PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

MOHAMED BOUDIAF UNIVERSITY - M'SILA

FACULTY: Mathematics and Computer Science

DEPARTMENT: Computer Science

N° : ……..

DOMAIN: Mathematics and Computer Science

FIELD: Computer Science

OPTION: Information Systems and Software Engineering

A Dissertation Submitted in Partial Fulfilment of the

Requirements for the Degree of Academic Master

By:

Mira Zakarya Abderrahmen

Benlaiter Mohammed

SUBJECT

Enhancing Computer Science Advancements through a Skill Exchange Platform.

Board of Examiners:

Kettaf Abdelouahab	University of M'sila	President
Dr. Brahimi Mahmoud	University of M'sila	Reporter
Madiha Hallassa	University of M'sila	Examiner

Academic year: 2023 /2024

DEDICATION

" Starting with the name of Almighty Allah, I thank Him for guiding me to success and giving me the strength to work on this special project. All sincerity and effort are for Him. I dedicate this thesis to my parents for the love they have always shown me, the courage and support I have received from them to complete my studies. Not forgetting my siblings whom always been there for me to help me through everything, supporting me through everything. And a special dedication to anyone who has helped me in my life."

Benlaiter Mohammed

" First of all I want to say alhamdulillah, Allah gave me everything I need, alhamdulillah, then I want to send a huge thank you to my lovely parents, for all their hardwork and dedicating a huge part of their life just for me to be the person who I'm today, and to my amazing siblings, my Allah bless them, I've always felt so lucky to have them by my side growing up, they are the best sibling a younger brother could ask for, and after all these times I can only say, it was a wonderful experience."

Mira Zakarya Abderrahmen

Acknowledgments

This thesis would have been possible without the advice, experience, and skill of Professor Mahmoud Brahimi. We thank him for standing with us and helping us throughout the preparation of this thesis.

We would also like to thank the committee Ms Madiha Hallassa and Mr Kettaf Abdelouahab for their time and the experience they will share with us.

We would also like to thank all the people and friends who helped us in any way, and all the wonderful teachers and classmates we met on this educational journey.

Table of content

List of Figures

Introduction

Over time, people have managed to accomplish difficult tasks in various areas of life, especially in computer science, due to the revolution of technologies and their impact on society, enabling them to connect with each other across geographical boundaries and acquire knowledge through the internet.

In the developer community in particular, the nature of their work requires familiarity with many fields and technologies, making it challenging to complete projects independently. Given the interweaving of multiple technologies in modern development, cooperation becomes essential, as developers often need assistance with specific fields or tools.

In this perspective, our thesis aims to propose a process development tool of web application that enhances Computer Science Advancements through a Skill Exchange Platform, designed to facilitate the exchange of skills online.

the concept of skill exchange is built upon the individuals sharing their knowledge in a certain domain of computer science allowing them to use their abilities as an alternative way to access and learn new skills from other people in the community of the developers, additionally enabling people to request assistance in a specific service where they need help, in exchange for payment, that approach allows the community developers to grow over time.

In this thesis, there is three chapters under discuss, the first chapter is about the theoretical side which touches with the operation of skill exchanging and making skill requests.

the second chapter is about the general structure of this application and showing the design of the system and the UML diagrams.

the third chapter focuses on the implementation of the application, showcasing screen captures and detailing the technologies utilized in its developmen.

Chapter I. System Analysis

1. Introduction

This chapter covers the operational processes involved to achieve the primary goal of developing this software, the intended users of the software, and the benefits introduced to the community.

2. Presentation of the Project

2.1. Object of the Project

- The project's main goal is to build a service barter system that facilitates the exchange of skills and knowledge among users.
- Individuals can offer their skills and services in one area and, in return, receive assistance or lessons in another skill they want to learn.
- Here, skills are considered as the user’s payment method.
- The platform allows users to create skill request within a specific domain of knowledge, other users can make offers on that request for an amount of money, allowing the user who made the request to select the offer that suits their needs.

2.2. The contribution of the project

- Enhance learning from the internet.
- Advancement in programming knowledge.
- Establish a scalable community of developers.
- The platform allows users to easily meet in one place.

3. Structure of the Project

3.1. Individuals

- **Platform team**: is the creators of the websites and the individuals who can manage the website.
- **Users**: they are the individuals who make skill exchange and skill offers inside the platform.

3.2. Requirements

- Website will have categories (websites, gaming developing …).
- Create listening to his skills can offer by filling out the service information form.
- Can upload a short video to his service to explain it.
- Could exchange skill through the platform from the skill listing's that is already created.
- Have a profile with their own info and services.
- When user offer skill exchange he will provide all of his skills.
- The receiver user has the option to accept and refuse the offer and in case he accepts he picks the times the sender user chose.
- When exchange skills happen a messaging box appears for both users the sender and the receiver to talk and work.
- Exchanging skills happens by real time messages between the two of users.
- Make skill requests.
- Submitting an offer for a skill request in exchange for a certain amount.
- The user who made the skill request will receive not only one offer but many offers and he picks the offer they like.
- When the user who made skill request accept one offer the others will automatically got refused.
- Users can upload their feedbacks.

