Order number: .....

Report submitted to the

UNIVERSITY OF MOHAMED BOUDIAF – MSILA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

In partial fulfillment of the requirements for the degree of

Master in **Computer science**

By

Mohamed Nafea, Baza

Chahrazed, Bouzahar

Mhenni: An Online Marketplace for On-demand Services

Defended in front of the jury composed of:

Mr. Mohamed Kamel	Associate Professor, University of M'Sila	President
Mr. Noureddine Amraoui	Associate Professor, University of M'Sila	Reporter
Mr. Seddik Bougherara	Associate Professor, University of M'Sila	Examiner

June, 2024

To my mother, father, siblings, and friends, you have been my pillars of support throughout my education journey. I must also acknowledge my professors, whose invaluable guidance and knowledge were crucial. I dedicate my graduation thesis to you, praying that the Almighty may grant you long life and bless you with goodness.

B. Mohamed Nafaa

By the end of this final project in my education journey, I had a special feeling when I wrote this dedication. This dissertation is dedicated to my beloved parents for their love and support throughout my life and for giving me the strength to reach and chase my dreams. I hope that I will never stop making you feel proud of me. To all my sisters and brothers who helped me in every single step. And I finally would like also to dedicate this dissertation to my friends.

B. Chahrazed

Acknowledgements

First and foremost, we express our profound gratitude to Allah for His mercy and guidance throughout this journey.

We would like to extend our heartfelt appreciation to our supervisor, Dr. Noureddine Amraoui, for his invaluable guidance, support, and patience. His expertise and encouragement have been instrumental in shaping this research and our academic growth.

We would also like to thank all the jury's members for accepting to review and judge this project, namely, Dr. Mohamed Kamel and Dr. Seddik Bougherara. The final version of this report has benefited from their careful reading and valuable comments.

We are also deeply thankful to the teachers from our faculty of mathematics and computer science for providing us with the necessary knowledge and skills to succeed in our studies.

Special thanks go to our friends, Nadjet B and Ahmed K, for their unwavering moral support and encouragement during challenging times.

Lastly, we offer our regards to everyone who has stood by us and supported us during this journey.

Table of contents

List of figures

List of tables

Introduction

People are continuously looking for services of different categories such as home and transportation services, as they seek convenience, efficiency, and quality in various aspects of their daily lives. Whether it is finding a reliable plumber to fix a leaky faucet, a ride-sharing service to get them to their destination, or specialized assistance for tasks ranging from pet care to event planning, the demand for diverse service options continues to grow in our increasingly interconnected and fast-paced world.

Problem and Objective

While many business and individual providers are making these services available, these latter still cannot be easily found by customers. In fact, people are still nowadays suffering from finding a construction man or hiring a plumber to fix something in their house.

To respond to such a problem, our work aims to bridge the gap between service providers and customers. On one hand, we aim to give the opportunity to services providers to offer their services to potential customers and showcase their skills and accumulate reviews and ratings. On the other hand, customers can effortlessly find providers, schedule appointments, negotiate prices, and manage their budgets.

Motivation

On the international stage, many online marketplaces for on-demand services have emerged in recent years including Thumbtack [29] and Urban Company [32]. These platforms connect users with local service providers, offering a convenient way to access a wide range of services, from home maintenance to event planning. Thumbtack, for instance, has revolutionized how customers find professionals, while Urban Company specializes in home services, making tasks like cleaning and repairs hassle-free. These platforms exemplify the growing trend of using technology to simplify service provision on a global scale.

In the Algerian scale, we can find many marketplace platforms for digital services (e.g., voice over, graphic design) including Freehali [9] and WorkVally[36]. We can also find other marketplaces for both products and services such as swiga[26]. However, they are not dedicated to local services, and people generally use them as a product marketplace such as Ouedkniss [18]. Consequentyl, the Algerian market is still lacking an online marketplace for on-demand local services.

Proposed Solution

We propose Mhenni, an online marketplace for on-demand services [15]. Our solution dedicated to the Algerian market revolutionizes service access and delivery, offering tailored features for both parties. Providers effortlessly manage profiles, showcase services, and suggest new categories. Feedback tools enable engagement with ratings and comments, while a support contact feature ensures assistance as needed. For customers, a secure profile management system enhances personalization. A robust review system facilitates informed decisions, with intuitive service discovery tools simplifying searches. Seamless payment processing and feedback management further enrich the user experience, making our platform versatile and inclusive.

Methodology

The development of Mhenni involved several key phases including research and analysis, design, and implementation. However, w did not followed a strict and linear sequence of phases but an incremental methodology instead [13]. In particular, a basic market research and user requirements analysis were initially conducted to determine the needs of potential users and identify critical features as well as studying similar international platforms. From this data, the system architecture and user interface were designed to prioritize scalability, security, and user-friendliness. Simultaneously, the database schema was designed using PostgreSQL for efficient data management, while Node.js with Express.js framework handled backend logic. Prisma was used to simplify database access, and Postman ensured API reliability through comprehensive testing. An admin web dashboard was prioritized to verify core business processes before developing the End-users Application, which was designed for a responsive user experience. The final stages involved thorough testing and adjustments to ensure stability and readiness for deployment.

Scope and Limitations

The current version of Mhenni presented in this report serves as a Proof of Concept to demonstrate the feasibility and functionality of such type of platforms. Upon full-scale launch, the platform will be available in Arabic to cater to the primary language of the target users in Algeria. Additionally, we plan to develop a mobile application to provide a more accessible and user-friendly experience, as mobile devices are more commonly used.

The primary features and functionalities of Mhenni that have been successfully implemented include:

- **Visitor Features**:
  - Search for services based on city and search queries.
  - Sort and filter services by various criteria such as price, last updated, and review ratings.
  - User registration and login using Email/Password or Google OAuth.
  - Option to sign up as a provider.
- **Provider Features**:
  - Profile management, including viewing and updating personal information (except email).
  - Creating, editing, and deleting services.
  - Managing service listings, including viewing details and archiving services.
  - Receiving notifications for service requests.
  - Accepting, rejecting, and confirming service requests.
  - Confirming the completion of services.
- **Customer Features**:
  - Profile management, including viewing and updating personal information (except email).
  - Managing service requests, including viewing current and past requests and their statuses.
  - Viewing service details before booking.
  - Requesting services and handling offers from providers.

  - Confirming and reviewing completed services.

- **Admin Features**:
  - Viewing statistics on service distribution by category.
  - Managing service categories.
  - Viewing lists of providers, customers, services, and requests.
  - Approving or rejecting service creations by providers.

Despite the comprehensive scope, several limitations were identified and need to be acknowledged:

- **Visitor Limitations**:
  - The functionality to describe needs in their own words using AI (e.g., ChatGPT) for language barrier removal is not implemented.
  - Filtering the list of services by scale (local, regional, and national) is not implemented.
- **Provider Limitations**:
  - Viewing statistics on revenues for tracking income and analyzing trends is not implemented.
- **Customer Limitations**:
  - Recommendation of new services based on historical needs is not implemented.
  - Sending private messages to providers for direct communication is not implemented.
- **Admin Limitations**:
  - Viewing statistics on revenues (e.g., providers' subscription fees, customers' premium accounts) is not implemented.
- **General Platform Limitations**:
  - Secure payment processing is not included in the current implementation.
  - Geographic expansion is initially limited to specific regions within Algeria, with further expansion requiring additional resources and localization efforts.

- Service category support is limited in the initial version, with future plans to extend to more diverse or niche services.
- Dependence on internet connectivity and digital literacy may limit accessibility in regions with poor infrastructure or lower technological adoption.

Report structure

This report is structured into three chapters. Chapter 1 covers the business analysis, examining the market needs and defining the project's objectives. Chapter 2 delves into the system design, detailing the architectural and technical specifications of the platform. Chapter 3 discusses the system implementation and deployment, outlining the development process, technologies used, and functionalities built. Finally, the report concludes with an analysis of the results and potential directions for future work.

Chapter 1

Business Analysis

In this chapter, we present the business analysis of this work which is necessary step to gather all the requirements for the project. In particular, before analyzing the system requirements, we discuss the the state of the art and understand the challenges and opportunities in connecting local service providers with customers in Algeria.

1.1 Background

On-demand services refer to a category of services that are provided to customers at their request, rather than on a scheduled or regular basis. These services leverage digital platforms to connect service providers with customers in real-time, offering convenience and flexibility. For Example ride-hailing services like Uber[30], food delivery services like DoorDash [7], and freelance platforms like TaskRabbit[28]. The key characteristic of on-demand services is their ability to fulfill customer needs quickly and efficiently, often through mobile apps or web-based platforms.

1.1.1 Services Categories

On-demand services encompass a wide range of categories, each addressing different customer needs. The main categories include but are not limited to:

Categories	Services
Home Services	House Cleaning/Housekeeping, Babysitting/Nanny Services, Plumbing, Electrical repairs, Carpentry/Handyman services, Painting/Wallpapering, Lawn Care/Gardening, Pest Control, Appliance Repair, Carpet Cleaning, Window Cleaning, Home Organization/Decluttering
Auto Services	Car Washing/Detailing, Oil Changes/Maintenance, Tire Services, Dent Removal, Windshield Repair/Replacement, Towing Services, Roadside Assistance
Moving/Delivery Services	Local Moving, Long-Distance Moving, Packing/Unpacking Services, Furniture Assembly, Courier/Delivery Services
Repair Services	Appliance Repair, Furniture Repair, Shoe Repair, Jewelry Repair, Bicycle Repair, Computer/Electronics Repair
Construction/Home Improvement	Carpentry/Framing, Masonry/Concrete Work, Roofing, Fencing, Deck Building/Repair, Drywall Installation/Repair, Flooring Installation/Refinishing, Bathroom Remodeling, Kitchen Remodeling, Landscaping/Hardscaping
Professional Services	Accounting/Tax Preparation, Legal Services, Tutoring/Academic Support, Web Design/Development, Graphic Design, Photography/Videography, Event Planning, Virtual Assistant Services
Personal Services	Hairstyling/Barbering, Makeup/Cosmetology, Personal Training/Fitness Instruction, Massage Therapy, Tailor/Alteration Services, Laundry/Dry Cleaning, Pet Grooming/Walking
Health/Wellness	In-Home Care Services, Nursing Services, Counseling/Therapy, Nutrition/Dietary Services, Yoga/Pilates Instruction
Entertainment/Activities	DJ Services, Catering, Party/Event Rentals, Local Tours/Experiences, Performing Arts/Music Lessons

Table 1.1 Categories and Services

1.1.2 Terminology

On-demands services marketplaces is a topic that involves many key-terms that should be clearly defined:

- **Service**: A functionality or offering provided through the platform (e.g., plumbing).
- **Visitor**: A person that navigates through the platform without a personal account.

- **Customer**: A person having a personal account that consumes a service.
- **Provider**: A person having a personal account that provides a service.
- **Request**: A request made by the customer for a specific service from a provider.
- **Offer**: A response from the provider to the customer's service request specifying the availability date and time.
- **Appointment**: An agreement between the provider and customer after the latter has accepted the offer of the provider.
- **Review**: A textual comment and an out-of-five points rating submitted by the customer.

1.1.3 Service Life-cycle

The service life-cycle in an on-demand platform typically involves several states with transitions, as illustrated in Figure 1.1

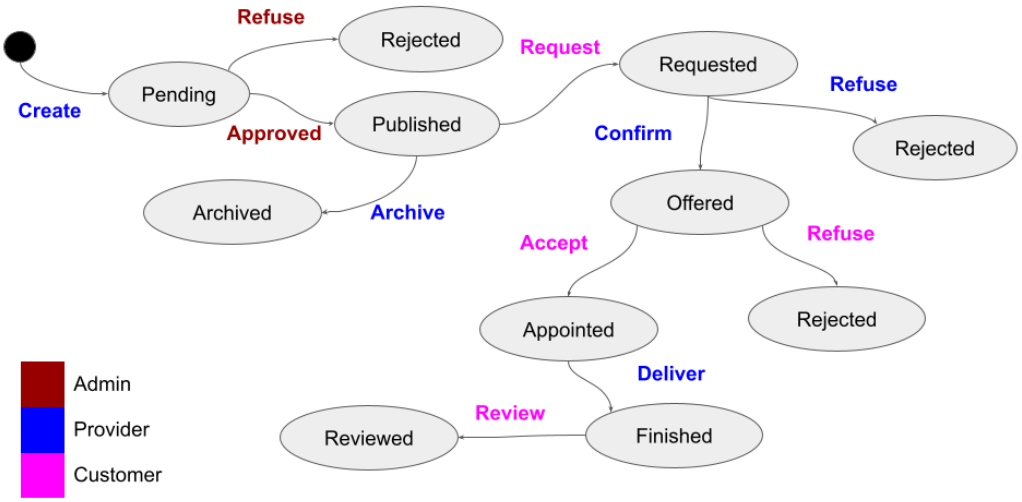

Fig. 1.1 Illustration of the Service Life-cycle

- **Pending**: Once a service is created by a provider, it should be pending, waiting for the platform administrator's approval.
- **Creation Rejected**: If the admin refuses the created service, it becomes rejected.

- **Published**: If the admin accepts the created service, it becomes published and visible to customers.
- **Archived**: If the provider decides to archive the service, it becomes invisible to customers.
- **Requested**: Once a service is requested by a customer, it becomes requested, waiting for the response of the provider.
- **Rejected**: If the provider refuses to deliver the service, it becomes rejected.
- **Offered**: If the provider accepts to deliver the service, they provide an offer, and thus the service becomes offered.
- **Rejected**: If the customer refuses the offer, it becomes rejected.
- **Appointed**: If the customer accepts the offer, it becomes appointed.
- **Finished**: Once the provider delivers the appointed service to the customer, it becomes finished.
- **Reviewed**: If the customer decides to leave a review, it becomes reviewed.

1.2 State of The Art

In our busy world today, people are always looking for help with various tasks, whether it is fixing something at home or getting a ride somewhere. However, finding the right person for the job can be really hard. You might ask friends or search online, but it takes time and sometimes you still cannot find the right person. This causes frustration for both the person needing help and the one offering it. Consequently, there is still a gap between people who need services and those who offer them.

To bridge the gap between service providers and customers, there have been many platforms developed at the international scale such as TaskRabbit [28] or Thumbtack [29]. These platforms help in connecting people who need services with those who can provide them, but they are only dedicated to the American market.

In Algeria, we can find some platforms such as Freehali [9] and WorkVally[36], but they are only dedicated for digital services and do not cover all the tasks people need help with. Even marketplace apps such as Ouedkniss [18] and Swiga [26] more for buying and selling stuff, not specifically for finding local services.

So, while these platforms give us some ideas, there is still room to create something better tailored to Algerians, making it easier for people to find and hire local service providers.

1.3 Business Model

Our proposed solution is an online marketplace for on-demand services tailored to the Algerian market, where users can hire professionals for various tasks, such as home maintenance, tutoring, event planning, and more.

1.3.1 Customer Segments

The target customers of our online marketplace include:

Service Seekers

- **Individuals**: Consumers who require various services for their personal needs, such as home cleaning, tutoring, pet care, etc.
- **Businesses**: Organizations seeking services to support their operations, such as office maintenance, IT support, event planning, etc.

Service Providers

- **Freelancers**: Independent professionals offering specialized services in various fields.
- **Small Businesses**: Local businesses providing services such as home repair, beauty services, landscaping, etc.
- **Enterprises**: Larger companies offering professional services or specialized solutions in specific industries.

Additionally, specific groups that can benefit from our platform include busy professionals, seniors, families, small business owners, individuals with disabilities, students, event planners, homeowners, and commuters. These groups often have diverse needs and are likely to benefit from the convenience and efficiency of our platform.

1.3.2 Problems Addressed

Our platform addresses several significant issues:

- Inefficiency and inconvenience in finding and booking service providers.

- Lack of reliable information about service providers.
- Difficulty in scheduling appointments that fit busy schedules.

1.3.3 Solutions Provided

To solve these problems, our platform offers:

- A centralized platform connecting service providers with customers.
- A wide range of service providers with transparent information.
- A reviews and ratings system to guide decision-making.
- Easy booking and scheduling options.

1.3.4 Benefits Created

The benefits of our platform include:

- **Convenience**: Offering a one-stop platform for both service providers and customers to find, book, and manage services conveniently from anywhere with an internet connection.
- **Efficiency**: Streamlining the process of connecting service providers with customers, reducing the time and effort required to search for, schedule, and fulfill service requests.
- **Choice**: Providing customers with a wide selection of service providers and services to choose from, allowing them to find the best fit for their needs and preferences.
- **Trust**: Building trust between service providers and customers through transparent information, reviews, and ratings, fostering long-term relationships and repeat business.
- **Revenue Opportunities**: Creating opportunities for service providers to expand their customer base, increase their visibility, and grow their businesses through the platform.
- **Community Building**: Facilitating connections between service providers and customers, fostering a sense of community and collaboration within the marketplace ecosystem.

1.4 Requirements Analysis

Requirement analysis is a critical phase in our system development life cycle that involves determining the needs and conditions to meet for a new or altered product [24]. This process is fundamental for ensuring that our project meets its goals and delivers value. The primary objective of requirement analysis is to identify, document, and manage the requirements of our system developed.

1.4.1 User Identification

We categorized our potential user base into four types:

- **Visitors**: Visitors are individuals who access the platform without initially registering or committing to any specific role as a service provider or customer. They may be prospective users exploring the platform to understand its offerings and functionality.
- **Providers**: These are individuals or businesses offering various services. They could be freelancers, professionals, or companies specializing in different fields such as home maintenance, tutoring, event planning, and more.
- **Customers**: These are individuals or businesses seeking specific services. Customers will use the platform to search for services they need, browse through available service providers, compare options, schedule appointments, communicate with service providers, and make payments.
- **Admin**: The Admin role is responsible for managing the platform, overseeing user activity, and ensuring the functionality. Admins typically have access to tools and dashboards to moderate content, manage user accounts, resolve disputes, and perform other administrative tasks.

1.4.2 User Stories

A user story describes functionality that will be valuable to either a user or purchaser of a system or software [5]. Unlike traditional requirements, user stories are concise, informal descriptions of features told from the perspective of the end user. They focus on *what* the user needs and *why*, rather than how to implement those needs.

In the following, we describe the stories of our identified users, i.e., visitors, customers, provider, and the admin.

As a Visitor, I want to:

- Search for services and get a list of suggested services based on my city and search query so that I can see the available services in the selected area.
- Describe my needs in my own words instead of choosing from a predefined list of key words.
- Sort the list of services by different options (e.g., price, last updated, review average point, etc.) so I an compare and find the best options based on my priorities.
- Filter the list of services by scale (i.e., local, regional, and national) so I can see the available services in the chosen area.
- Sign up or login (Email/Password or Google OAuth) for easy access to features.
- If I decide to provide a service, I can sign up for a provider account.
- Receive automatic "Continue as" Google popup for easier login/sign-up.

