clicking on it redirects users to another page that features a form for submitting comments or inquiries. When users fill out the form and click the "Send" button, an email is sent to the Tradehub team, facilitating efficient communication and customer support(Figure 3.26).

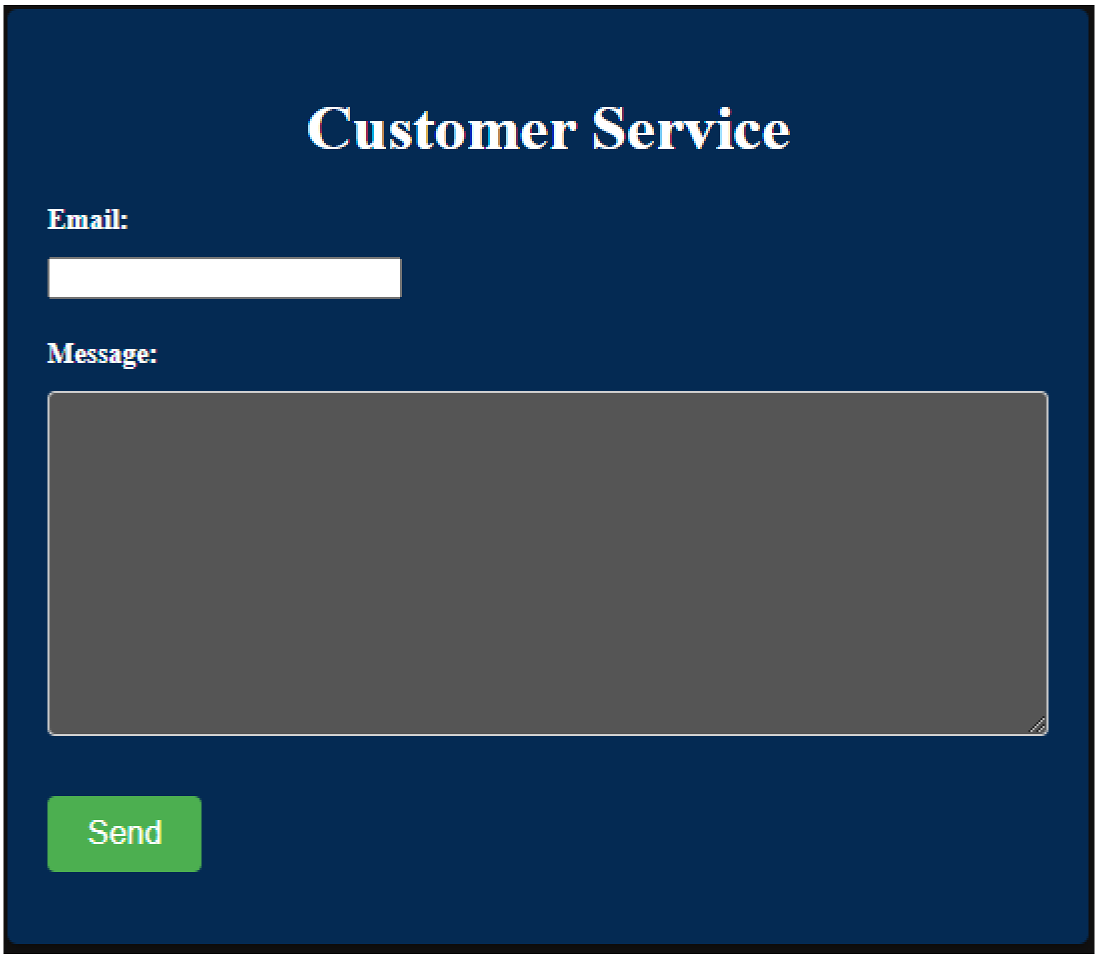


Figure 3.27 Customer service form

The main page interface serves as the central hub for users to explore job listings, access relevant information through the blog, engage with comments from the Tradehub community, and interact with essential features such as user registration and customer service. Its intuitive design and clear navigation contribute to an enhanced user experience on the Tradehub website.