4. Knowledge and services sharing platforms

4.1. An entrance to knowledge sharing websites

Knowledge sharing is an activity through which knowledge, namely, information, skills, or expertise, is exchanged between people, friends, families, communities or organizations. Online knowledge sharing activities are flourishing with the advent of social media and digital life [1].

With the continuous development of information technology, knowledge sharing platforms (KSPs) have ushered in prosperity in recent years and attracted the widespread attention of consumers. KSPs are online communities where users can share, sell, and buy knowledge [2].

Social media environments offer unique features, including openness, two-way communication, and open-ended feedback. These characteristics have made it possible for large numbers of people to freely and easily share their thoughts, opinions, experiences, perspectives, information, and knowledge through social media [3].

4.2. Examples of real-world knowledge sharing platforms

A large number of knowledges sharing economic platforms, such as Skillshare, Quora (launching a new function named knowledge prize), and Zhihu, is emerging in the global market and exploring sustainable business models. In summary, there are four main business models in the knowledge sharing: Paid Subscriptions, Q&A Services, Public Consulting, and "Live Sessions" [4], and there are more platforms like reddit, Stack Overflow.

4.3. Problems and challenges

Most web applications and blogs cannot be helpful to get the specific information you need and It makes the user distracted and confused, that back to many factors, one of them is the lack of friendly user interfaces.

4.4. Value of our platform

Our website aims to facilitate the exchange of knowledge by designing a software interface that is both simple and flexible, and not overly detailed.

You can access to the platform full content without distractions, our website is designed to cover all categories of users, including people who are willing to invest money in requesting consultations and paid services, as well as individuals looking for learning through the exchange of knowledge in one platform.

5. Web technologies

5.1. What is a web application?

A web application is software that runs in your web browser. Businesses have to exchange information and deliver services remotely. They use web applications to connect with customers conveniently and securely. The most common website features like shopping carts, product search and filtering, instant messaging, and social media newsfeeds are web applications in their design. They allow you to access complex functionality without installing or configuring software [5].

5.2. Web app Architecture

5.2.1. Overview of architecture web app architecture

In order to facilitate this complex flow of data, web applications are usually designed with different layers. The most common design paradigm is a three-layered design consisting of a presentation layer (web browser), application layer (server), and storage layer (database). In this system, the presentation layer is responsible for relaying user data to the application layer, which can process that data and do any number of things, including passing it to the storage layer for "safe-keeping."[6].

Many times, web applications can grow to be very complex. In these cases, a three-layered design may fall short. This may necessitate the introduction of additional layers to handle this complexity. For instance, the introduction of an integration layer between the application and storage layers can help provide a uniform interface for data access, allowing the application layer to be insulated from changes that occur to the storage layer implementation [6].

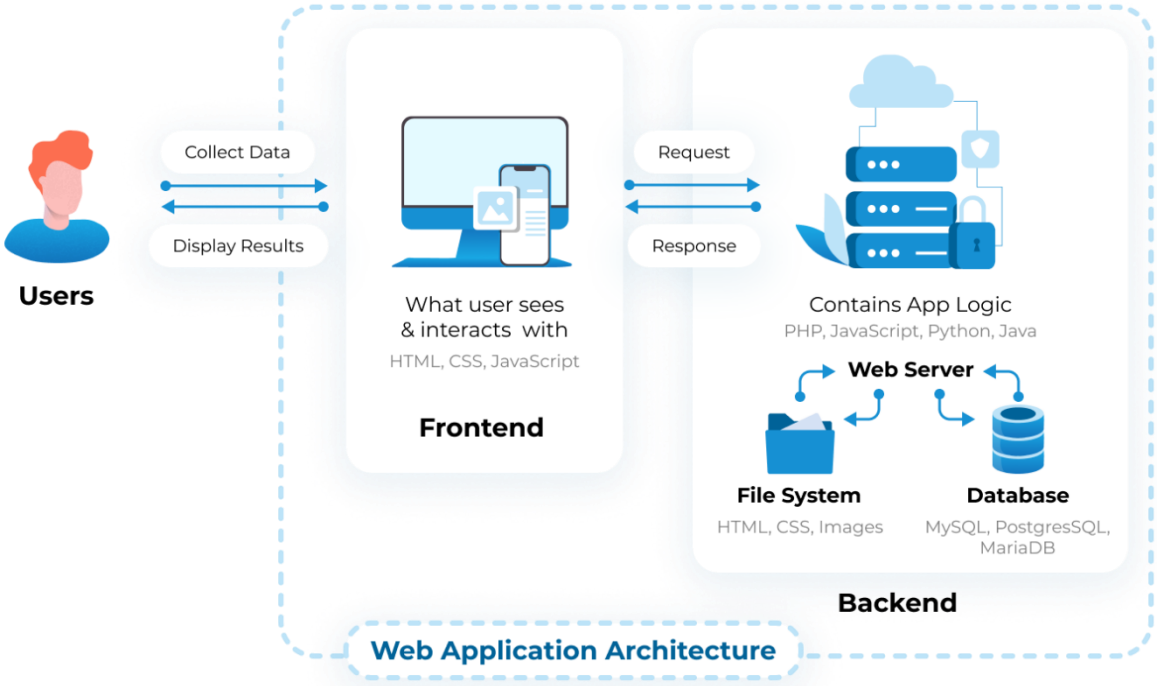

Figure 1. Web Application Architecture [7]