As a Provider, I want to:

- View/update my profile so I have always have the ability to view my information and update it in case it is empty or needs to be changed.
- Create/edit/delete a service so that I can manage my service offerings efficiently, including creating new services, editing existing ones to update information or pricing, and deleting outdated services to maintain accurate and relevant listings.
- View the list of my services including active and archived ones so that I can manage my offerings effectively, track current services, and access archived ones for historical reference.
- View the details of each service so that I can access important information such as service description, pricing, availability, and any other relevant details. This helps me accurately communicate information to customers, manage service delivery, and make informed decisions regarding my offerings.
- Get notified when my service is requested so that I can promptly respond to customer inquiries and ensure timely communication.
- Reject a request for my service so that I can decline requests if I am unable to fulfill them due to scheduling conflicts, capacity constraints, or other reasons.

- Confirm and offer the service (e.g., set availability date and time, set estimated required time, set estimated price) so that I can set clear expectations with the customer by confirming availability, estimated time, and pricing.
- Confirm the accomplishment of the service so that I can verify that the service has been successfully completed as agreed upon.
- Archive my service so it will not be shown in search results and I can ensure it is no longer displayed when the service is no longer available or relevant.
- View statistics on my revenues so that I can track my income, analyze trends, and make informed decisions to optimize my business strategies.

As a Customer, I want to:

- View/update my profile so that I always have the ability to view my information and update it in case it is empty or needs to be changed.
- View the list of my service requests, including current and previous ones, and show the request status (i.e., pending, offered, rejected, finished) so that I have visibility and control over my service interactions. This facilitates better communication, accountability, and overall customer experience on the platform.
- View the details of each service so that I can see more information about the service such as description, price, and the provider who offers the service before deciding to book it.
- Request a service so that I can create a request to the provider who offers the selected service. The provider needs to offer an available date for me to choose.
- Accept or reject an offer of service so that if I am satisfied with a the date and time offered by the provider, I can choose that date and time and accept the offer. At this point, the request will be confirmed by both me and the provider.
- Confirm and post a review on a service after it is finished so that I can rate the provider by giving them a comment and a star review.
- Be recommended with new needs based on my historical ones so that I can discover relevant services that align closely with my past preferences and requirements, saving time and enhancing my overall experience by presenting options that are likely to meet my needs.

- Send private messages to the provider so that I can have direct communication to discuss specific details, negotiate terms, and clarify any questions or concerns I may have before committing to their services.
- Get customer support so that I can receive assistance, resolve issues, and get answers to questions regarding the services or products I'm interested in or have already purchased.

As an Admin, I want to:

- View statistics (e.g., services ratios by category) so that I can understand service distribution, identify trends, and allocate resources effectively.
- Create/edit categories so that I can organize and classify services effectively, improving searchability and user experience.
- View the list of providers/customers/services/requests so that I can manage and monitor the platform effectively, ensuring smooth operation and addressing any issues that may arise.
- Approve/reject a service creation by a provider so that I can maintain quality control and ensure that only appropriate and relevant services are offered on the platform.
- View statistics of revenues (e.g., providers' subscription fees, customers' premium accounts, etc.) so that I can track financial performance, analyze revenue sources, and make informed decisions to optimize business strategies.

1.4.3 Non-functional Requirements

Non-functional Requirements (NFRs) are system qualities that guide the design of the solution and often serve as constraints across the relevant backlogs [5]. Unlike functional requirements, which specify how a system responds to specific inputs, non-functional requirements are used to specify various system qualities and attributes. The target non-functional requirements for an efficient online marketplace include:

- **Performance**: Ensuring the system responds quickly to requests and transactions, meeting defined performance criteria.
- **Scalability**: How well the system can handle an increase in users, data, or transactions without impacting performance or availability.

- **Security**: Ensuring the system protects against unauthorized access, data breaches, and other security threats.
- **Usability**: How easy the system is to use, including accessibility for users with disabilities and intuitive user interfaces.
- **Maintainability**: How easy it is to update, modify, and maintain the system over time.

1.5 Conclusion

In conclusion, while platforms like TaskRabbit and Thumbtack provide valuable insights, there is a clear opportunity to develop a more tailored solution for the Algerian market. Our platform aims to bridge the provider-customer gap, offering a seamless, efficient, and reliable way for users to find and hire local service providers. This not only addresses the unique challenges faced by Algerians but also leverages the existing gaps in the market to create a comprehensive and user-friendly service marketplace.

Chapter 2

System Design

In this chapter, we provide a comprehensive explanation of our system design. We begin by outlining the overall architecture of the system, detailing its key components and their interactions. Following this, we delve into the specifics of each major component, namely the Back-end Server, the Front-end Application, and the Admin Dashboard. Each section will cover the individual elements, data models, API endpoints, and user interfaces associated with these components, providing a thorough understanding of how the system operates as a cohesive whole.

We have designed the architectural diagrams following the C4 model [2], whereas, to design system behavior and processes, we have used UML [31].

2.1 System Architecture

In this section, we present the architecture of our proposed system using two different perspectives: Context and Containers.

2.1.1 Context

Figure 2.1 shows the system context of Mhenni. In particular, it depicts the various actors involved and their interactions with the central Mhenni system, which serves as the platform connecting customers with service providers. The key actors are Visitors, who can discover services and browse content without an account; Customers, who have personal accounts to request services; Providers, with personal accounts to create, manage, and offer services; and the Admin, responsible for managing and overseeing the operations of the marketplace.

Mhenni also asks the help of third-party services for performing users authentication and authorization via Auth0 as well as storing and retrieving images via Cloudinary.

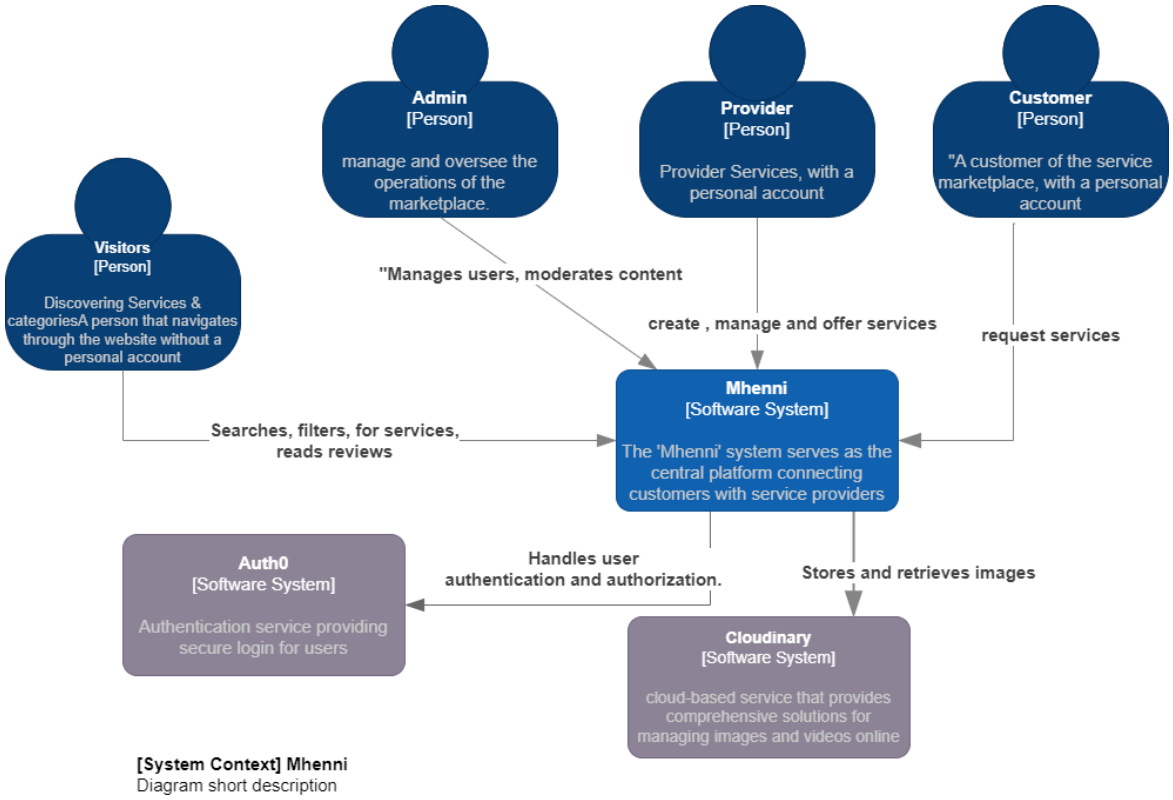

Fig. 2.1 System Context

2.1.2 Containers

Figure 2.2 zooms-in to the system boundary and depicts the containerized architecture of the Mhenni system. It shows the various components and their interactions, including the end-users application that handles the requests coming from visitors, customers, and providers; the admin dashboard that handles the requests coming from the platform administrator; the back-end server that handles API requests and business logic; and the database management system for storing user information, service details, and reviews.

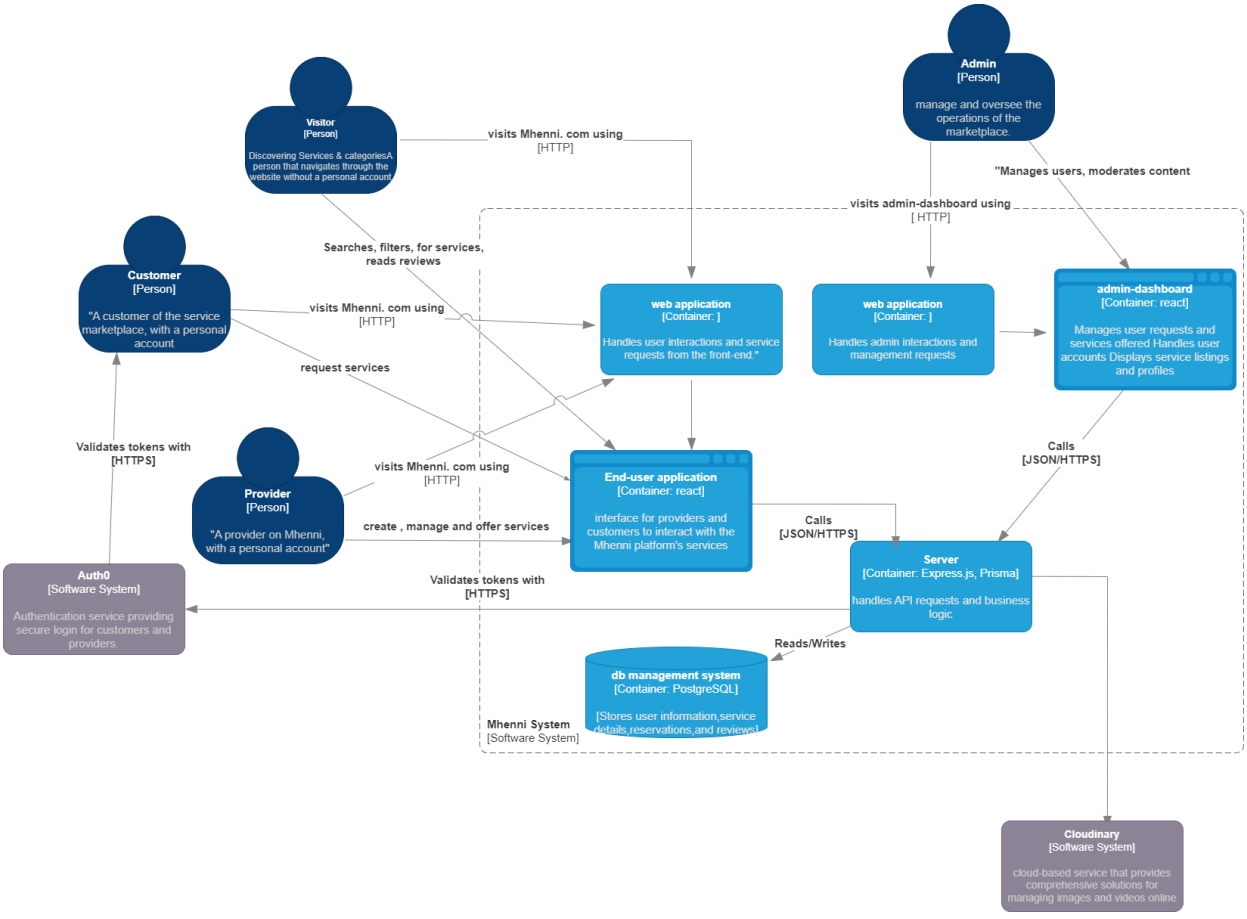

Fig. 2.2 System Containers

2.2 Back-end Server

The back-end server is the application that processes clients requests (i.e., end-users and admin dashboard application) and communicates with the database server to fetch and store data.

Figure 2.3 shows the components of our back-end server. The Router component that handles incoming HTTP requests and directs them to the appropriate controllers; the middleware component that manages requests validation, authentication, and other pre-processing tasks; the controller component that handles actual the business logic for handling requests related to users, services, bookings, and reviews; the Object Relational Mapping (ORM) client component that executes database queries and returns search results on behalf the controller.

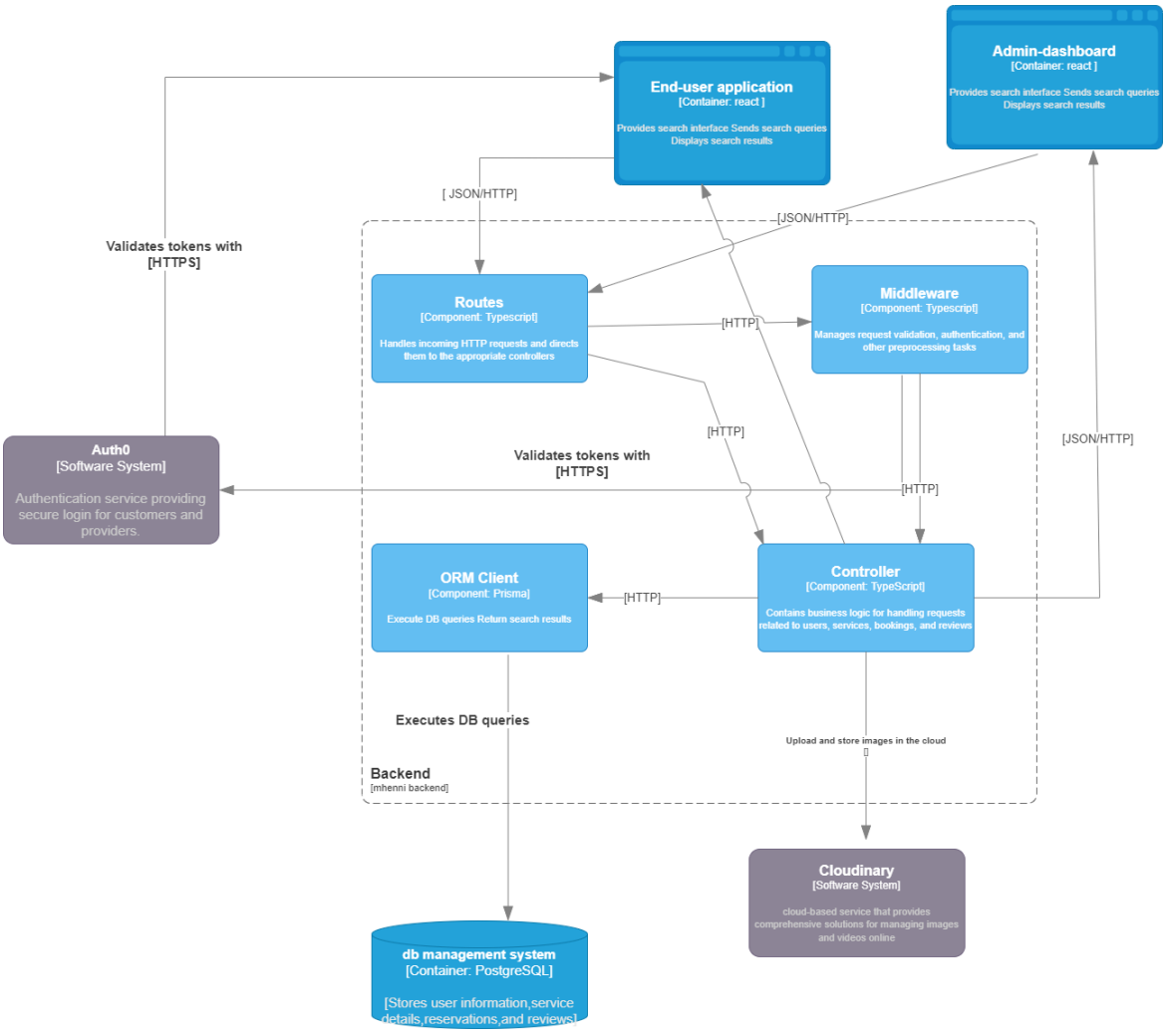

Fig. 2.3 Back-end Server Components

2.2.1 Data Model

The figure 2.4 shows the entity-relationship (ER) diagram depicting the data model of the system. It illustrates the various entities, such as User, Customer, Provider, Service, Category, Request, Address, and their relationships. The diagram also includes details about the attributes and data types associated with each entity. This ER diagram provides a comprehensive view of how the data is structured and organized within the system's database.

The three main entities of our data model are: The User which can be a Customer, a Provider, or an Admin. The Service, and The Request.

The User table captures common details between customers, providers, and the Admin. This table includes the following attributes but is not limited to:

- `id` – A unique number assigned to each user by the server. Also the primary key of this table.
- `auth0Id` - A unique number assigned to each user by Auth0 service.
- `firstName` – The first name of the user.
- `lastName` – The last name of the user.
- `password` – The user's password encrypted using JWT.
- `mobile` – The user's mobile number.
- `image` – The user's profile image.
- `type` – The type of user, which can be Admin, Provider, or Customer.
- `admin` – A relation to the Admin entity, if the user is an admin.
- `customer` – A relation to the Customer entity, if the user is a customer.
- `provider` – A relation to the Provider entity, if the user is a provider.

The Service entity represents the services offered within the system. It includes the following attributes:

- `id` – A unique identifier for the service.
- `service_name` – The name of the service.
- `service_category_id` – The ID of the category to which this service belongs.