3.4 Deployment and Hosting:

In order to make the Tradehub web application accessible to users, it is necessary to host and deploy the website. However, during the deployment process, several challenges were encountered when attempting to deploy the Tradehub website on various free hosting platforms. These challenges were primarily related to the compatibility of Python and SQLite versions.

Many free hosting platforms do not support the latest versions of Python and SQLite, which resulted in deployment errors and compatibility issues. It was observed that the Tradehub project was developed using the latest versions of these technologies, making it difficult to find a suitable free hosting platform that fully supported the required versions.

Considering these limitations, a decision was made to deploy and run the Tradehub web application locally. By deploying the website locally, we could ensure compatibility with the required Python and SQLite versions, eliminating the deployment errors experienced on external hosting services.

Although the Tradehub website is deployed locally, it is still possible to demonstrate the functionality and features of the application to stakeholders or users. By providing access to the locally hosted environment, users can interact with the Tradehub web application and experience its core features.

3.5 Functionality Evaluation:

The implemented features, including the login and registration system, job posting mechanism, comments section, and blog posts, were evaluated for their effectiveness and user satisfaction. The login and registration system provided a seamless experience, enabling users to easily create and access their accounts and post new jobs. The comments section helped give a positive image for our website, while the blog posts offered valuable information to tradesmen.

During the evaluation, some challenges were encountered, such as occasional delays in job posting updates or the misplacement the of profile image right after the job is posted and minor issues with the comments rotation feature where it wont go back to the first comment after reaching the end. These challenges were promptly addressed and resolved to ensure a smooth user experience.

3.6 Performance and Scalability:

Tradehub demonstrated commendable performance throughout the evaluation period. The platform maintained efficient response times and loading speeds, ensuring a seamless browsing experience for users, there was no delay or staggering of the website when scrolling up and down. Performance optimizations, including database indexing and caching mechanisms, were implemented to enhance platform scalability.

Unfortunately we didn’t manage to deploy the website online, so we couldn’t test its performance when multiple users were on it.

3.7 Comparison with Initial Goals:

The achieved results were compared against the initial goals and requirements defined for the Tradehub project. The platform met the majority of the set objectives, delivering essential features and functionalities as planned. While minor adjustments and modifications were made during the implementation phase, they were aligned with the project's scope and contributed positively to the overall outcome.

One point that we weren’t able to implement is a rating system for tradesmen that would decide the placement of their job posts on the job list, so that the ones with the highest ratings would have their jobs displayed at the front.

3.8 Limitation:

Despite the successful implementation and positive user feedback, there are certain limitations and areas for future improvement. One limitation is the lack of support for newer versions of Python and SQLite on free hosting platforms, which restricted the ability to deploy the website externally. Another limitation is the lack of security measure since our website only included basic user authentication and authorization mechanisms As a result, the platform was deployed locally for evaluation purposes only.

3.9 Conclusion:

In this combined chapter, we have presented the implementation details and evaluated the functionality, performance of the Tradehub web application. Throughout the implementation process, we focused on leveraging Django as our core framework, along with HTML, CSS, and JavaScript to create an engaging front-end interface. Key features, such as user management and database connectivity, were successfully developed, providing a seamless user experience.

During the evaluation of the implemented features, we found that they aligned well with the intended requirements and demonstrated good performance during testing. However, it's important to note that scalability challenges may arise with an increasing number of users, and further optimizations might be required to ensure efficient scaling.

Comparing the system with the initial goals, Tradehub has successfully achieved its core functionalities, providing tradesmen a platform to showcase their services and clients an easy way to find the right professionals for their needs. However, we also identified certain limitations, such as limited mobile responsiveness, security considerations, and scalability issues. These limitations present opportunities for future improvement, focusing on enhancing usability, security, scalability, and overall user experience.