5.2.2. MVC

(Model-View-Controller) is a pattern in software design commonly used to implement user interfaces, data, and controlling logic. It emphasizes a separation between the software's business logic and display. This "separation of concerns" provides for a better division of labor and improved maintenance. Some other design patterns are based on MVC, such as MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter), and MVW (Model-View-Whatever) [8].

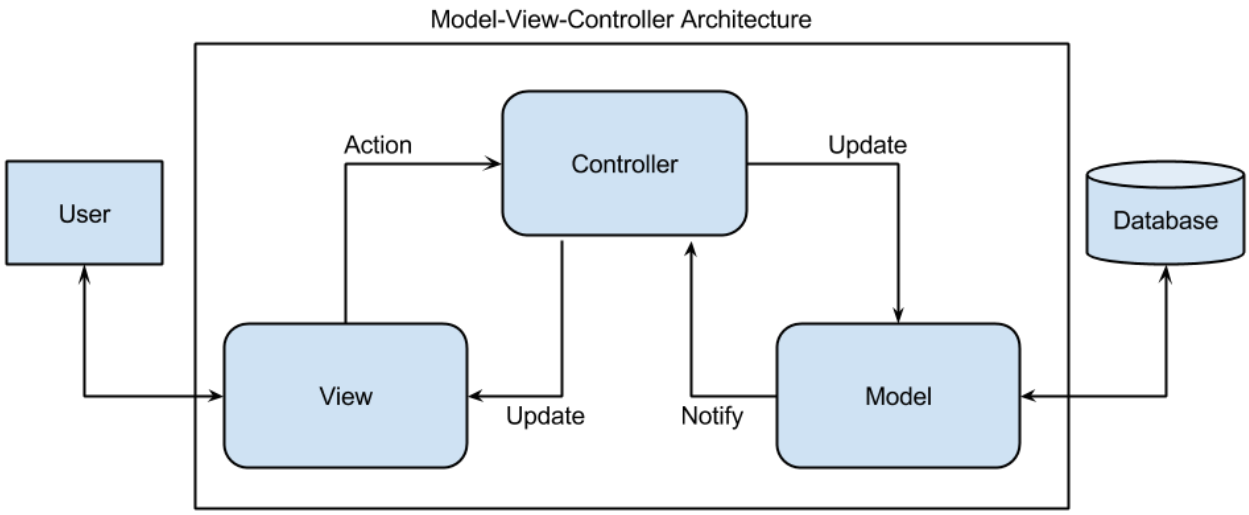

Figure 2. Model View Controller Architecture [9]

5.2.3. Rest API

REST, or REpresentational State Transfer, is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. REST-compliant systems, often called RESTful systems, are characterized by how they are stateless and separate the concerns of client and server. We will go into what these terms mean and why they are beneficial characteristics for services on the Web. Pay close attention: If you're looking for a career in tech, you may be asked to define rest during an interview [10].

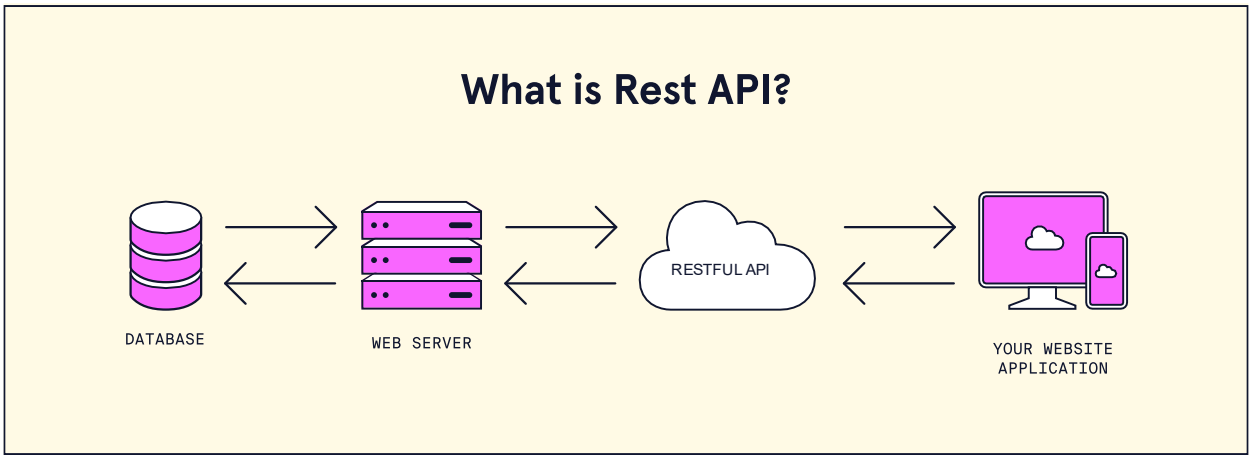

Figure 3. What is Rest API? [10]

5.3. Types are web application

5.3.1. Static web apps

This is the most basic kind of web app. It typically displays information without any fanfare and with limited content. It's built with HTML and CSS and is not very customizable, nor is it flexible. One advantage of static web applications is that they function well in offline mode [11].