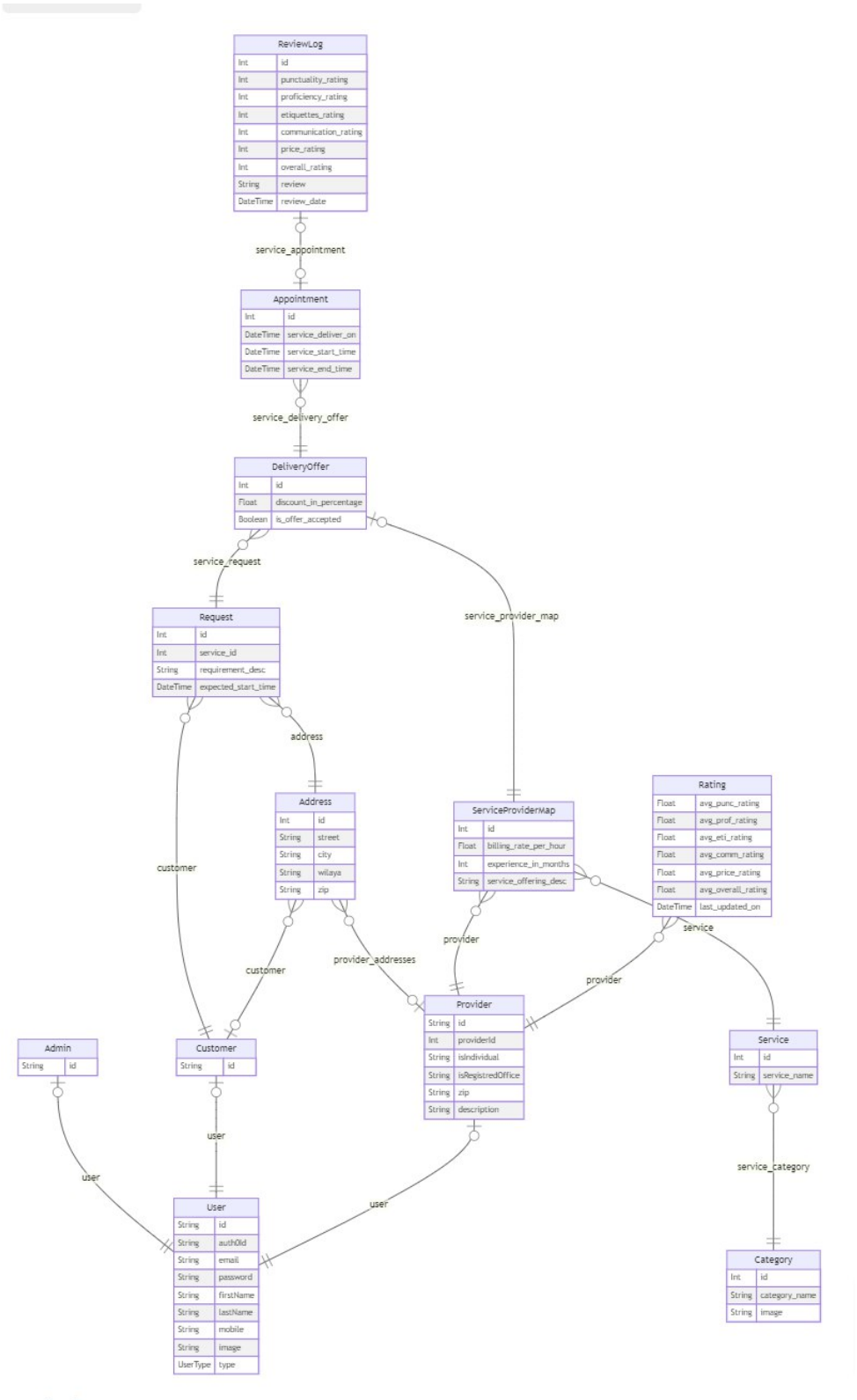

Fig. 2.4 Entity-Relationship Diagram

The Request entity represents a service request made by a customer. It includes the following attributes:

- `id` – A unique identifier for the request.
- `customer_id` – The ID of the customer making the request.
- `service_id` – The ID of the service being requested.
- `requirement_desc` – Description of the specific requirements for the service.
- `expected_start_time` – The expected start time for the service.
- `addressId` – The ID of the address where the service is requested.
- `custom_address_city` – The city of the custom address if provided.
- `custom_address_street` – The street of the custom address if provided.
- `custom_address_wilaya` – The wilaya (province) of the custom address if provided.
- `custom_address_zip` – The ZIP code of the custom address if provided.
- `providerId` – The ID of the provider assigned to this request.
- `state` – The current state of the request (e.g., requested, offered, ongoing, completed).

2.2.2 Router

The back-end server is exposed to clients through a set of public and protected API endpoints. The router module is responsible for handling HTTP requests coming to different API endpoints to determine which controllers and middleware should respond to various actions and inputs from clients.

In the following, we present different API endpoints categorized by public, customer, provider, and admin endpoints.

Public Endpoints

The following endpoints are publicly available for any visitor of the application:

- Category Route: `/api/public/category`

Method	Endpoint	Result
GET	/list	Get the list of categories
GET	/:categoryId	Get a specific category with its ID

Table 2.1 Category Public API Endpoints

- Service Route: `/api/public/service`

Method	Endpoint	Result
GET	/list	Get the list of services
GET	/:id	Get a specific service with its ID
GET	/category/:categoryId	Get services by category ID

Table 2.2 Service Public API Endpoints

Users Endpoints

The following endpoints are only available for authenticated provider and customer users:

- User Route: `/api/my/user` (Logged in user)

Method	Endpoint	Result
POST	/	Create a new user (email, auth0Id)
GET	/	Get the current user
PUT	/	Update the current user
PUT	/image	Update user profile picture
PUT	/type	Update user type

Table 2.3 User Protected API Endpoints

- Provider Route: `/api/my/provider` (Logged in provider)

Method	Endpoint	Result
GET	/	Get the current provider
POST	/add-service	Add a service for the provider
GET	/requests	Get provider requests
GET	/services	Get provider services
PUT	/services/:id	Update a provider service
DELETE	/services/:id	Delete a provider service
POST	/requests/:requestId/offer	Create a provider date and time offer for a request
POST	/requests/:requestId/complete	Confirm request completion

Table 2.4 Provider Protected API Endpoints

Admin Endpoints

The following endpoints are only available for an authenticated admin user:

- Authentication Route: `/api/admin/auth`

Method	Endpoint	Result
POST	/register	Create a new user (email, password, type)
POST	/login	Login
GET	/refresh	Refresh the token
POST	/logout	Logout

Table 2.5 Authentication Protected API Endpoints

- Provider Route: `/api/admin/provider`

Method	Endpoint	Result
GET	/list	Get the list of providers
GET	/:providerId	Get a specific provider with its ID
PUT	/:providerId	Update a specific provider
DELETE	/:providerId	Delete a specific provider
GET	/ratings/:providerId	Get the list of ratings of a specific provider

Table 2.6 Provider Protected API Endpoints

- Category Route: `/api/admin/category`

Method	Endpoint	Result
GET	/list	Get the list of service categories
GET	/:categoryId	Get a specific category with its ID
POST	/	Create a new category
PUT	/:categoryId	Update a specific category
DELETE	/:categoryId	Delete a specific category (will delete all services included in the category)

Table 2.7 Category Protected API Endpoints

- Service Route: `/api/admin/service`

Method	Endpoint	Result
GET	/list	Get the list of all services of all providers (with category included)
GET	/list/:providerId	Get the list of services of a specific provider (with category included)
GET	/:serviceId	Get a specific service with its ID (with category included)
PUT	/:serviceId	Update a specific service (including the category)
DELETE	/:serviceId	Delete a specific service
GET	/ratios	Get the ratios of categories by number of services they include

Table 2.8 Service Protected API Endpoints

- Customer Route: `/api/admin/customer`

Method	Endpoint	Result
GET	/list	Get the list of customers
GET	/:customerId	Get a specific customer with its ID
PUT	/:customerId	Update a specific customer
DELETE	/:customerId	Delete a specific customer

Table 2.9 Customer Protected API Endpoints

- Request Route: `/api/admin/request`

Method	Endpoint	Result
GET	/list	Get the list of service requests made by all customers

Table 2.10 Request Protected API Endpoints

2.2.3 Controller

Once is triggered by the router, the controller module is responsible for handling the actual business logic by interpreting requests, processing data, and returning the appropriate HTTP responses accordingly (e.g., fetching the list of services with 200 status).

The interaction between the controller and the database is done through the ORM client.

2.2.4 Middle-ware

Middleware is a set of functions to handle requests and responses in the request-response cycle. They have access to the request and response objects and can execute code, modify these objects, end the request-response cycle. Middlewares provide a flexible way to add reusable functionality to an application's request processing pipeline in frameworks like Express.js.

The middleware functions of our system include:

- JWT Checker: that verifies the validation of the recevied JWT token along the HTTP request.
- JWT Parser: that decodes JWT token and extracts the user ID.
- Data Validator: that verifies if required data is provided along the request, e.g., customer profile update information.
- Body Parser: to parse request body content such as multipart/form-data when a file is uploaded (e.g., customer profile image update).

2.2.5 ORM Client

An Object-Relational Mapping (ORM) client is a library or framework that simplifies database interactions in object-oriented programming languages. It maps database tables and their relationships to objects and provides an API to perform CRUD (Create, Read, Update, Delete)

operations on these objects. ORM clients bridge the gap between relational databases, which store data in tables, and object-oriented programming languages, which represent data as objects. They automate the translation between objects in code and the corresponding rows in the database, abstracting away the need for manual SQL queries in many cases.

2.3 End-users Application

The end-users application is the primary interface for visitors, customers, and providers to interact with the system. In this section, we describe the various pages and functionalities available to each user type.

2.3.1 Pages

Table 2.11 lists the public pages accessible to all visitors of the application.

Route	Description
/	Home page
/about	About us page
/contact	Contact page
/search	Search page
/service-details/:servceid	Service details page

Table 2.11 End-users Application Public Pages

Table 2.12 details the pages available to authenticated users (both providers and customers).

Route	Description
/choose-type	Choose user type
/complete-registration	Complete registration
/profile	View/update profile information

Table 2.12 End-users Application Users Pages

Table 2.13 outlines the pages available to authenticated providers.

Route	Description
/manage-services/	Add and Manage provided services
/manage-requests-provider/	View the list of the service requests

Table 2.13 End-users Application Providers Pages

Table 2.14 details the pages available to authenticated customers.

Route	Description
/manage-requests-customer	View the list of the service requests

Table 2.14 End-users Application Customers Pages

2.3.2 Use Cases

In this section, we outline the use cases relevant to the end-users application. These use cases describe the interactions of visitors, customers, and providers with the application, highlighting the functionalities available to each type of user.

Figure 2.5 illustrates a visitor's use case diagram for a front-end application.

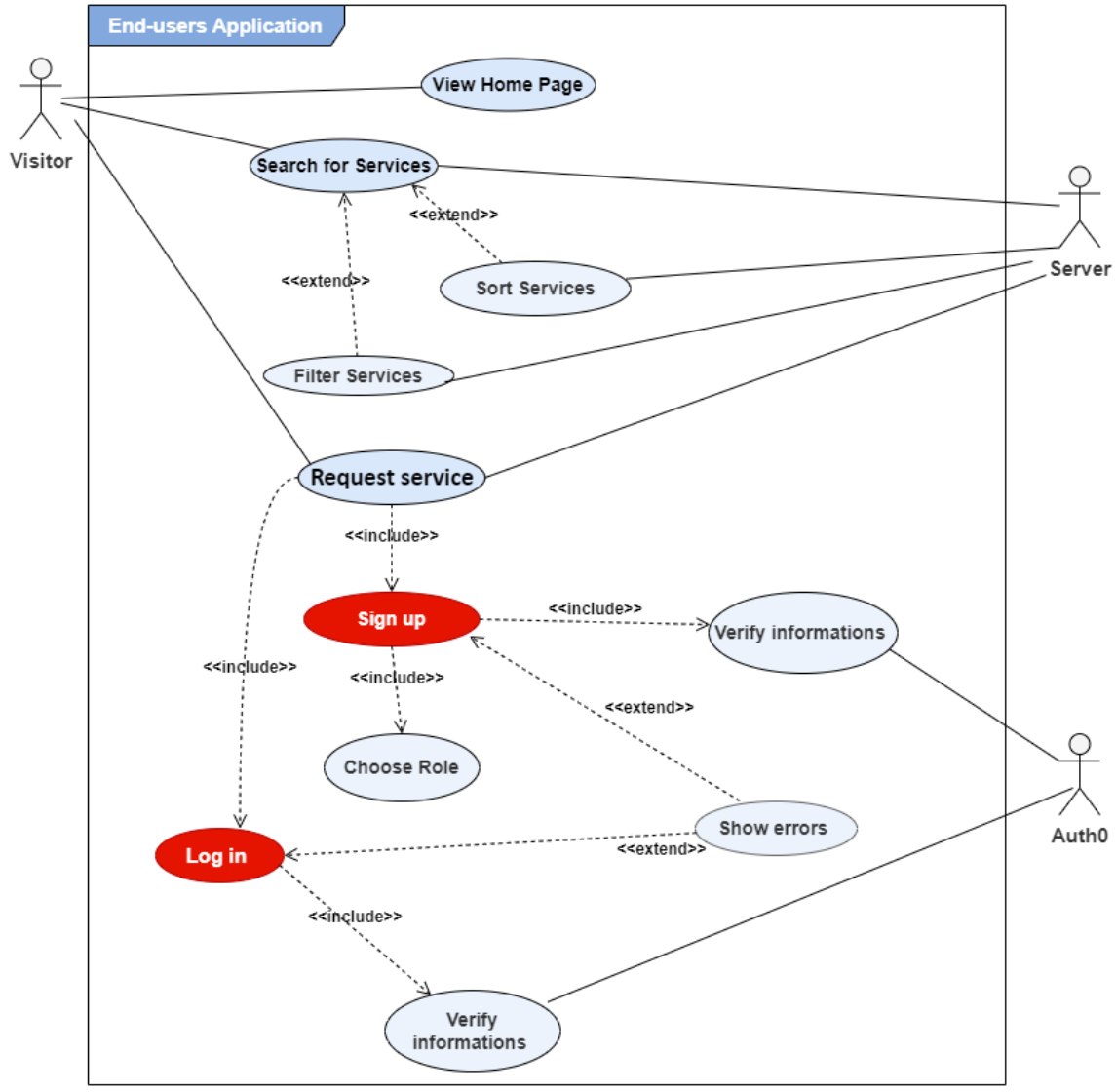

Fig. 2.5 Visitor Use Case Diagram

Figure 2.6 illustrates a use case scenario for a customer to request and manage services.

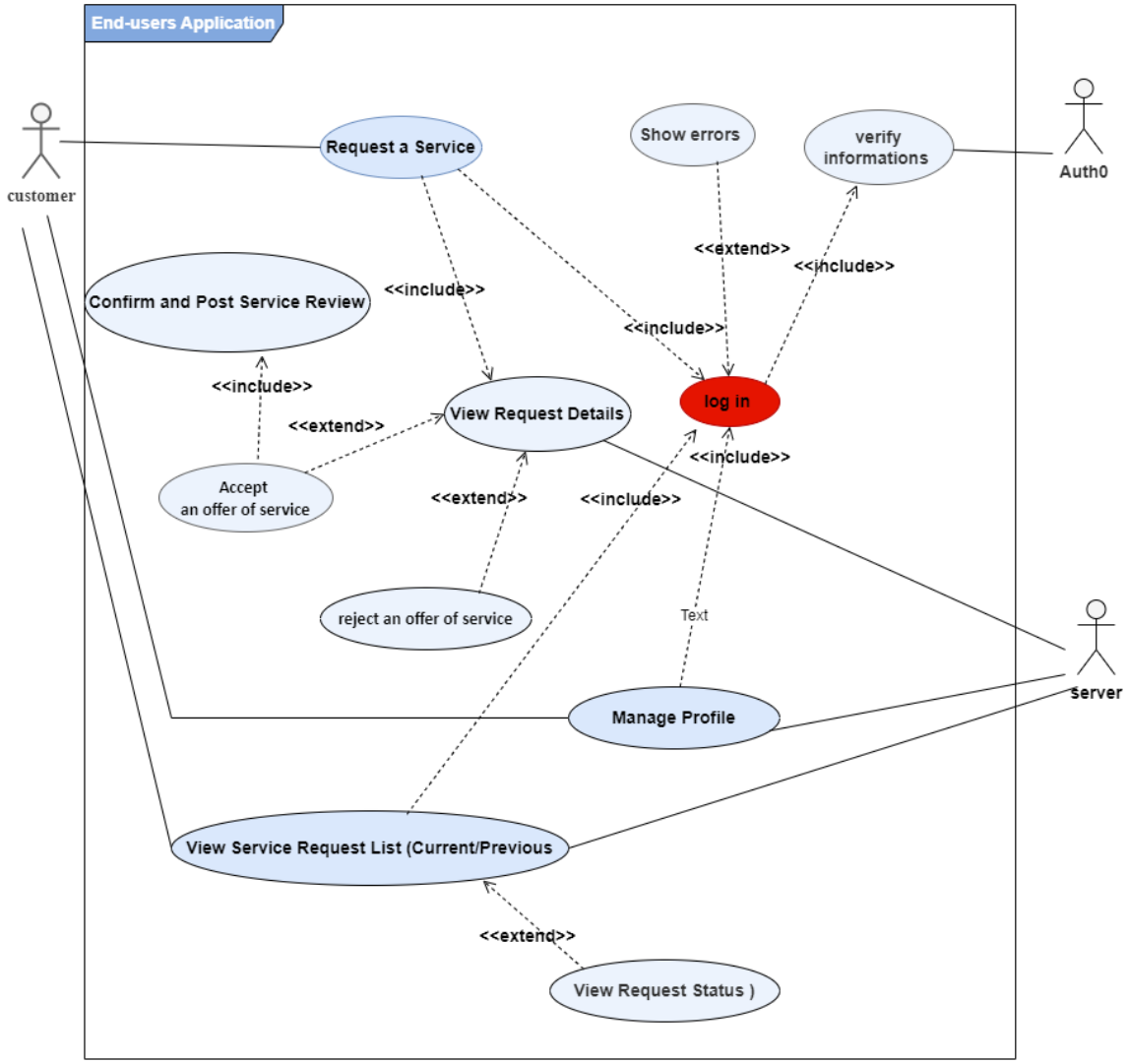

Fig. 2.6 Customer Use Case Diagram

Figure 2.7 illustrates the use case scenario for a provider actor interacting with a front-end application.