In conclusion, the Tradehub project has successfully delivered a functional web application, showcasing the potential of our technology stack and implementation efforts. While acknowledging the achievements, we also recognize the importance of ongoing enhancements to address the identified limitations and ensure the platform's continued growth and success.

General Conclusion:

The Tradehub project has successfully developed a web application to facilitate trade activities and connect tradesmen with customers. Through the utilization of technologies such as Django, HTML/CSS, and JavaScript, the project aimed to facilitate the connection between tradesmen and their customers by creating a user-friendly platform for job postings. The implementation phase involved the careful design and implementation of various interfaces, including login and registration pages, job posting functionality, and an admin interface for user and job management.

While facing challenges with hosting and deployment compatibility, the project overcame limitations to provide a functional trade platform for tradesmen and customers. The interfaces created, such as the login and registration pages, enable users to easily access and utilize the platform's features. The admin interface empowers administrators to manage users, job postings, and categories efficiently. Additionally, the main page provides a comprehensive overview of available jobs and encourages tradesmen to engage in trade activities. Overall, the Tradehub web application offers a seamless and user-friendly experience.

Future Works:

Looking ahead, future improvements could focus on enhancing mobile responsiveness, strengthening security measures, and ensuring scalability to accommodate a growing user base. By incorporating user feedback and continually refining the application, Tradehub has the potential to become a reliable and efficient platform for tradesmen and customers. The knowledge gained from this project serves as a foundation for future advancements in web development and trade management.

References

[1] [Online]. Available: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app. [Accessed 4 06 2023].

[2] [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/SPA. [Accessed 5 06 2023].

[3] [Online]. Available: https://www.codingninjas.com/codestudio/library/single-page-apps-vs-multi-page-apps. [Accessed 7 06 2023].

[4] [Online]. Available: https://www.pcmag.com/encyclopedia/term/web-client. [Accessed 7 06 2023].

[5] [Online]. Available: https://economictimes.indiatimes.com/definition/web-server. [Accessed 7 06 2023].

[6] [Online]. Available: https://www.techtarget.com/whatis/definition/HTTP-Hypertext-Transfer-Protocol. [Accessed 7 06 2023].

[7] [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL. [Accessed 7 06 2023].

[8] [Online]. Available: https://blog.hubspot.com/marketing/parts-url. [Accessed 7 06 2023].

[9] [Online]. Available: https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/p/Programming_language.htm. [Accessed 8 06 2023].

[10] [Online]. Available: https://www.freecodecamp.org/news/what-is-html-definition-and-meaning/. [Accessed 8 06 2023].

[11] [Online]. Available: https://www.w3schools.com/css/css_intro.asp. [Accessed 8 06 2023].

[12] [Online]. Available: https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/. [Accessed 8 06 2023].

[13] [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction. [Accessed 8 06 2023].

[14] [Online]. Available: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html. [Accessed 8 06 2023].

[15] [Online]. Available: https://www.methodsandtools.com/tools/staruml.php. [Accessed 10 06 2023].

[16] [Online]. Available: https://www.techopedia.com/definition/28227/django. [Accessed 10 06 2023].

[17] [Online]. Available: https://sendgrid.com/wp-content/uploads/2016/09/SendGrid-Implementation-Review.pdf. [Accessed 10 06 2023].

[18] [Online]. Available: http://sahet.net/htm/swdev9.html. [Accessed 10 06 2023].

[19] [Online]. Available: https://www.futura-sciences.com/tech/definitions/informatique-uml-3979/. [Accessed 10 06 2023].

[20] [Online]. Available: https://www.oracle.com/database/what-is-database/. [Accessed 10 06 2023].

[21] [Online]. Available: https://aloa.co/blog/relational-vs-non-relational-database-pros-cons#:~:text=A%20relational%20database%20is%20structured,of%20a%20laundry%20list%20order..

[22] [Online]. Available: https://www.sqlite.org/about.html. [Accessed 10 06 2023].