5.3.2. Dynamic web application

Meanwhile, dynamic web applications are more sophisticated than static web apps, offering interactive content and generating data in real time in response to actions or requests from users [11].

5.3.3. Single-page Web application (SPA)

An SPA is a dynamic web application that functions more quickly than the standard web app, performing logic within the browser, as opposed to the server. This offers a better navigation experience for the user [11].

5.3.4. Multi-page Web Application

Multi-page web apps are a more traditional approach in that they allow navigation to different pages and reload, as well as display different pages, with every new user action. One benefit is that every page in the web app can be optimized for SEO [11].

5.3.5. Progressive Web Application

Progressive web apps are becoming increasingly ubiquitous. These responsive apps look, feel, and behave like native mobile apps, but they don't need to be downloaded or installed. They also work in both online and offline modes [11].

5.4. The difference between a web application and a website?

When the internet was newly invented, websites had significantly less functionality than web apps. They were only capable of delivering information to users through static content. You had to install and run software with complex functionality. Web apps were built to bridge the gap between software and static sites. They had functionality and interactive user elements like the software but were delivered using a web browser URL [5].

However, web technology has evolved significantly since then. Most modern websites are complex web applications in their design [5].

5.5. Web Application Development Process

The web application development process involves several phases, each with its own set of tasks, activities, and considerations. While the specific steps may vary depending on the project scope, technology stack, and team structure, here is a general outline of the web application development process [12].

5.5.1. Requirement Gathering and Analysis:

- Understand the project's goals, target audience, and business requirements [12].
- Define functional and non-functional requirements for the web application [12].

5.5.2. Planning:

- Determine the technology stack, frameworks, and tools that will be used [12].
- Plan the application's architecture, including data models, components, and interactions [12].

5.5.3. Design:

- Design the user interface (UI) and user experience (UX) of the web application [12].
- Design database schemas, data flows, and system diagrams [12].

5.5.4. Front-End Development:

- Develop the client side of the web application using HTML, CSS, and JavaScript [12].

5.5.5. Back-End Development:

- Develop the server-side logic, business logic, and APIs using the chosen programming language and framework [12].

5.5.6. Database Development:

- Design and create the database schema based on the application's data requirements [12].

5.5.7. Integration:

- Integrate third-party services, APIs, and libraries that enhance the application's functionality (e.g., payment gateways, social media integration, geolocation services) [12].

5.5.8. Testing:

- Conduct various levels of testing, including unit testing, integration testing, and user acceptance testing [12].
- Test the application's functionality, performance, security, and compatibility across different browsers and devices [12].

5.5.9. Deployment:

- Deploy the web application to a production environment, such as a web server, cloud platform, or hosting service [12].

5.5.10. Maintenance and Updates:

- Regularly update and maintain the application, including bug fixes, security patches, and feature enhancements [12].

6. Conclusion

In this chapter we identified the general structure of the project, the system requirements, and an overview into the users' needs.

Chapter II. System Design

1. Introduction

This chapter covers the core components of software system design through the use of architecture diagrams and database design, these blueprints provide developers with a better perspective and vision of the platform, to achieve an efficient goal.

2. The modeling Unified Modeling Language (UML)

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems [13].

3. Diagrams

3.1. Use Case Diagram

A Use Case Diagram is a vital tool in system design, it provides a visual representation of how users interact with a system. It serves as a blueprint for understanding the functional requirements of a system from a user's perspective, aiding in the communication between stakeholders and guiding the development process [14]. see figure (4)