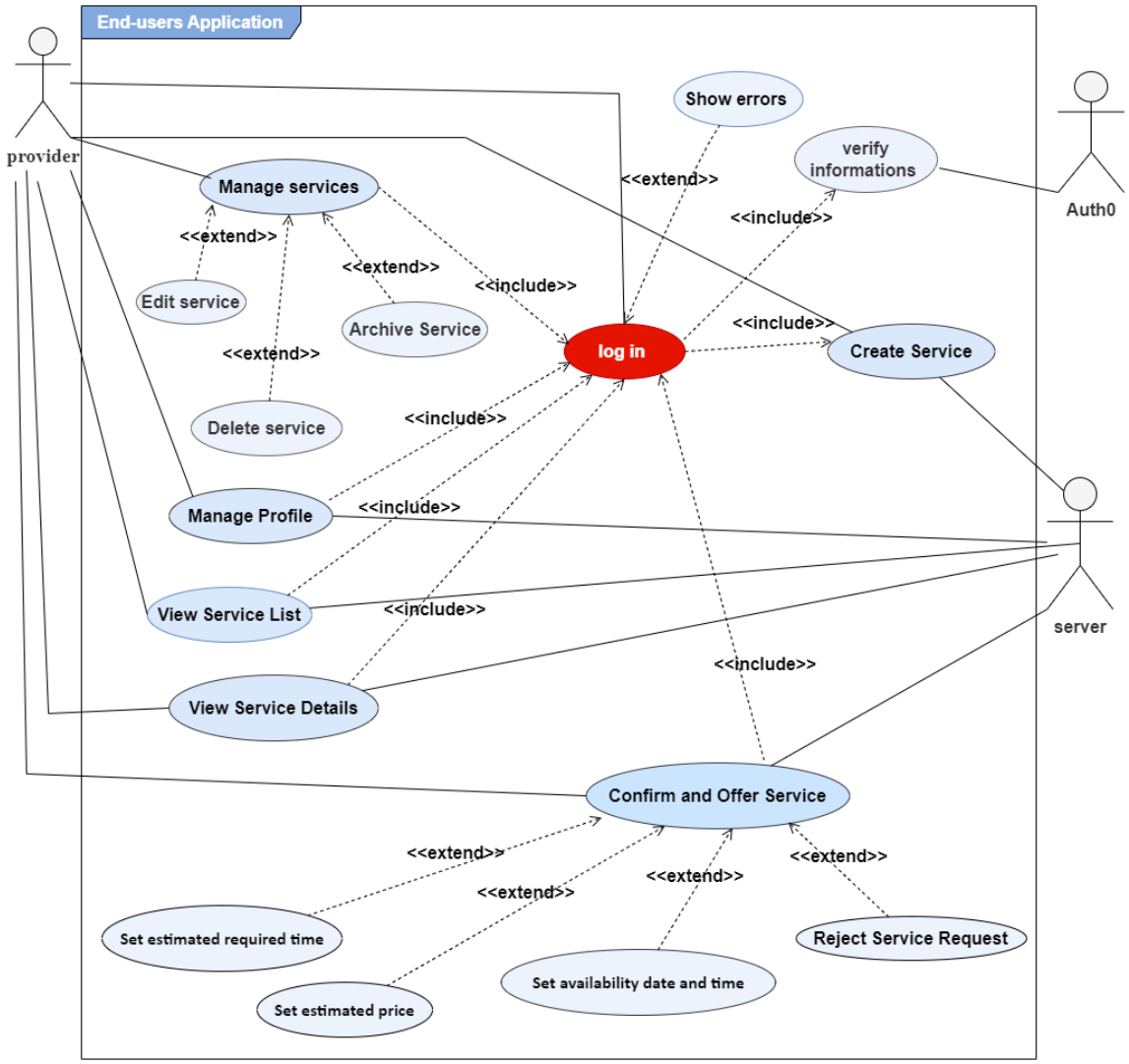

Fig. 2.7 Provider Use Case Diagram

2.3.3 Flow Charts

The flow charts in this section depict the processes and workflows within the end-users application. These charts help visualize the steps involved in common user actions, providing a clear understanding of the application's behavior.

Signing Up and Publishing a Service

Figure 2.8 outlines the process for a provider to sign up and create and publish a new service on the platform

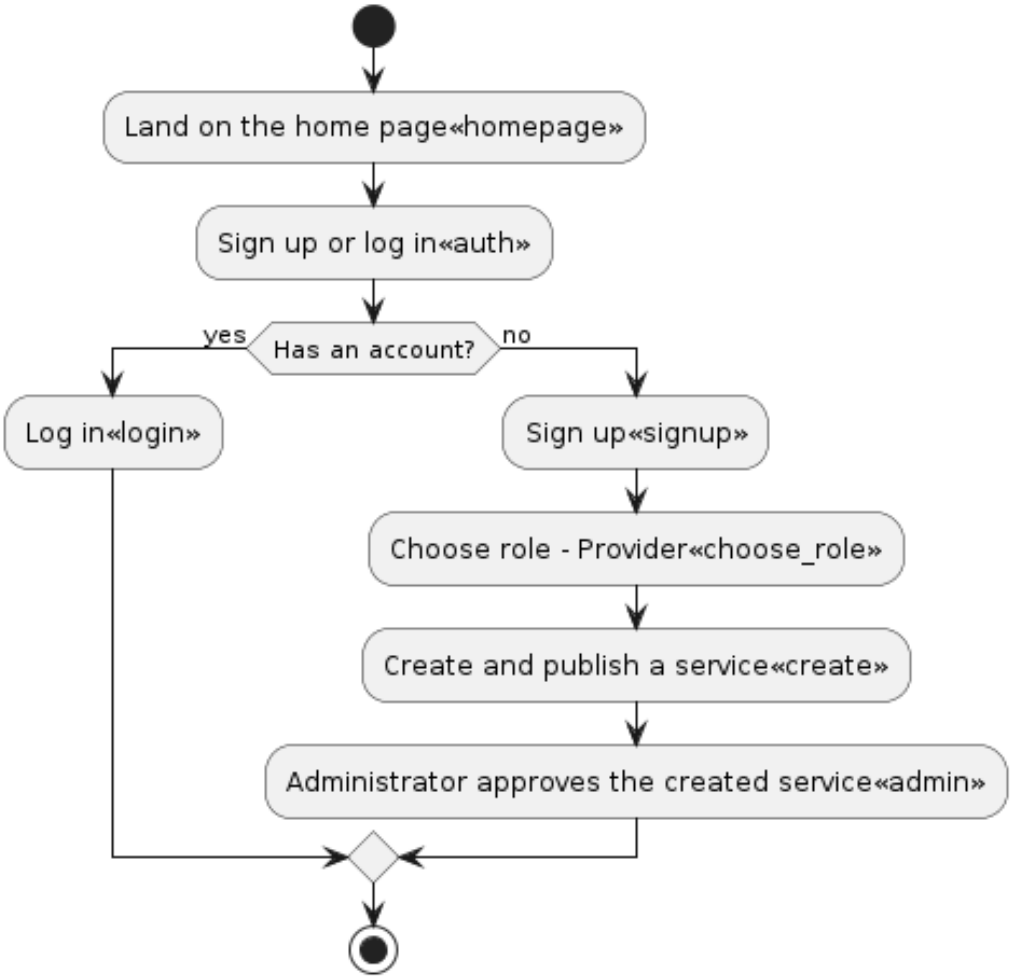

Fig. 2.8 Signing Up and Publishing a Service

Searching and Requesting a Service

Figure 2.9 depicts the flow for a user requesting a service on the platform

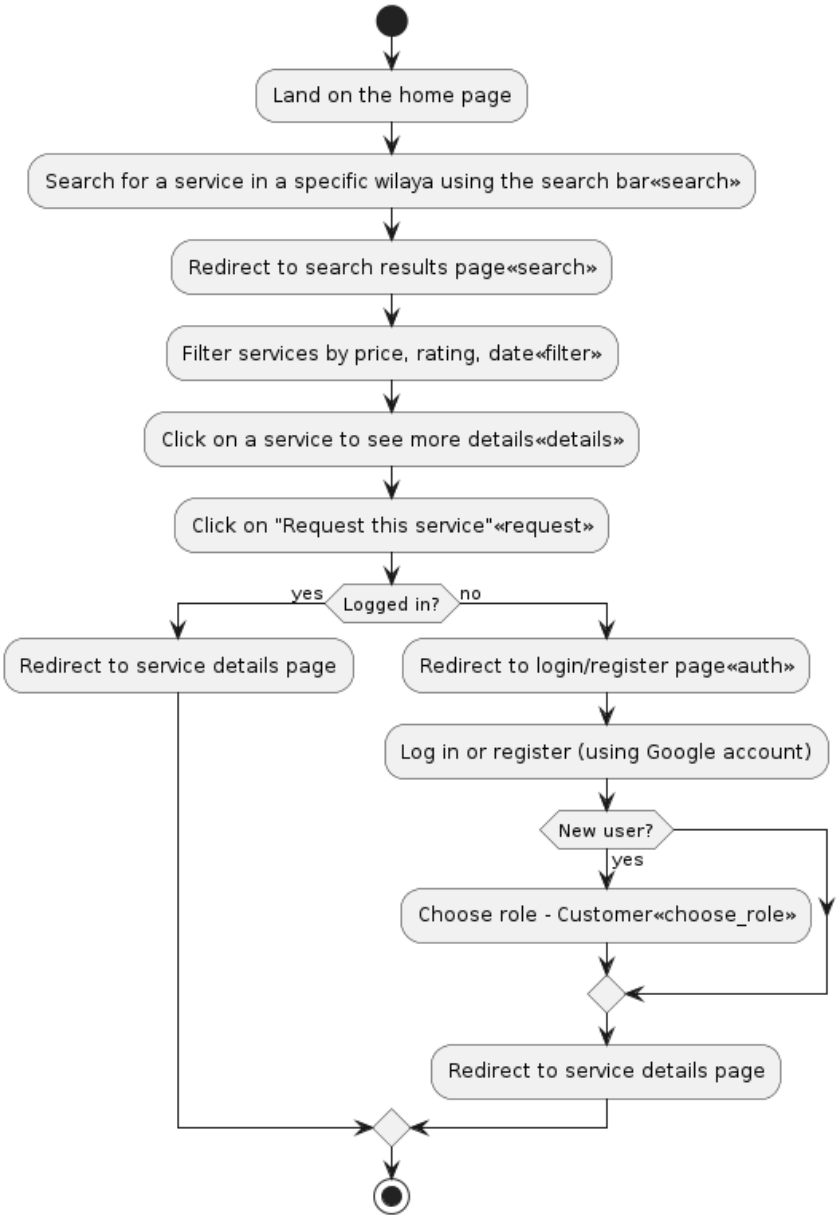

Fig. 2.9 Searching and Requesting a Service

Approval, Appointment, and Completion Process

Figure 2.10 outlines the process in which the Customer requests and schedules a service from a provider.

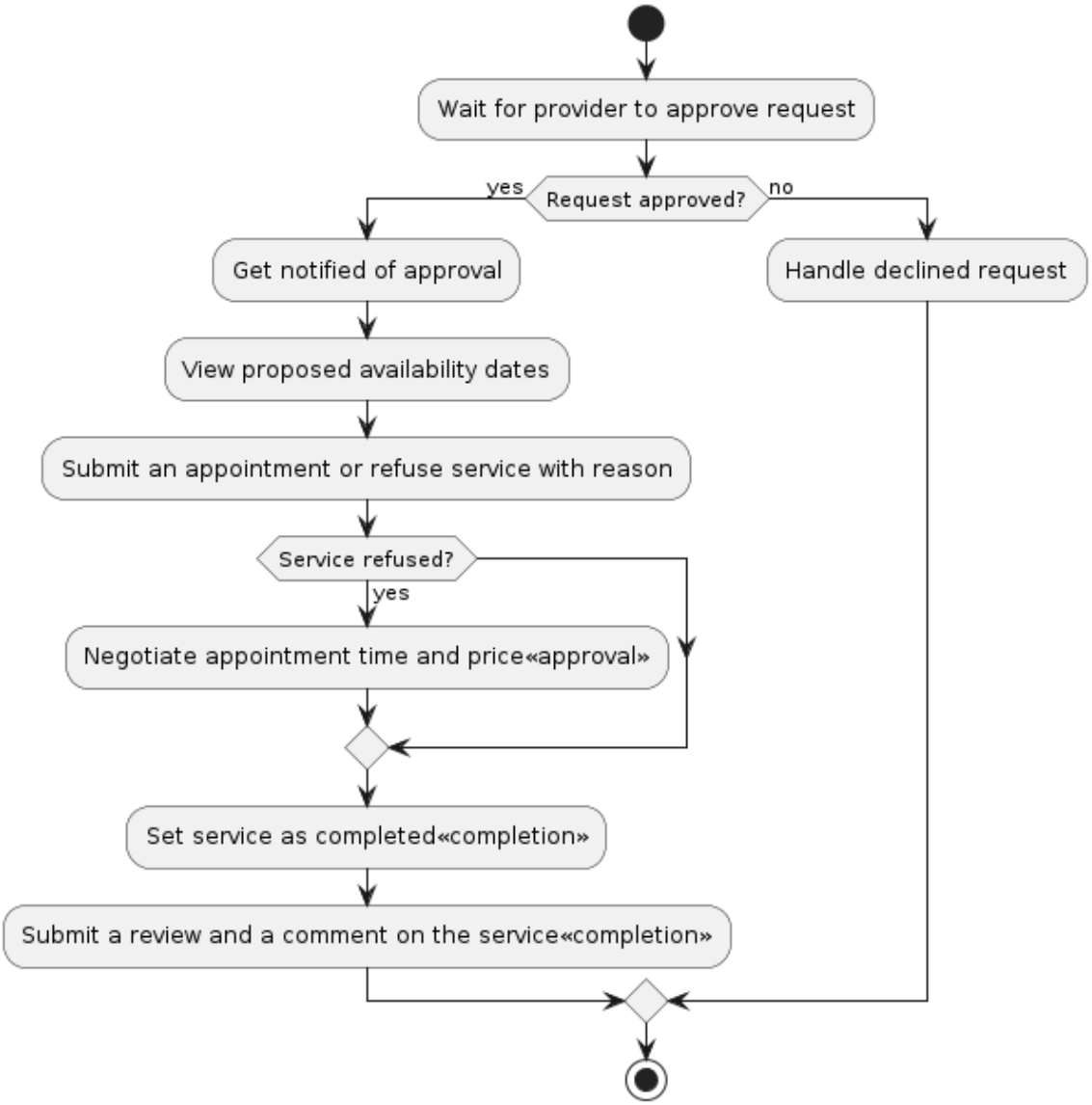

Fig. 2.10 Approval, Appointment, and Completion Process

Managing Service Requests and Reviews

Figure 2.11 illustrates the workflow from the provider's perspective in the service request process

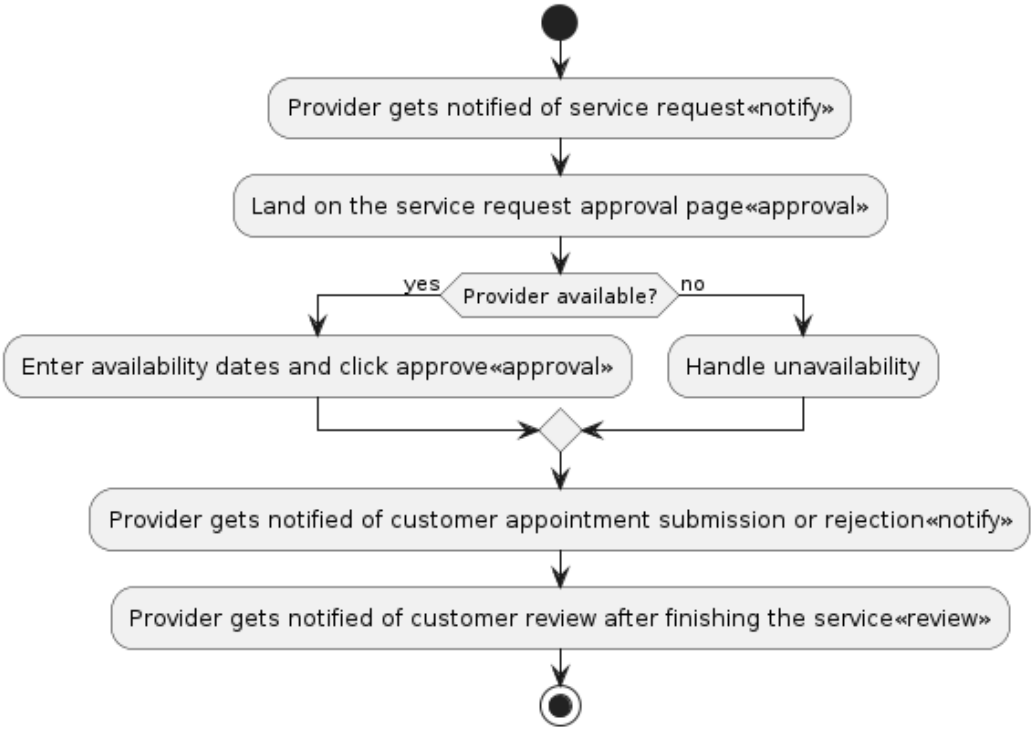

Fig. 2.11 Managing Service Requests and Reviews Process

2.3.4 Sequence Diagrams

The sequence diagrams in this section illustrate the interactions and message flows within the end-users application. These diagrams depict the chronological order of interactions between users and the system, detailing the steps involved in completing various tasks from both the customer and provider viewpoints.

Sign-up Process

Figure 2.12 depicts this sequence diagram that illustrates the user registration and role selection process

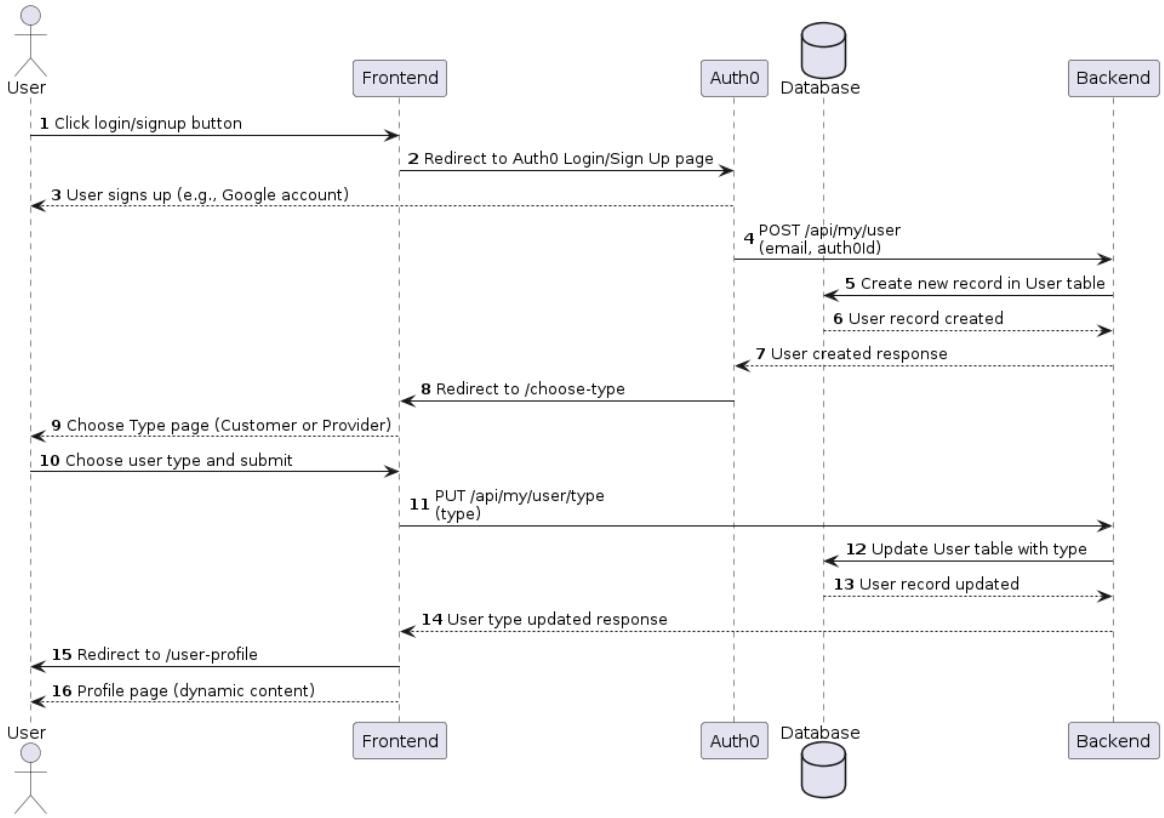

Fig. 2.12 Sign-up Sequence Diagram

Search and filter Process

Figure 2.13 illustrates the typical flow of the search and filtering.