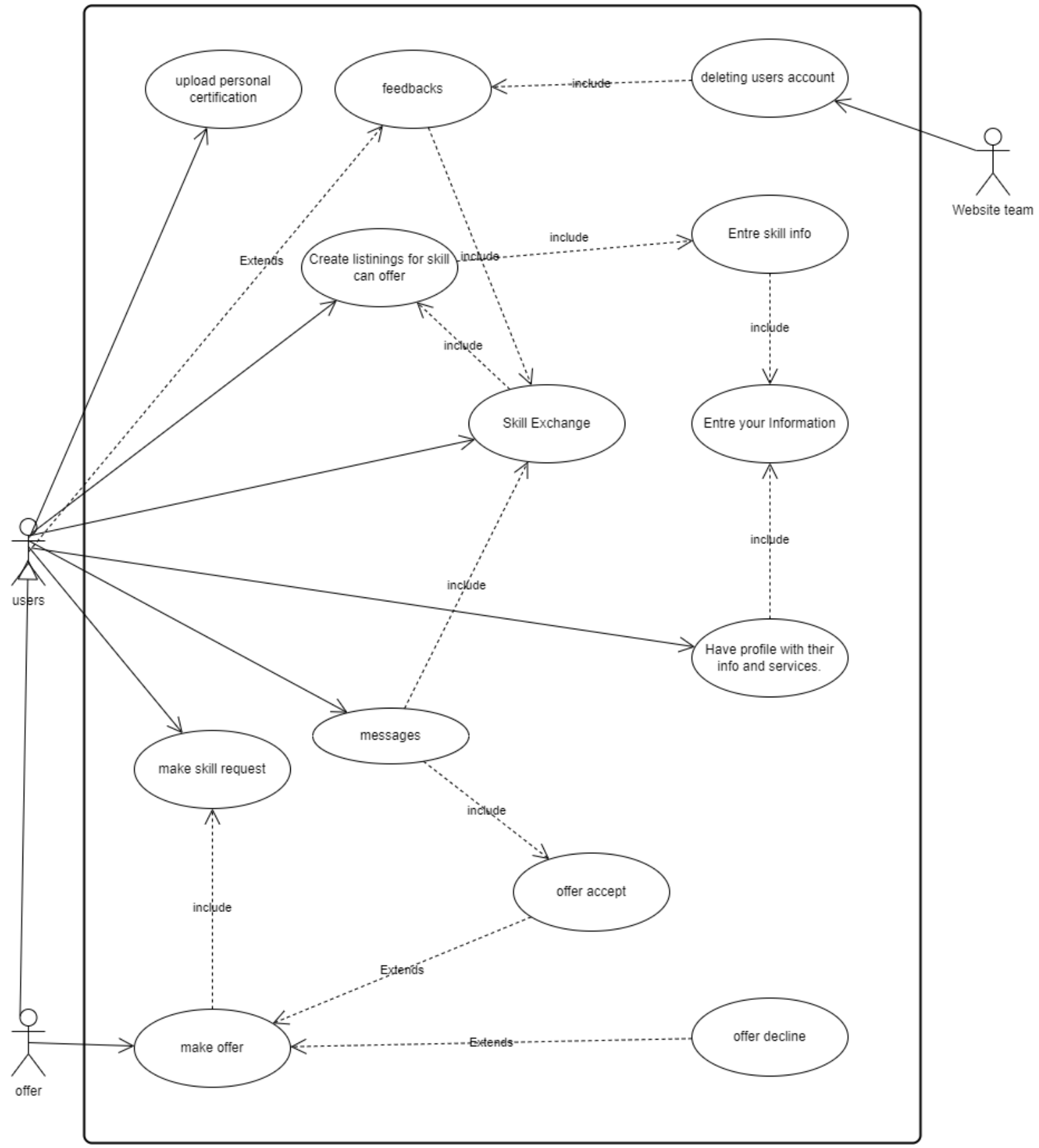

Figure 4. Use Case Diagram

3.2. Class Diagram

Class diagrams are one of the most useful types of diagrams in UML as they clearly map out the structure of a particular system by modeling its classes, attributes, operations, and relationships between objects. With our UML diagramming software, creating these diagrams is not as overwhelming as it might appear. This guide will show you how to understand, plan, and create your own class diagrams [15].