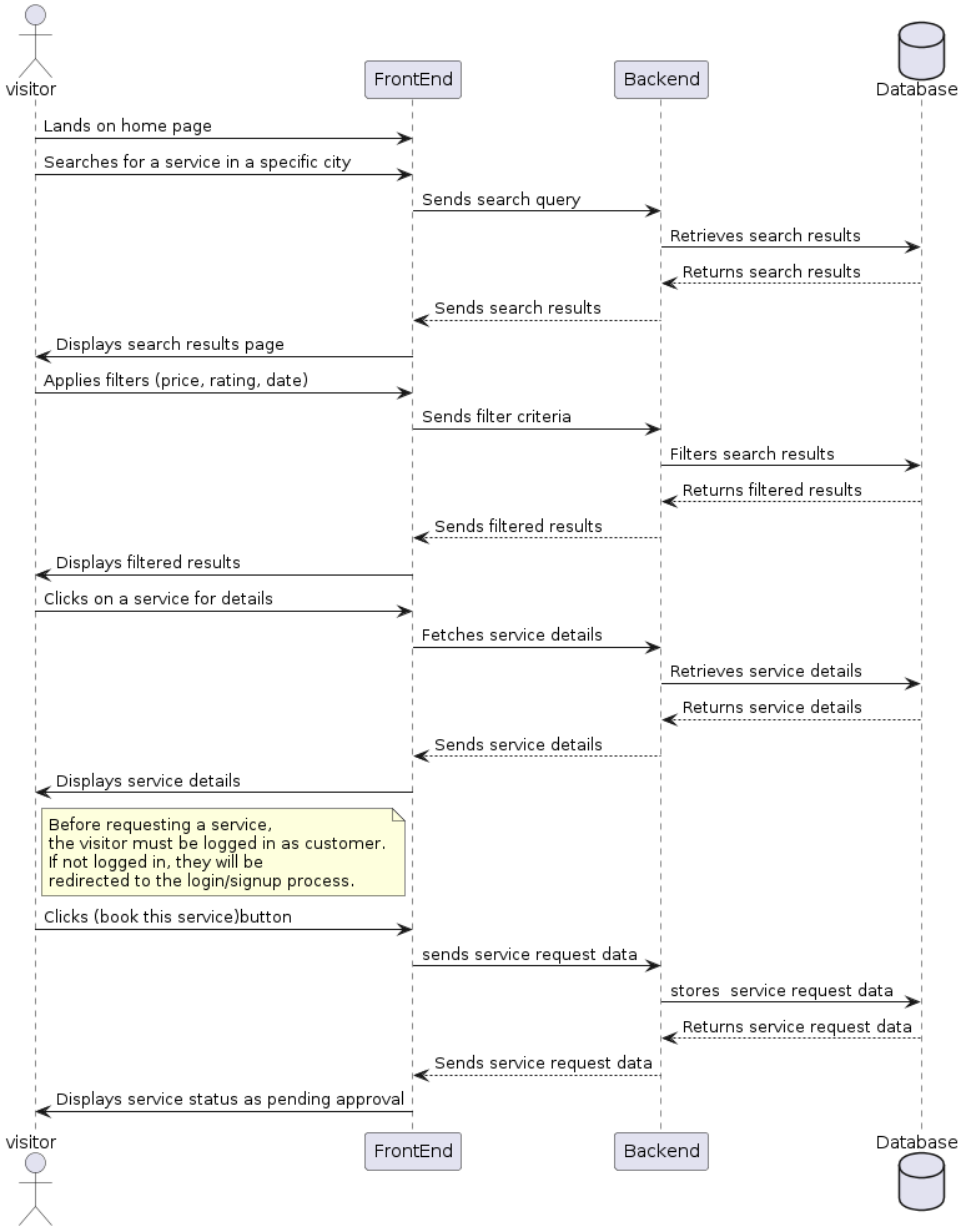

Fig. 2.13 Search and filter Sequence diagram

Service Creation and Approval Process

Figure 2.14 depicts the process of a service provider publishing a new service on the platform.

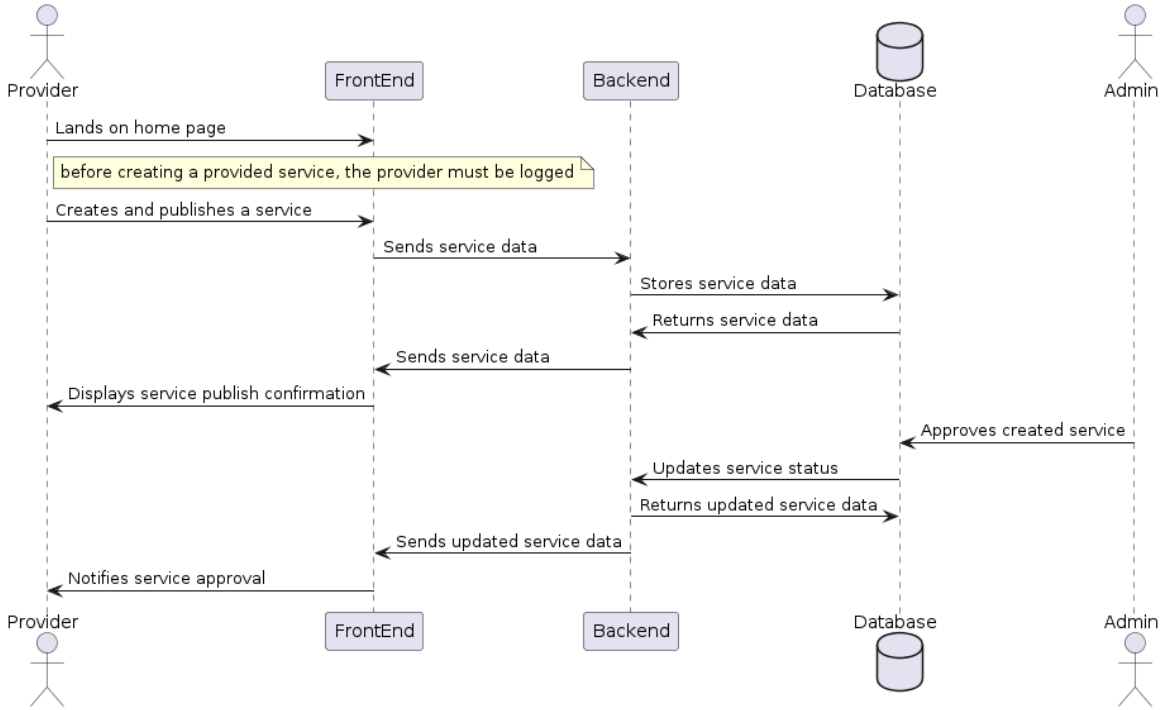

Fig. 2.14 Service Creation and Approval Process Sequence diagram

Typical Scenario Process

Figure 2.15 illustrates the service request life-cycle and the interactions between the customer, front-end, back-end, database, and provider in the platform.

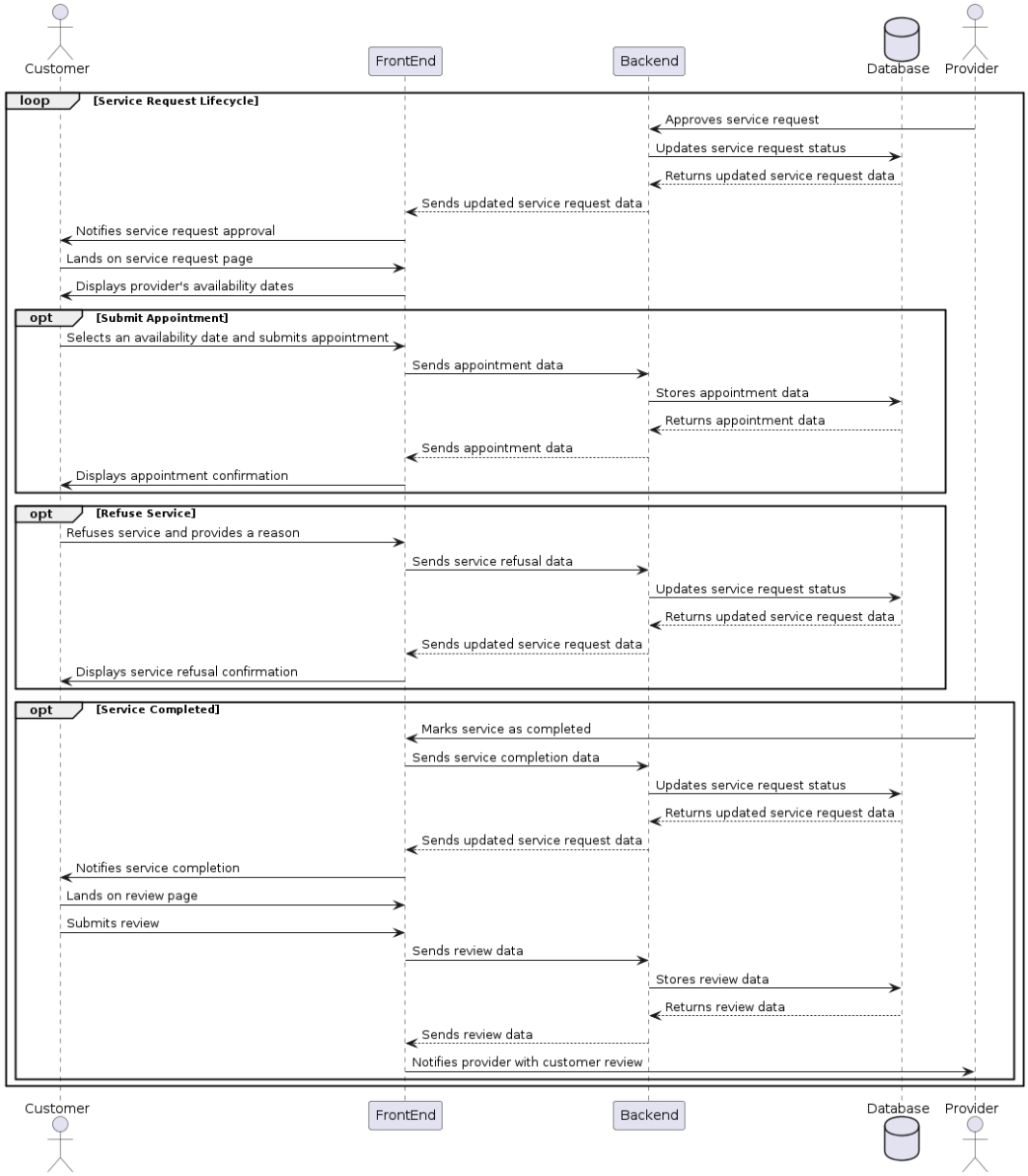

Fig. 2.15 Service Creation and Approval Process Sequence diagram

Request a Service Process

Figure 2.16 depicts the service request process when a customer clicks the "Book this Service" button to request a service.

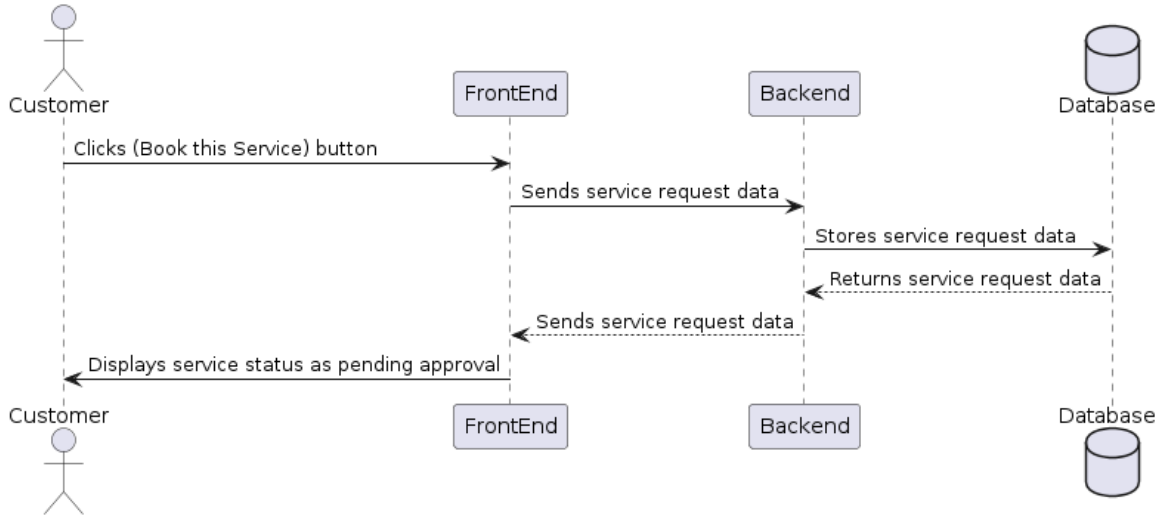

Fig. 2.16 Request a Service Sequence diagram

2.4 Admin Dashboard

The admin dashboard provides an interface for administrators to manage the system. It includes components and pages for viewing and managing users, services, categories, and requests.

2.4.1 Pages

The admin dashboard includes various pages that provide administrators with the necessary tools to manage the system. These pages allow administrators to oversee and control the different aspects of the platform, such as users, services, categories, and requests

Route	Description
/	Home page
/Tables/categories	Categories page
/Tables/Service	Services page
Tables/providers	Providers page
/provider/:providerId	Edit Provider Information page
/Tables/Customers	Customers page
/customer/:customerId	Edit Customer Information page

Table 2.15 Admin Dashboard Pages

2.4.2 Use Cases

In this section, we outline the use cases relevant to the admin dashboard. These use cases describe the interactions of administrators with the application, highlighting the functionalities available to them.

Figure 2.17 illustrates the use case scenario for an Admin actor interacting with an admin-dashboard

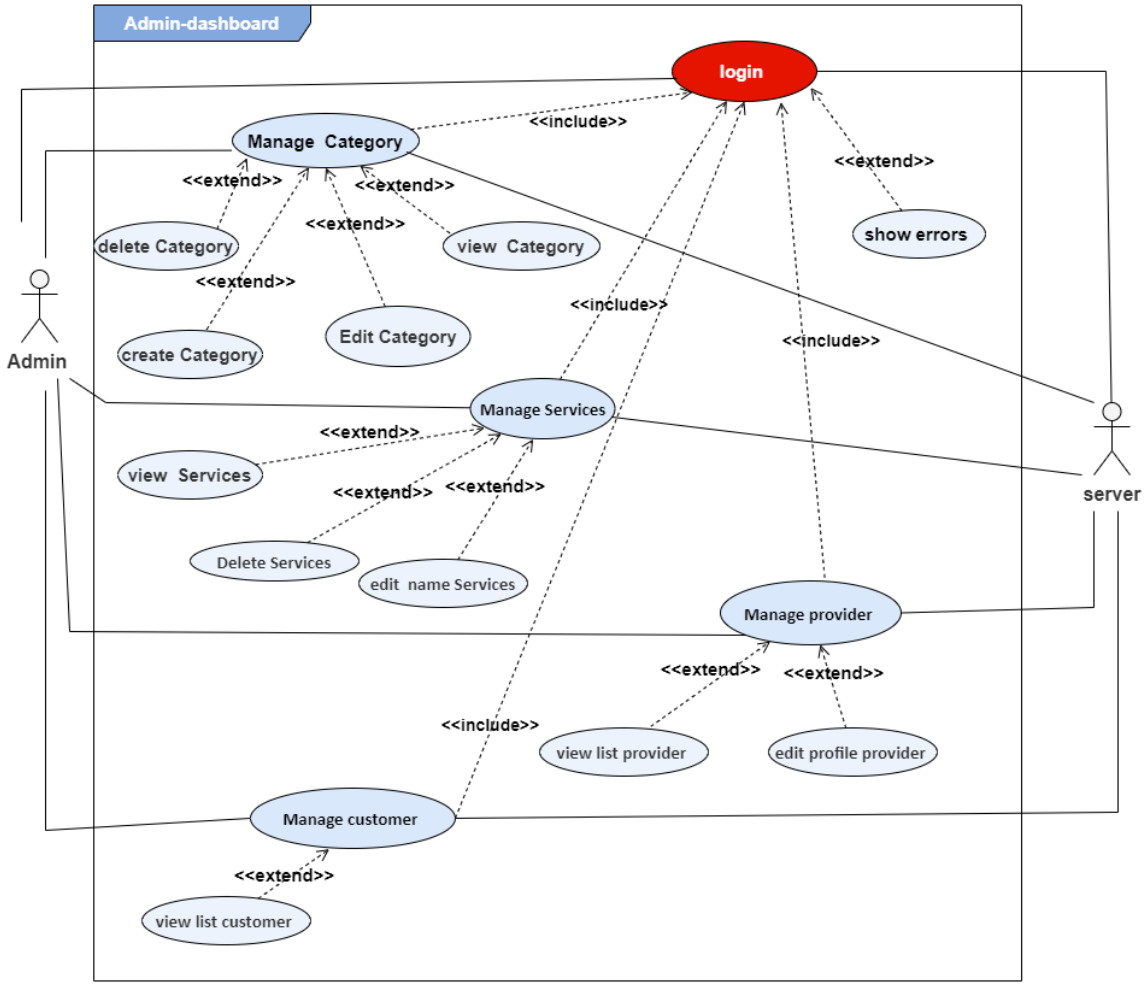

Fig. 2.17 Admin Use Case Diagram

2.4.3 Sequence Diagrams

Sequence diagrams illustrate the interactions and message flows within the admin dashboard. These diagrams show the chronological order of interactions between administrators and the system, detailing the steps involved in completing various administrative tasks.

Admin Login Process

Figure 2.18 illustrates the process of administrator login in the dashboard

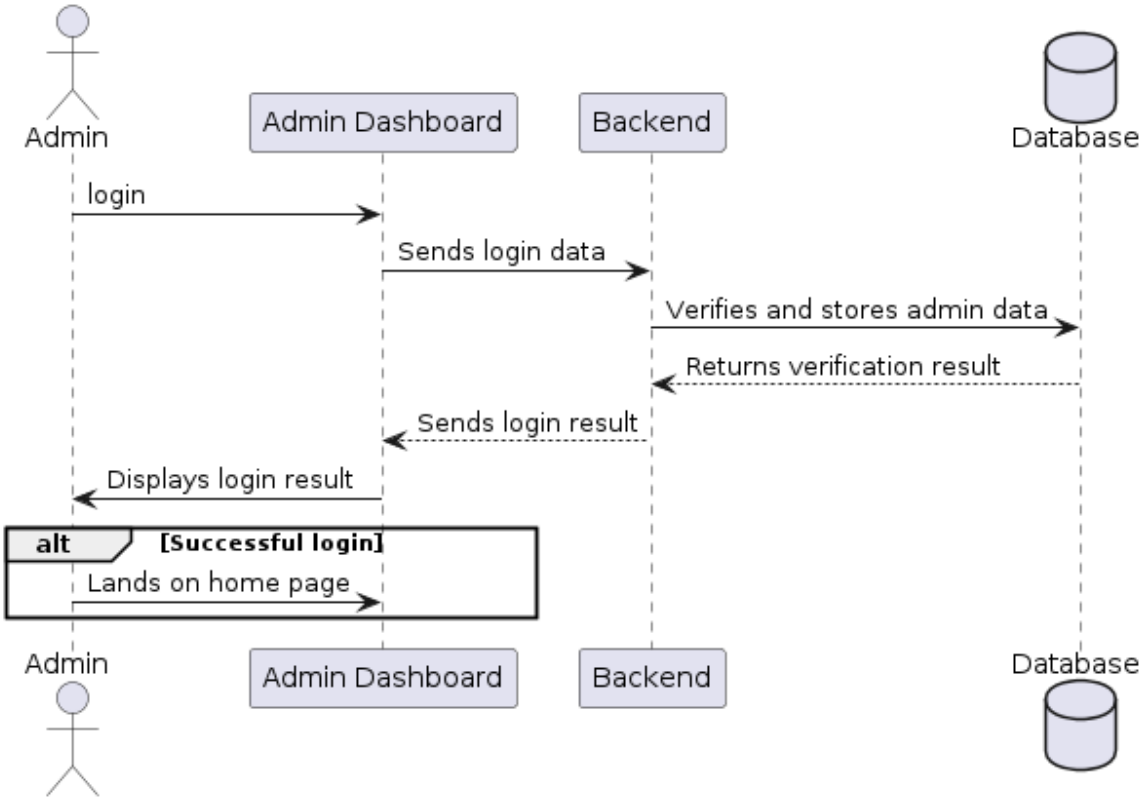

Fig. 2.18 Admin login Sequence Diagram

Display Users Data Process

Figure 2.19 illustrates the flow of data retrieval and display for different entities (providers, customers, services, and requests) in the system.

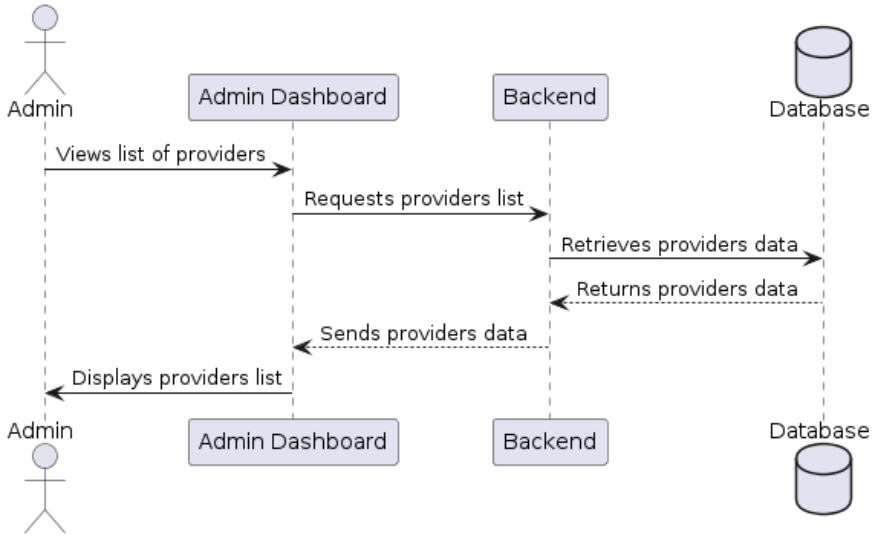

Fig. 2.19 Retrieving Lists Sequence Diagram

View Statistics Process

Figure 2.20 illustrates the process of an Administrator viewing statistics on the Admin Dashboard of the system.

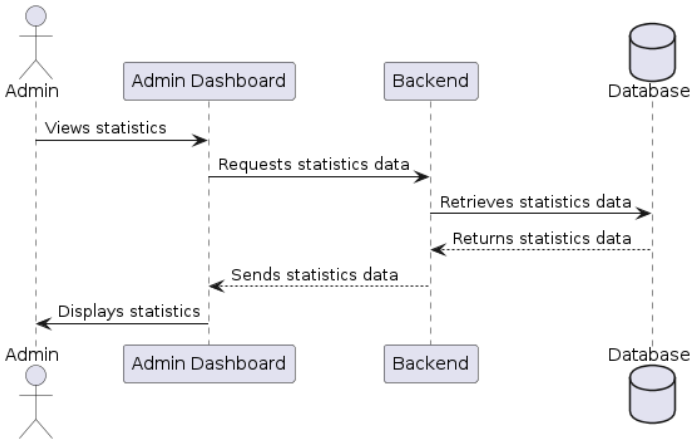

Fig. 2.20 View Statistics Sequence Diagram

Manage Pending Services Process

Figure 2.21 depicts the process of an Administrator managing pending services for approval and the subsequent notification to the Provider.

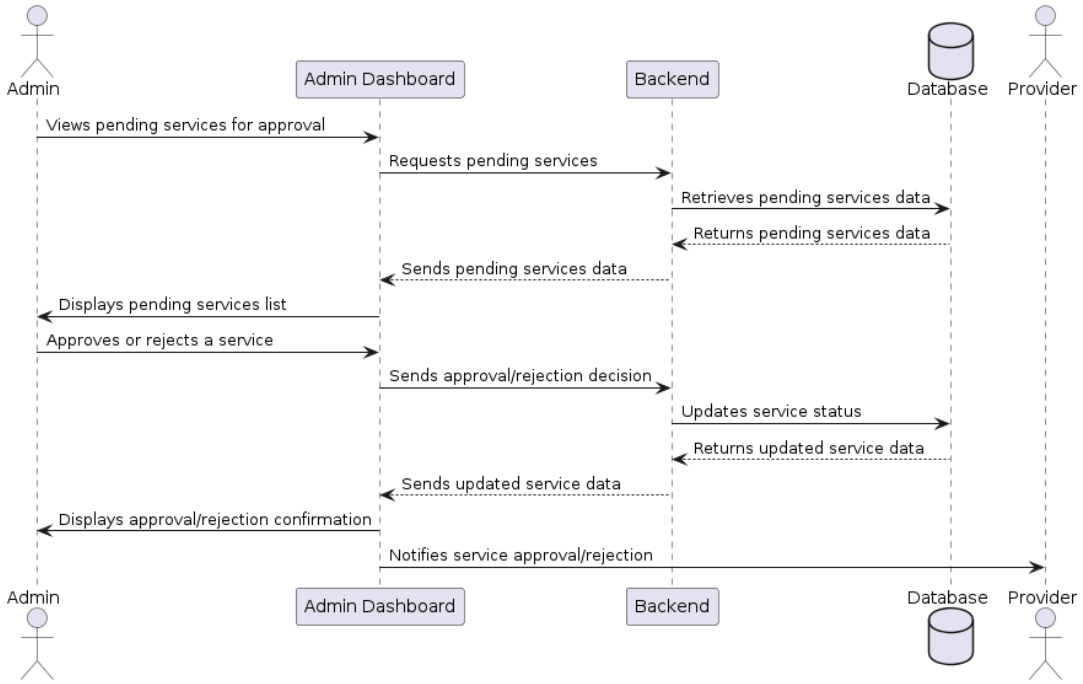

Fig. 2.21 View and Manage Pending Services Sequence Diagram

Manage Categories Process

Figure 2.22 illustrates the process of an Administrator creating/editing categories and services in the system.

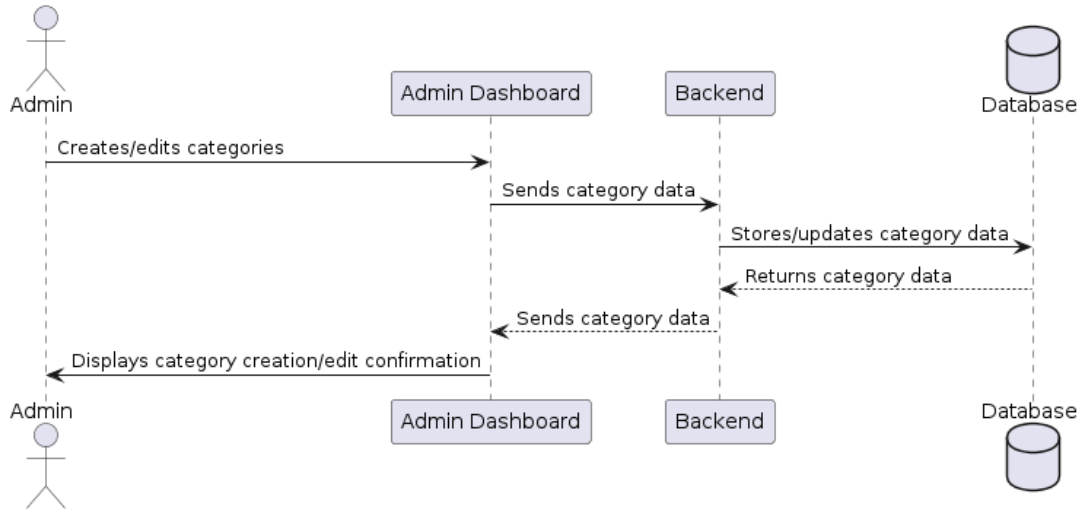

Fig. 2.22 Creating/Editing Categories and Services Sequence Diagram

2.5 Conclusion

The comprehensive system design outlined in this chapter demonstrates a well-structured and efficient architecture, leveraging both the C4 model and UML for clear and detailed representation. The system encompasses a robust back-end server with a sophisticated data model and API endpoints that ensure seamless interaction between the front-end applications and the database. The end-user applications provide interfaces for visitors, customers, and providers, facilitating service transactions. The admin dashboard equips administrators with tools to manage users, services, and requests, enabling effective oversight and control of the platform. Through detailed diagrams and flow charts, we have illustrated the interactions, workflows, and processes that underpin the system's architecture.

Chapter 3

System Implementation and Deployment

In this chapter, we first present the technologies used in the development, deployment, and maintenance of our system. This includes a comprehensive overview of our development environment, the development stack, and various services utilized for deployment, authentication, and image storage. After that, we present both the end-users and admin dashboard applications through a set of screenshots of different user interfaces provided with textual description.

3.1 Used Technologies

In this section, we present different modern web development technologies used to build our system, including the development environment and stack, as well as authentication and deployment services.

3.1.1 Development Environment

Our development environment consists of the following tools:

- **Visual Studio Code (VS Code)**: An open-source source code editor developed by Microsoft. It provides an integrated development environment for developers, with features such as error detection, auto-completion, code formatting, version control, and more [35].
- **Google Chrome**: A popular web browser developed by Google. It is known for its speed, simplicity, and user-friendly interface. Chrome offers an integrated web developer tools that lets developers edit pages on-the-fly and diagnose problems quickly, which helps you build better websites, faster [3].

- **Postman**: A collaboration platform for API development. Postman simplifies each step of building an API and streamlines collaboration so you can create better APIs faster [20].
- **Git**: A free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git tracks changes in source code during software development, facilitating collaboration [10].
- **GitHub**: A cloud-based code repository service that provides hosting for software development and version control using Git. It offers distributed version control and source code management functionality, along with features such as bug tracking, task management, and continuous integration [11].

3.1.2 Development Stack

Back-end Stack

Our back-end is built using the following technologies:

- **Node.js**: A JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, ideal for building scalable network applications [17].
- **Express**: A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It facilitates the rapid development of Node-based web applications [8].
- **PostgreSQL**: A powerful, open-source object-relational database system with over 30 years of active development. It has a strong reputation for reliability, feature robustness, and performance [19].
- **Prisma**: An open-source ORM (Object-Relational Mapping) tool for Node.js and TypeScript. Prisma makes database access easy by providing a type-safe API and simplifying the development of database queries [21].

Front-end Stack

Our front-end stack includes:

- **HTML**: Stands for Hypertext Markup Language. It's the standard markup language used for creating and designing web pages. HTML defines the structure and content of a web page by using a variety of tags and attributes [12].

- **CSS**: Stands for Cascading Style Sheets. It's a style sheet language used for describing the presentation of a document written in HTML or XML, including colors, fonts, spacing, and layout [6].
- **JavaScript**: A programming language used in web development to add dynamics and interactivity to web pages [14].
- **Tailwind CSS**: A utility-first CSS framework for rapidly building custom designs. Tailwind provides low-level utility classes that enable developers to build complex designs without writing custom CSS [27].
- **React**: A JavaScript library for building user interfaces, maintained by Facebook and a community of individual developers and companies. React allows developers to create large web applications that can update and render efficiently in response to data changes [22].
- **Vite**: A modern front-end build tool that provides a faster and leaner development experience for modern web projects. Vite leverages ES modules and is designed to be performant and easy to configure [34].

3.1.3 Deployment Services

We use the following services for deploying our back-end server, the database, and both the end-users and admin dashboard applications:

- **Vercel**: We have used Vercel service to deploy the back-end server. Vercel is a platform for frontend frameworks and static sites, built to integrate with headless content management systems (CMS) and APIs. Vercel offers features like automatic deployments, serverless functions, and edge caching to optimize performance and scalability [33].
- **Supabase**: We have used Supabase service to deploy the database. Supabase is an open-source Firebase alternative that provides backend services such as database, authentication, and storage. Supabase is built on top of PostgreSQL, enabling real-time capabilities and easy integration with front-end applications [25].
- **Render**: We have used Render service to host both the end-users and admin dashboard applications. Render is a unified cloud to build and run all your applications with ease. Render offers a straightforward deployment experience, automatic scaling, and a wide array of managed services for modern applications [23]. Render also can also host the

back-end services, but it limits the requests to 50 seconds, that is why we have used Vercel service instead.

3.1.4 Third-party Services

Mhenni also leverages third-party services to accomplish other tasks, namely, users authentication and authorization, and image storage:

- **Auth0**: We have used Auth0 service to manager users authentication and authorization to the end-users application. Auth0 provides various authentication methods, including social logins, multi-factor authentication, and single sign-on (SSO) [1].
- **Cloudinary**: We have used Cloudinary service to store images such users profile pictures. Cloudinary offers comprehensive media upload, storage, management, manipulation, and delivery capabilities, optimized for web and mobile applications [4].

3.2 Presentation of the Application

Mhenni's end-users application [15] and admin dashboard [16] are both already up and running. In what follows, we present an explanation of different users interfaces for both end-users and admin dashboard, highlighting key features and functionalities that facilitate user interaction and system management.

3.2.1 End-users Application

The end-users application is designed to provide a seamless and intuitive user experience. The following subsections detail the various interfaces and features available to end-users.

Public Pages

The public pages that are dedicated to the visitor include:

Home Page: The home page provides an overview of the application's main features, including navigation to different sections such as a search section and categories, see Figure3.1.

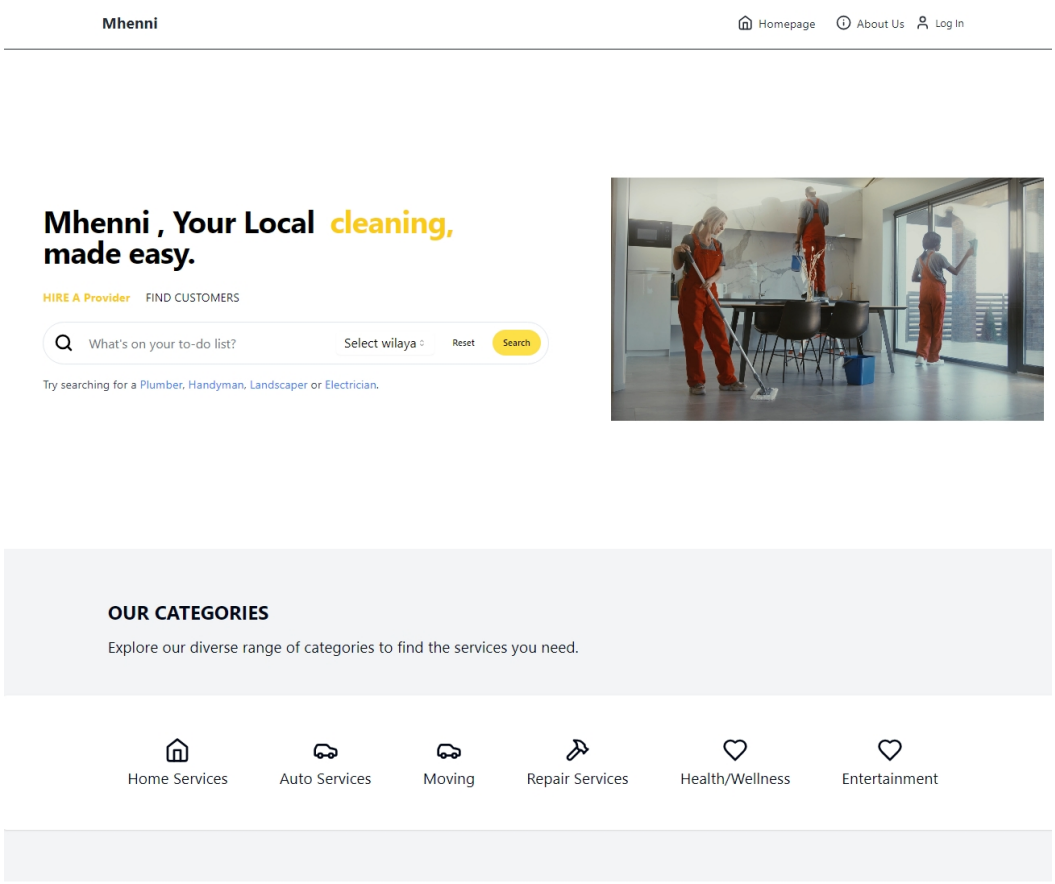

Fig. 3.1 The Home Page

Search Page: This page allows users to search for providers and services based on various criteria. see Figure3.2