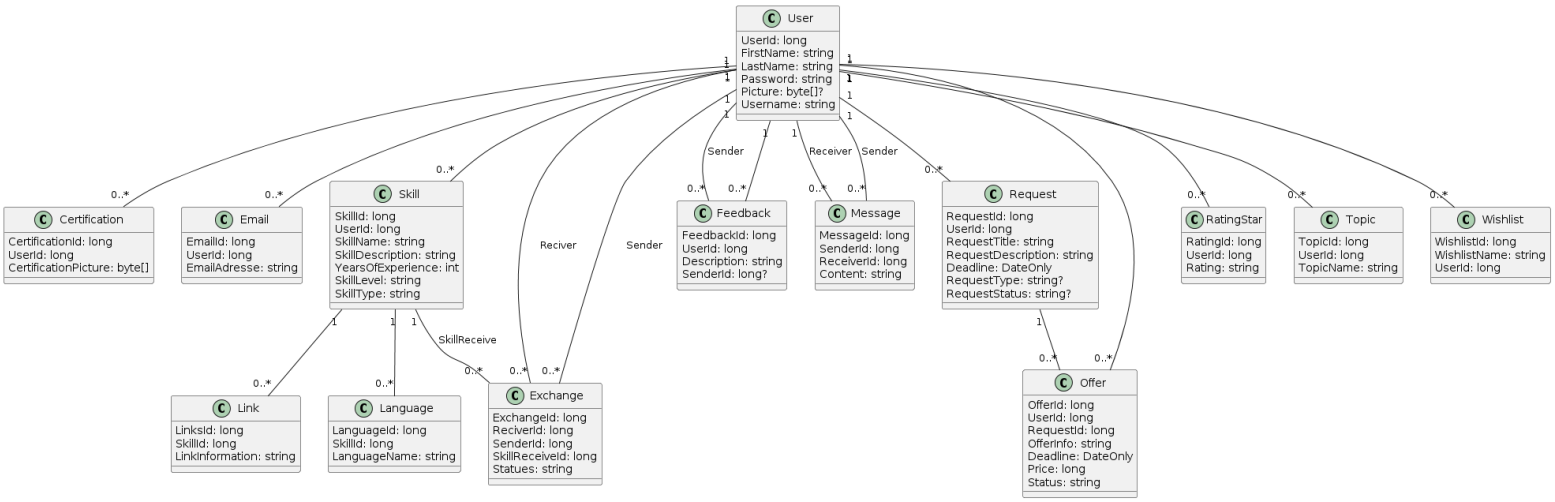

Figure 5. Class Diagram

3.3. Sequence Diagram

Sequence diagrams are a popular dynamic modeling solution in UML because they specifically focus on *lifelines*, or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends. Along with our UML diagramming tool, use this guide to learn everything there is to know about sequence diagrams in UML [16].

3.3.1. Create an account

The sequence diagram below Figure (6), represents the steps a user takes to register on the platform and the possible errors that can occur during the registration process.

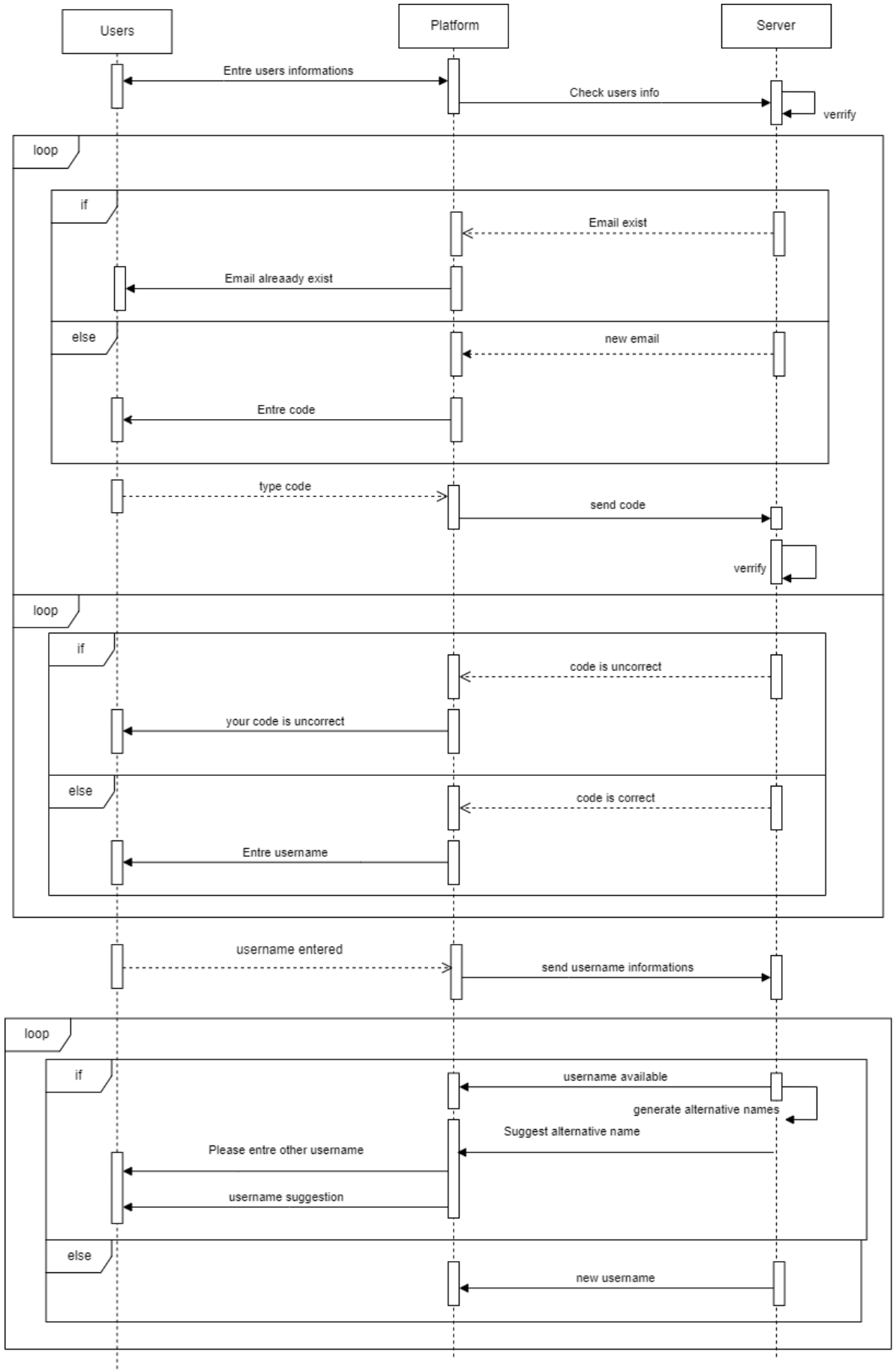

Figure 6. Sequence Diagram, registration

3.3.2. Skill exchange

The sequence diagram below Figure (7) represents how a skill exchange can occur within the platform and the possible paths it can happen.

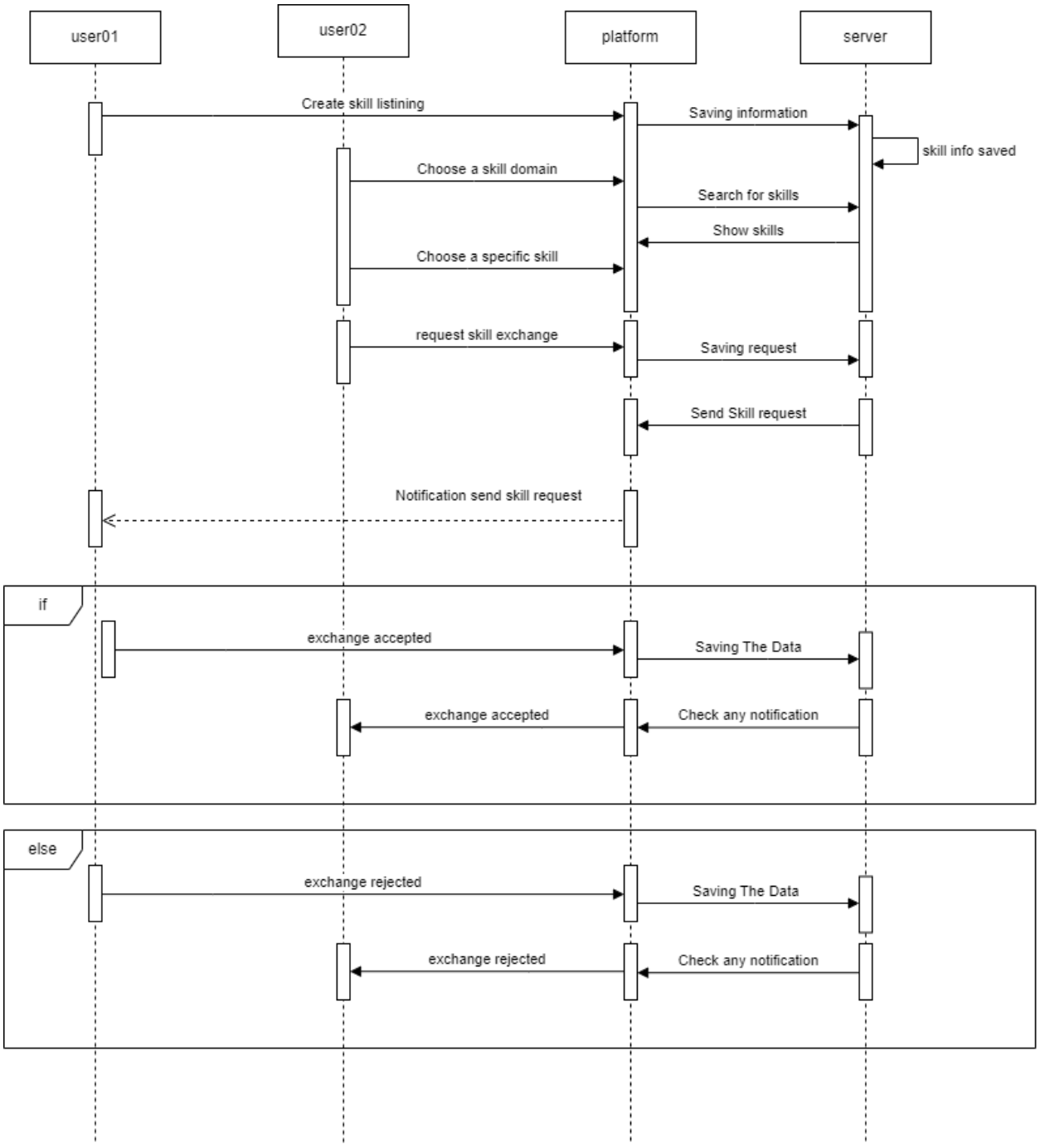

Figure 7. Sequence diagram, Skill Exchange

3.3.3. Skill offer

the sequence diagram below Figure (8) represents how users can make offers, the interactions they take from them, and their possible responses to those requests.