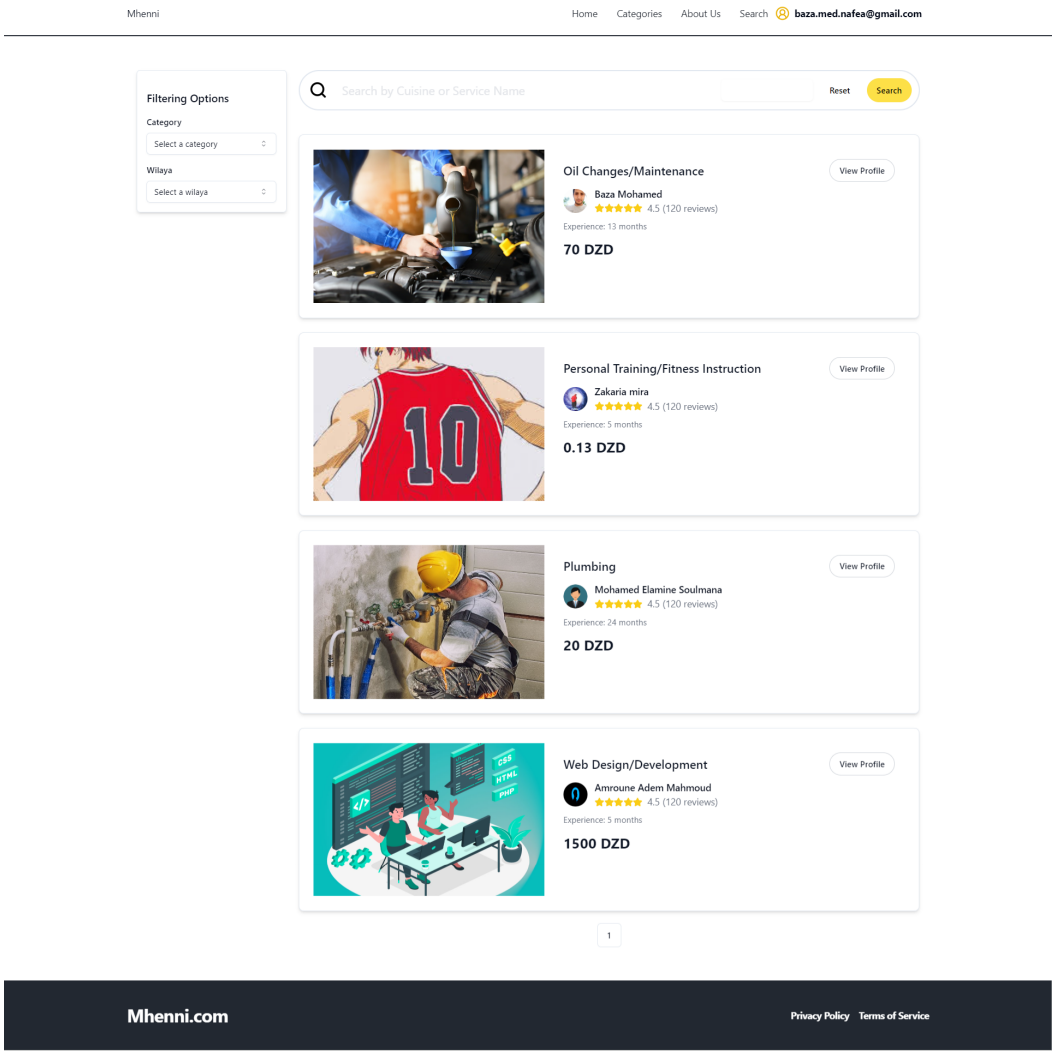

Fig. 3.2 The Search Page

User Profile Page: This page allow users(Providers/Customers) to modify their basic information such as First Name , Last Name , Address and their Profile Picture , see Figure3.3

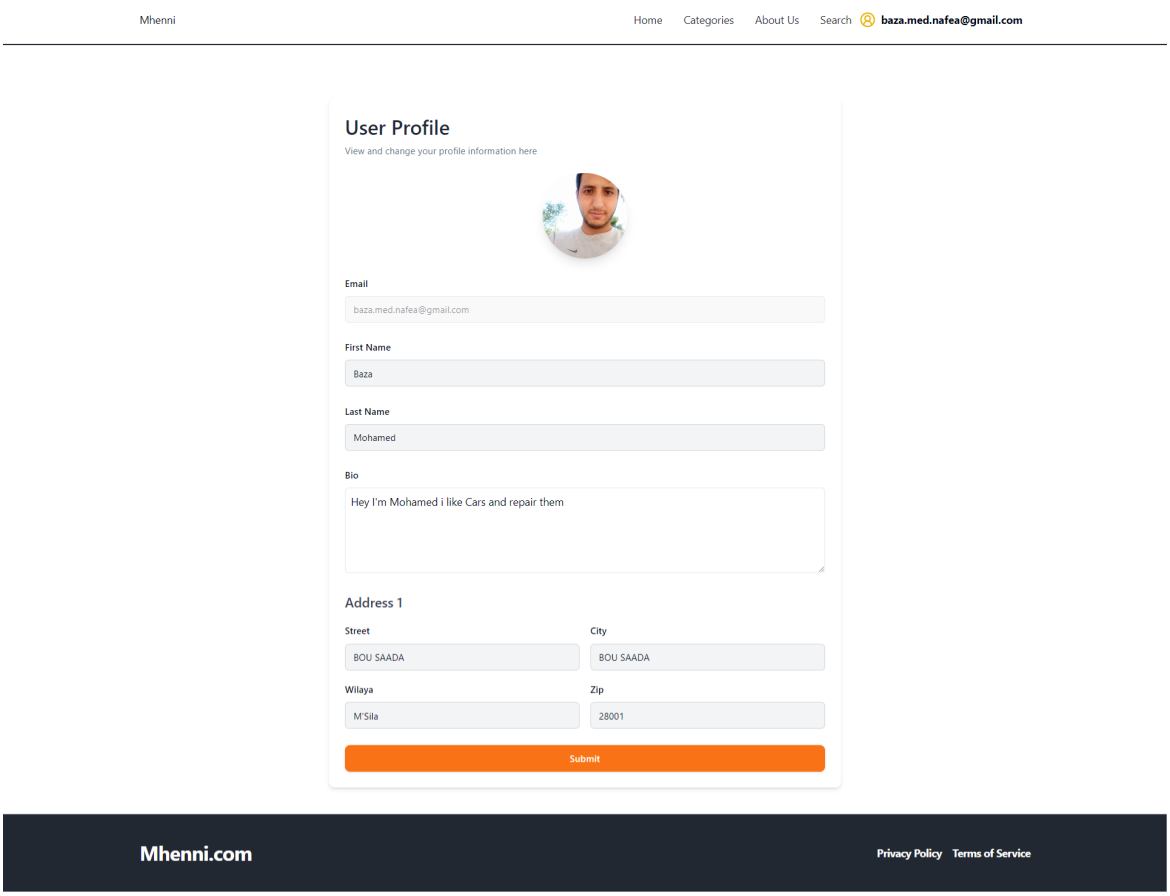

Fig. 3.3 The User Profile Page

Provider Details: This page displays detailed information about a selected provider, including their services, reviews, and contact information. see Figure3.4

Service Details: This page provides comprehensive information about a specific service, including descriptions, pricing, and the ability to book that selected service. see Figure3.5

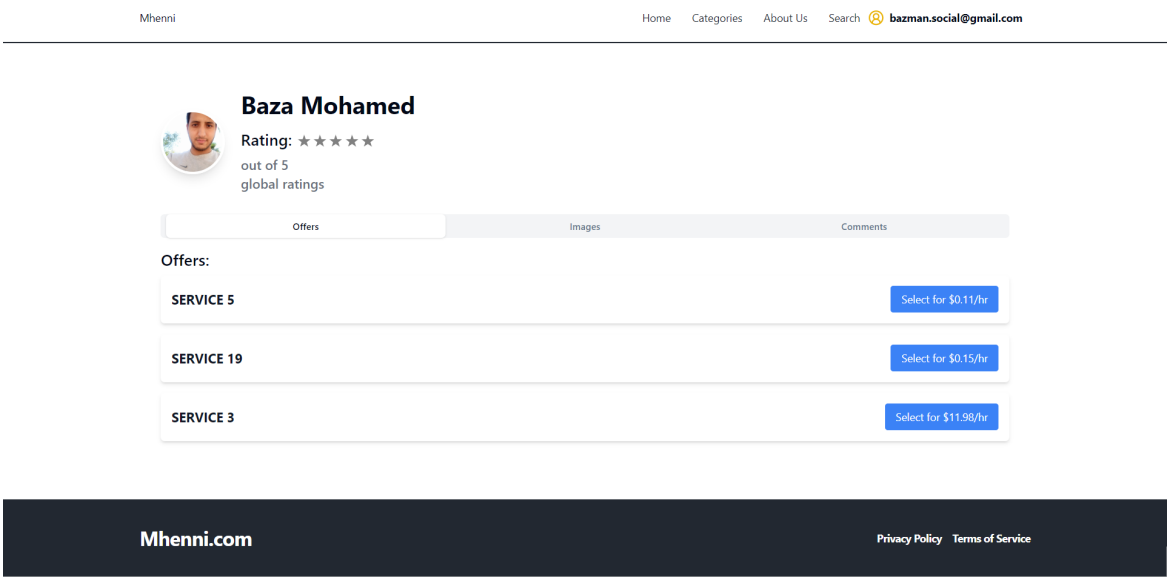

Fig. 3.4 The Provider Details Page

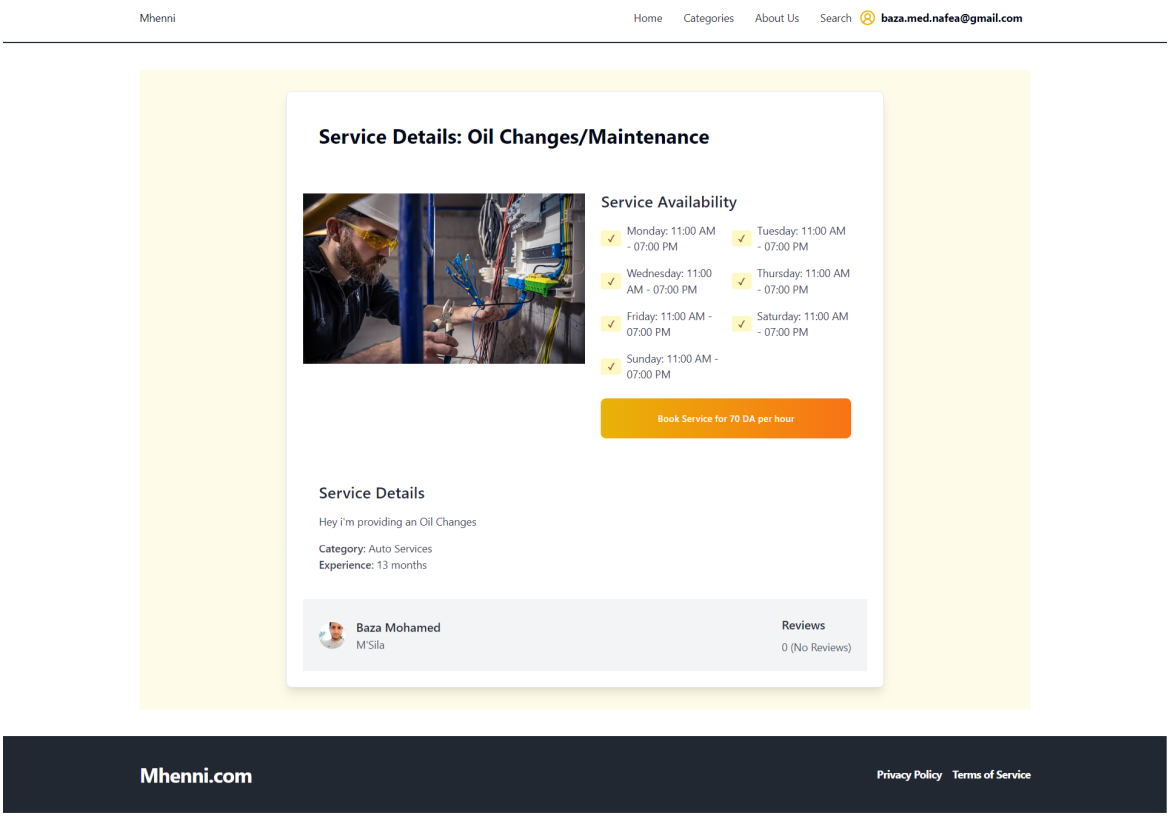

Fig. 3.5 The Service Details Page

Provider Pages

The provider pages enable service providers to manage their profiles, services, and interactions with customers. including:

Service Management: Providers can add, edit, and remove services they offer. see Figure3.6 and Figure3.7

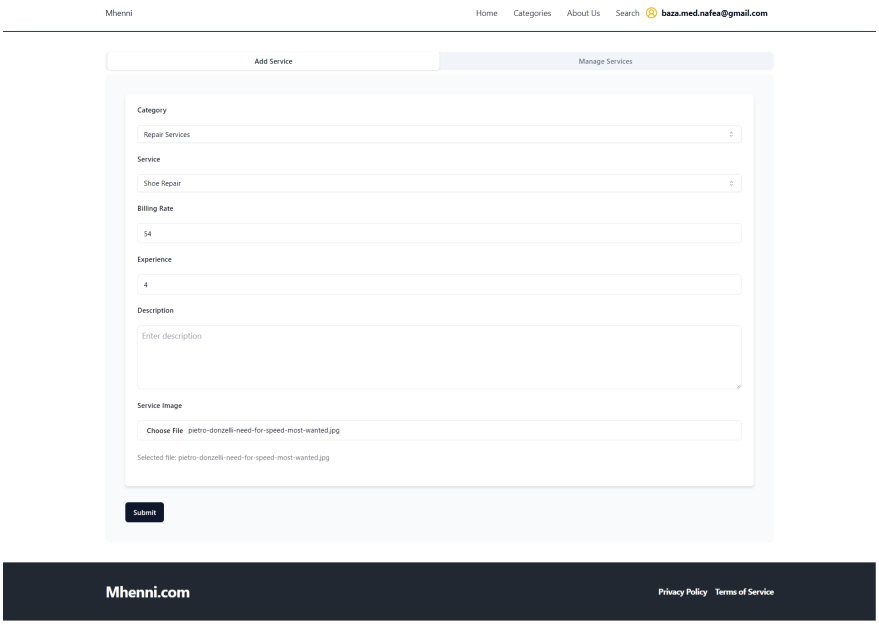

Fig. 3.6 Managing the Services Page

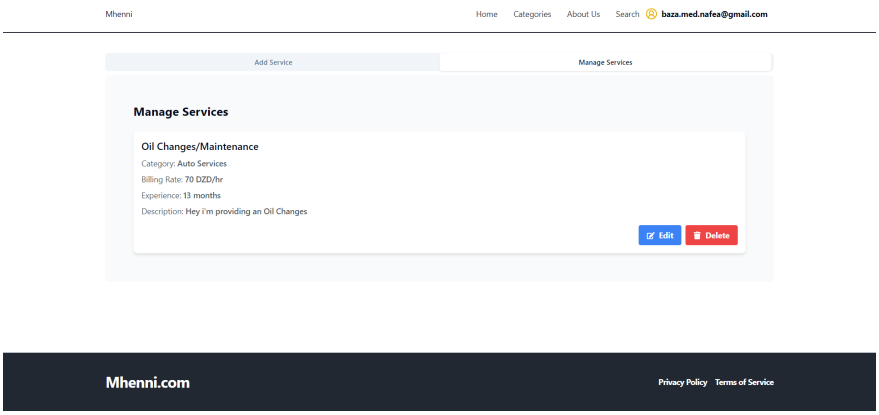

Fig. 3.7 Add a new Service Page

Request Management: This interface allows providers to manage requests with customers.see Figure3.8

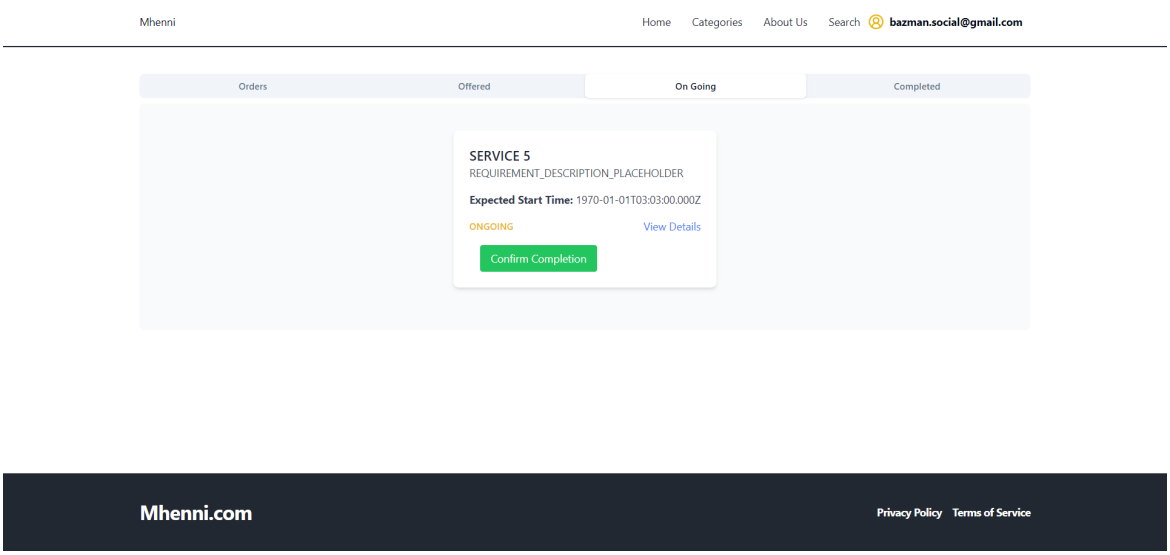

Fig. 3.8 Request Management Page(Customer)

Customer Pages

The customer pages allow users to manage interactions with providers. Key interfaces include:

Request Management: This interface allows customers to manage requests with providers. see Figure 3.9

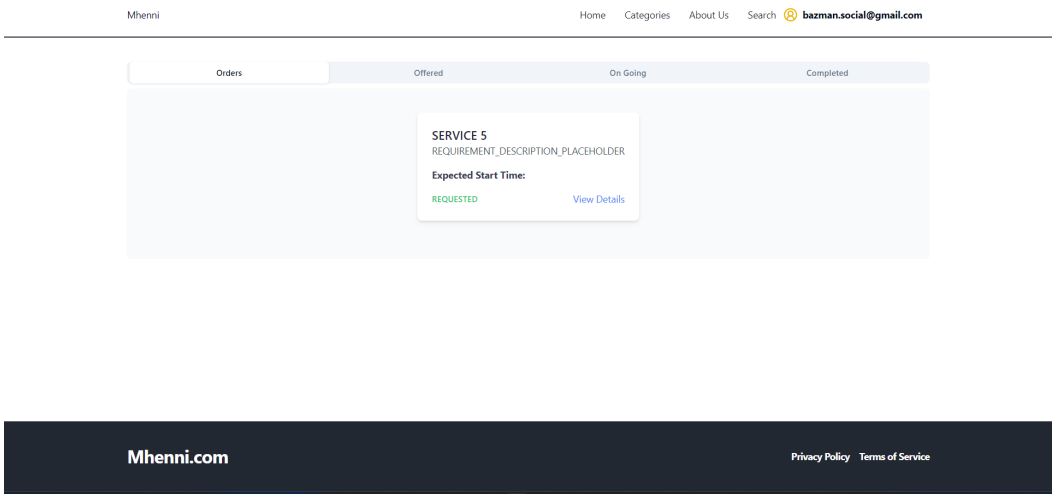

Fig. 3.9 Request Management Page(Customer)

3.2.2 Admin Dashboard

The admin dashboard is designed for administrators to manage the system efficiently. The following subsections describe the key interfaces and functionalities available in the admin dashboard.

The Dashboard: The main landing page for administrators, providing a comprehensive overview of the system's current status. This page serves as a quick snapshot of the platform's health and recent interactions, as shown in Figure 3.10.

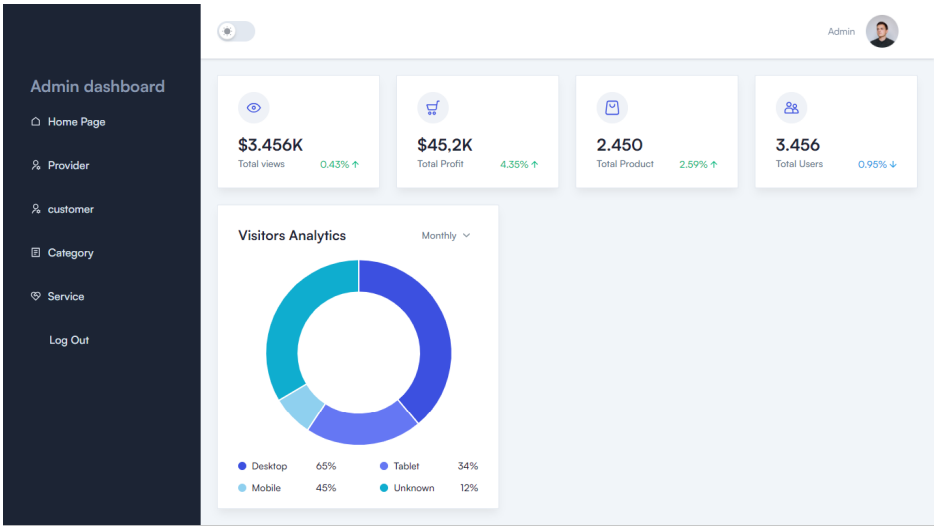

Fig. 3.10 Admin Dashboard Overview

Customer Management: This interface allows administrators to view, add, edit, or remove customer accounts. It includes detailed information about each customer, see figure 3.11

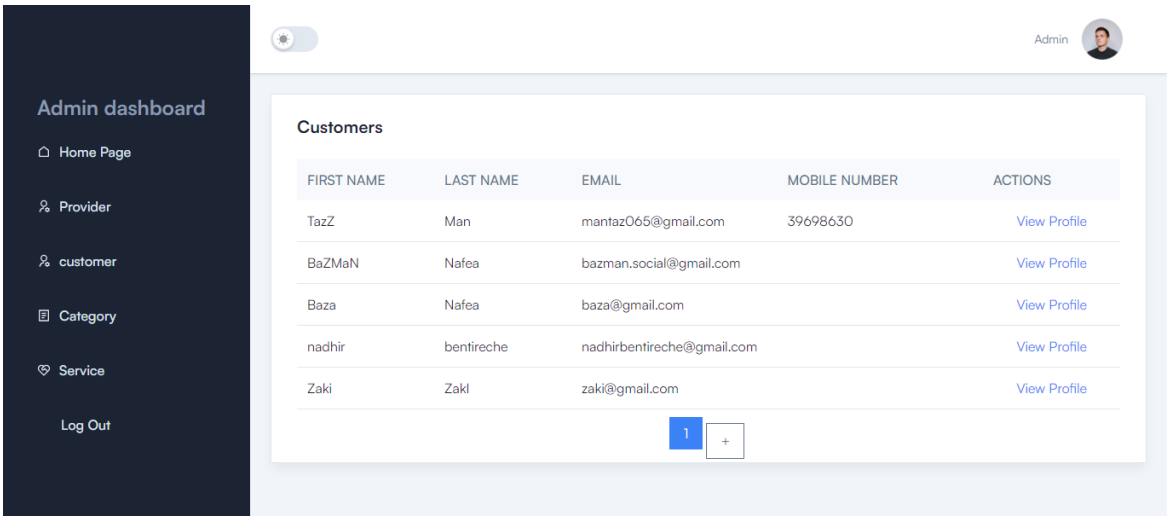

Fig. 3.11 Customer Management Interface

Provider Management: Similar to customer management, this section allows administrators to manage provider accounts. Admins can approve new providers, review and update provider details, and manage their service listings. See Figure 3.12 for an illustration of this interface.

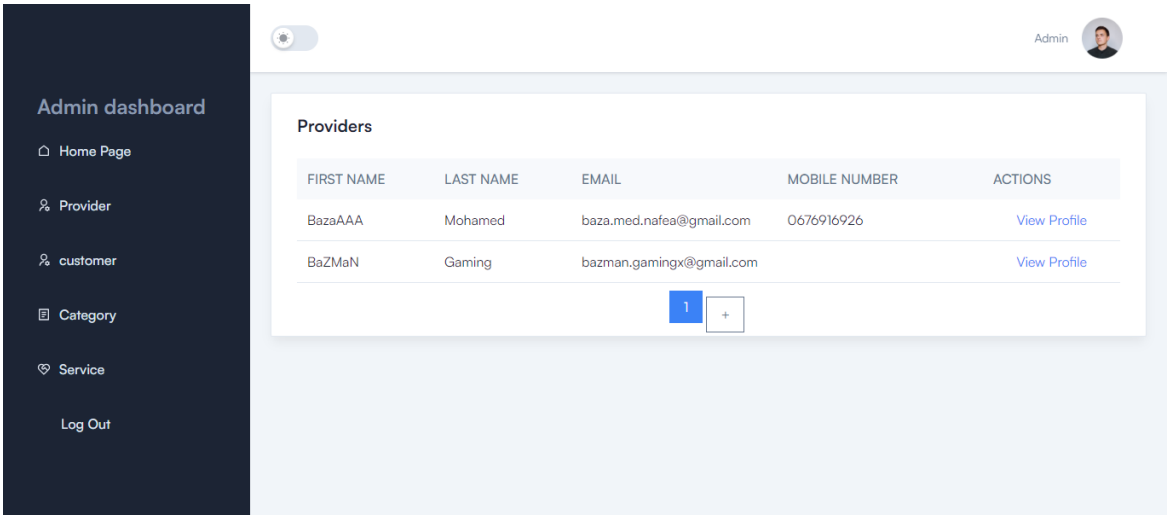

Fig. 3.12 Provider Management Interface

Service Management: This interface allows administrators to oversee all the services listed on the platform. Admins can add, edit, or remove services and categorize them appropriately. Figure 3.13 illustrates this management interface.

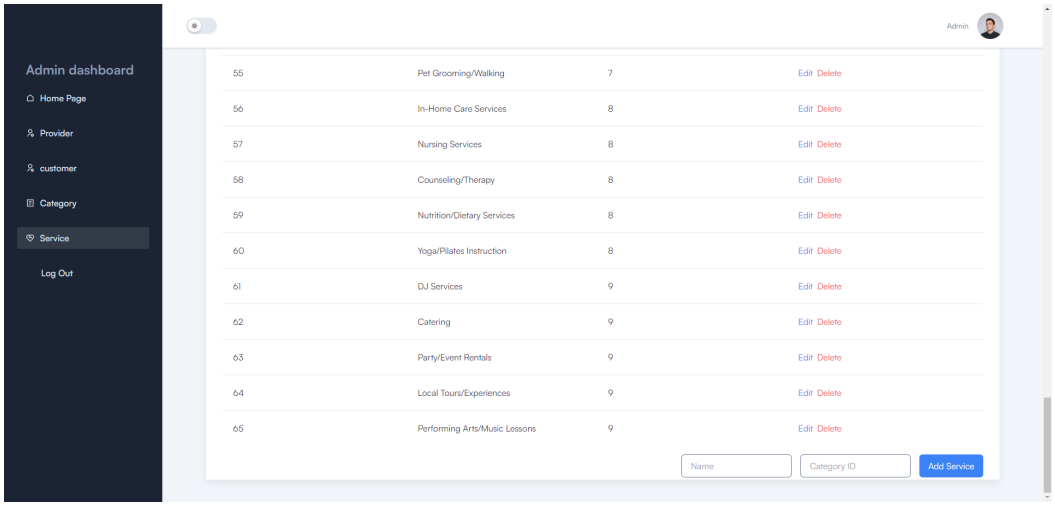

Fig. 3.13 Service Management Interface

Category Management: This interface allows administrators to oversee all the categories listed on the platform. Admins can add, edit, or remove services and categorize them appropriately. Figure 3.14.

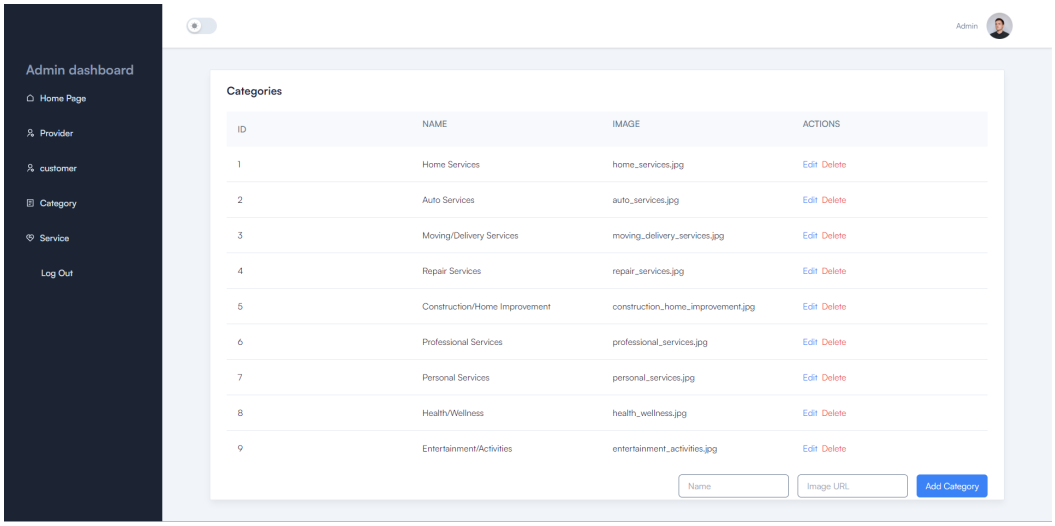

Fig. 3.14 Request Management Interface

3.3 Conclusion

In conclusion, this chapter has provided an overview of the technologies and tools used in the development, deployment, and maintenance of our system. We have detailed the development environment, the back-end and front-end stacks, and various deployment, authentication, and image storage services. Additionally, we have introduced the user interfaces for both end-users and administrators, highlighting key features and functionalities.

Conclusion

This report presented a comprehensive documentation of our proposed online marketplace for on-demand services, called Mhenni. The reports presents in details the whole software engineering stages starting from analysis, to the design, until implementation and deployment. By leveraging modern web technologies and tools such as Node.js, Express, React, and various deployment services, we have built a robust platform that addresses the needs of different types of users including service providers and customers as well as the administrators.

While we have successfully implemented a wide range of features, there are several avenues for future work to enhance the system further. This include leveraging AI models like ChatGPT, to allow visitors to describe their service needs in natural language; adding more refined filtering options for services, such as by scale (local, regional, national), to improve the accuracy and relevance of search results for visitors as well developing comprehensive revenue tracking and analytics features for providers to help them manage their business more effectively and make informed strategic decisions.

In conclusion, while the current system provides a solid and functional platform for service interactions, the outlined future work presents significant opportunities for enhancement and growth. By addressing these areas, we can further improve user experience, operational efficiency, and the overall value provided by the platform.

References

[1] Auth0. Auth0: Secure access for everyone. but not just anyone., 2024. URL https://auth0.com/. Accessed on June 07, 2024.

[2] C4 Model. The c4 model for visualising software architecture. URL https://c4model.com/. Accessed on June 07, 2024.

[3] Chrome. Google chrome - the fast, secure, and free web browser, 2024. URL https://www.google.com/chrome/. Accessed on June 07, 2024.

[4] Cloudinary. Cloudinary - image and video api platform, 2024. URL https://cloudinary.com/. Accessed on June 07, 2024.

[5] Mike Cohn. *User Stories Applied: For Agile Software Development*. Addison-Wesley, 2004. ISBN 978-0321205681.

[6] CSS. Css - cascading style sheets, 2024. URL https://www.w3schools.com/css/css_intro.asp. Accessed on June 07, 2024.

[7] Door Dash. Door dash - discover restaurants and more near you. URL https://www.uber.com/es/en/. Accessed on June 07, 2024.

[8] Express. Express.js - node.js web application framework, 2024. URL https://expressjs.com/. Accessed on June 07, 2024.

[9] Freehali. Freehali, 2024. URL https://www.freehali.com/. Accessed on June 07, 2024.

[10] Git. Git - distributed even if your workflow isn't, 2024. URL https://git-scm.com/. Accessed on June 07, 2024.

[11] GitHub. Github - let's build from here, 2024. URL https://www.github.com/. Accessed on June 07, 2024.

[12] HTML. Html - hypertext markup language, 2024. URL https://www.w3schools.com/html/. Accessed on June 07, 2024.

[13] Incremental Model. Incremental model in software engineering. URL https://www.javatpoint.com/software-engineering-incremental-model. Accessed on June 07, 2024.

[14] JavaScript. Javascript - 27 years, 1,444,231 libraries and counting..., 2024. URL https://www.javascript.com/. Accessed on June 07, 2024.

[15] Mhenni. Mhenni - services by anyone to everyone. URL https://mhenni.onrender.com/. Accessed on June 07, 2024.

[16] Mhenni Admin. Mhenni admin dashboard. URL https://mhenni-dashboard.onrender.com/. Accessed on June 07, 2024.

[17] NodeJS. Node.js — run javascript everywhere, 2024. URL https://nodejs.org/en. Accessed on June 07, 2024.

[18] Ouedkniss. Ouedkniss - annonces algérie | vente achat. URL https://www.ouedkniss.com/. Accessed on June 07, 2024.

[19] PostgreSQL. Postgresql - the world's most advanced open source database, 2024. URL https://www.postgresql.org/. Accessed on June 07, 2024.

[20] Postman. Postman, 2024. URL https://www.postman.com/. Accessed on June 07, 2024.

[21] Prisma. Prisma - simplify working and interacting with databases, 2024. URL https://www.prisma.io/. Accessed on June 07, 2024.

[22] React. React - the library for web and native user interfaces, 2024. URL https://www.react.dev/. Accessed on June 07, 2024.

[23] Render. Render - cloud application hosting for developers, 2024. URL https://www.render.com/. Accessed on June 07, 2024.

[24] StratoFlow. Stratoflow - high performance flow solutions, 2024. URL https://www.stratoflow.com/. Accessed on June 07, 2024.

[25] Supabase. Supabase - build in a weekend, scale to millions, 2024. URL https://supabase.com/. Accessed on June 07, 2024.

[26] Swiga. Swiga, 2024. URL https://swigaapp.com/. Accessed on June 07, 2024.

[27] TailwindCSS. Tailwind css - rapidly build modern websites without ever leaving your html, 2024. URL https://www.tailwindcss.com/. Accessed on June 07, 2024.

[28] TaskRabbit. Taskrabbit - book your next task, 2024. URL https://www.taskrabbit.com/. Accessed on June 07, 2024.

[29] Thumbtack. Thumbtack - care for your home | find local pros & reviews, 2024. URL https://www.thumbtack.com/. Accessed on June 07, 2024.

[30] Uber. Uber - drive when you want, make what you need. URL https://www.uber.com/es/en/. Accessed on June 07, 2024.

[31] UML. Uml - unified modeling language, 2024. URL https://www.uml.org/. Accessed on June 07, 2024.

[32] Urban Company. Urban company - experienced, hand-picked professionals to serve you at your doorstep, 2024. URL https://www.urbancompany.com/. Accessed on June 07, 2024.

[33] Vercel. Vercel - develop. preview. ship., 2024. URL https://vercel.com/. Accessed on June 07, 2024.

[34] Vite. Vite - next generation frontend tooling, 2024. URL https://vitejs.dev/. Accessed on June 07, 2024.

[35] VSCode. Vscode - code editing. redefined., 2024. URL https://code.visualstudio.com/. Accessed on June 07, 2024.

[36] WorkVally. Workvally - a freelance marketplace connects freelancers with clients and companies, 2024. URL https://www.workvally.com/. Accessed on June 07, 2024.

Abstract

This report presents Mhenni, an online marketplace for on-demand services such as home and transportation services. Mhenni bridges the current gap in the Algerian market between service providers and customers. On one hand, Mhenni gives providers the opportunity to offer their services to potential customers and showcase their skills and accumulate reviews and ratings. On the other hand, customers can effortlessly find providers, schedule appointments, negotiate prices, and manage their budgets. This report provides a comprehensive technical documentation of the system. First of all, in the business analysis stage, we have documented the process of collecting relevant information about system requirements and studying the life cycle of a service from the moment being created until being served. In the design stage, we provided a detailed description of the system in terms of architecture and behavior. The implementation and deployment of Mhenni's proof of concept has been documented as well by presenting different development and deployment technologies.

Keywords: Online Marketplace, On-demand Services

ملخص

يقدم هذا التقرير موقع مهني، وهو سوق عبر الإنترنت للخدمات حسب الطلب مثل خدمات المنازل والنقل. مهني يسد الفجوة الحالية في السوق الجزائرية بين مقدمي الخدمات والعملاء. من ناحية، يمنح مهني مقدمي الخدمات الفرصة لعرض خدماتهم للعملاء المحتملين وعرض مهاراتهم وتجميع التعليقات والتقييمات. ومن ناحية أخرى، يمكن للعملاء العثور بسهولة على مقدمي الخدمات، وجدولة المواعيد، والتفاوض على الأسعار، وإدارة ميزانياتهم. يقدم هذا التقرير توثيقًا فنيًا شاملاً للنظام. بداية، في مرحلة تحليل الأعمال، قمنا بتوثيق عملية جمع المعلومات ذات الصلة حول متطلبات النظام ودراسة دورة حياة الخدمة من لحظة إنشائها حتى تقديمها. وفي مرحلة التصميم قدمنا وصفاً تفصيلياً للنظام من حيث البنية والسلوك. وقد تم أيضًا توثيق تنفيذ ونشر إثبات مفهوم مهني من خلال تقديم تقنيات التطوير والنشر المختلفة.

الكلمات المفتاحية: السوق الالكتروني، الخدمات حسب الطلب