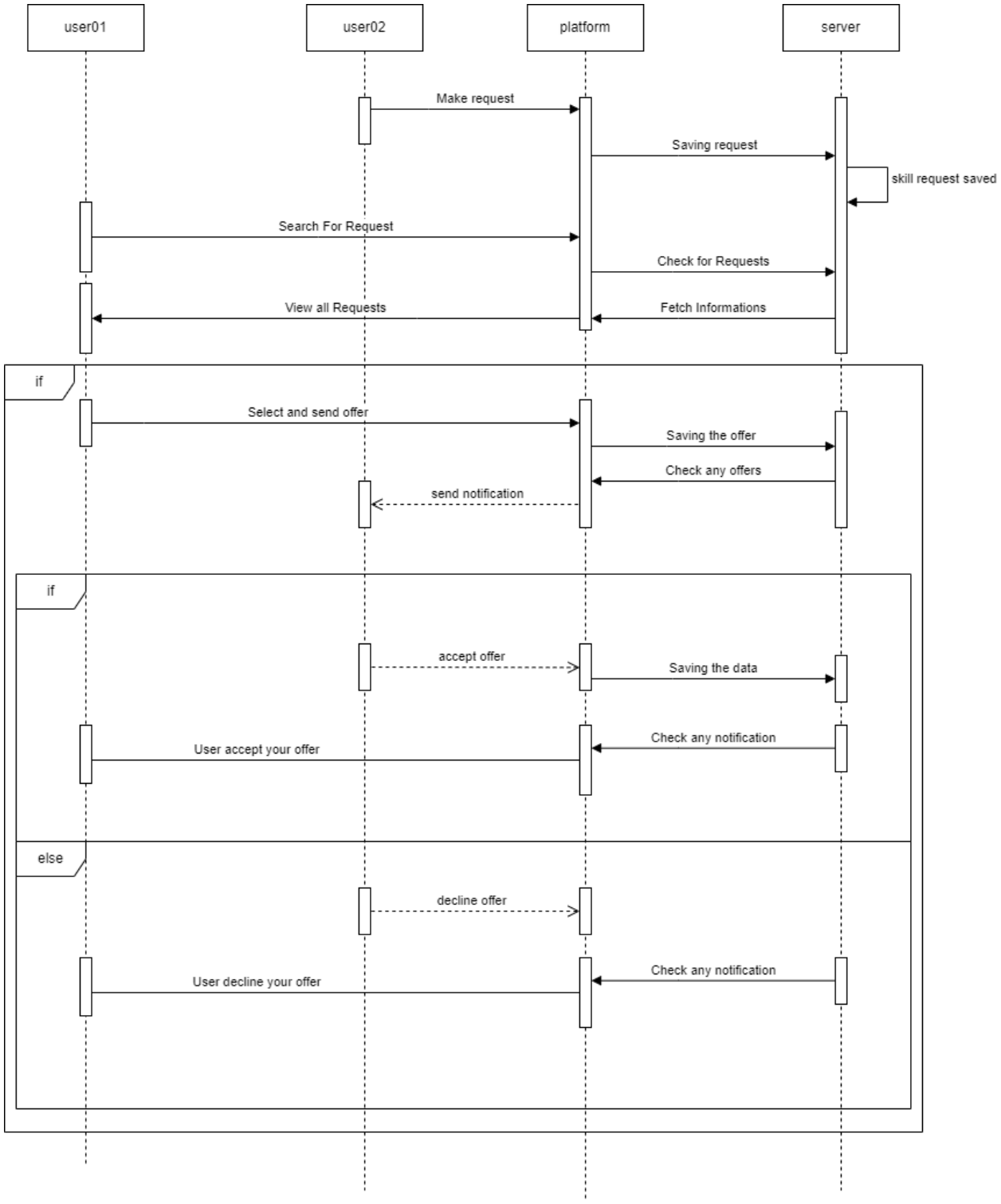

Figure 8. Sequence diagram, Skill Offer

4. Conclusion

In conclusion, utilizing architectural tools such as UML and database designs can aid developers to better understanding of the behaviors of complex systems, simplifying them to make them more adaptable for average users. A well-designed architecture ensures data integrity and facilitates the development of efficient software.

Chapter III. Implementation

1. Introduction

This chapter discusses the implementation part and presents the most important technologies used in this project and how they were used, along with displaying some screenshots of the application.

2. Development Environment

2.1. Technologies and Programming Languages

2.1.1. .NET

.NET is a secure, reliable, and high-performance application platform [17].

Figure 9. .NET logo

2.1.2. C#

C# is the programming language for .NET. It's strongly-typed and type-safe and has integrated concurrency and automatic memory management [17].

Figure 10. C# Programming Language Logo

2.1.3. JavaScript

JavaScript (JS) is a lightweight interpreted (or just-in-time compiled) programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles [18].

Figure 11. JavaScript Programming Language

2.1.4. HTML

HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behavior (JavaScript) [19].

Figure 12. HTML Logo

2.1.5. CSS

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media [20].

Figure 13. CSS Logo

2.2. Database Management System

2.2.1. PostgreSQL

PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance [21].

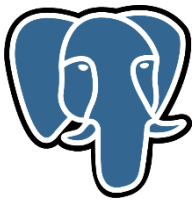

Figure 14. PostgreSQL Logo

2.2.2. PgAdmin4

pgAdmin is the leading Open Source management tool for Postgres, the world's most advanced Open Source database. pgAdmin 4 is designed to meet the needs of both novice and experienced Postgres users alike, providing a powerful graphical interface that simplifies the creation, maintenance and use of database objects [22], Create the database for this project see Figure (15).

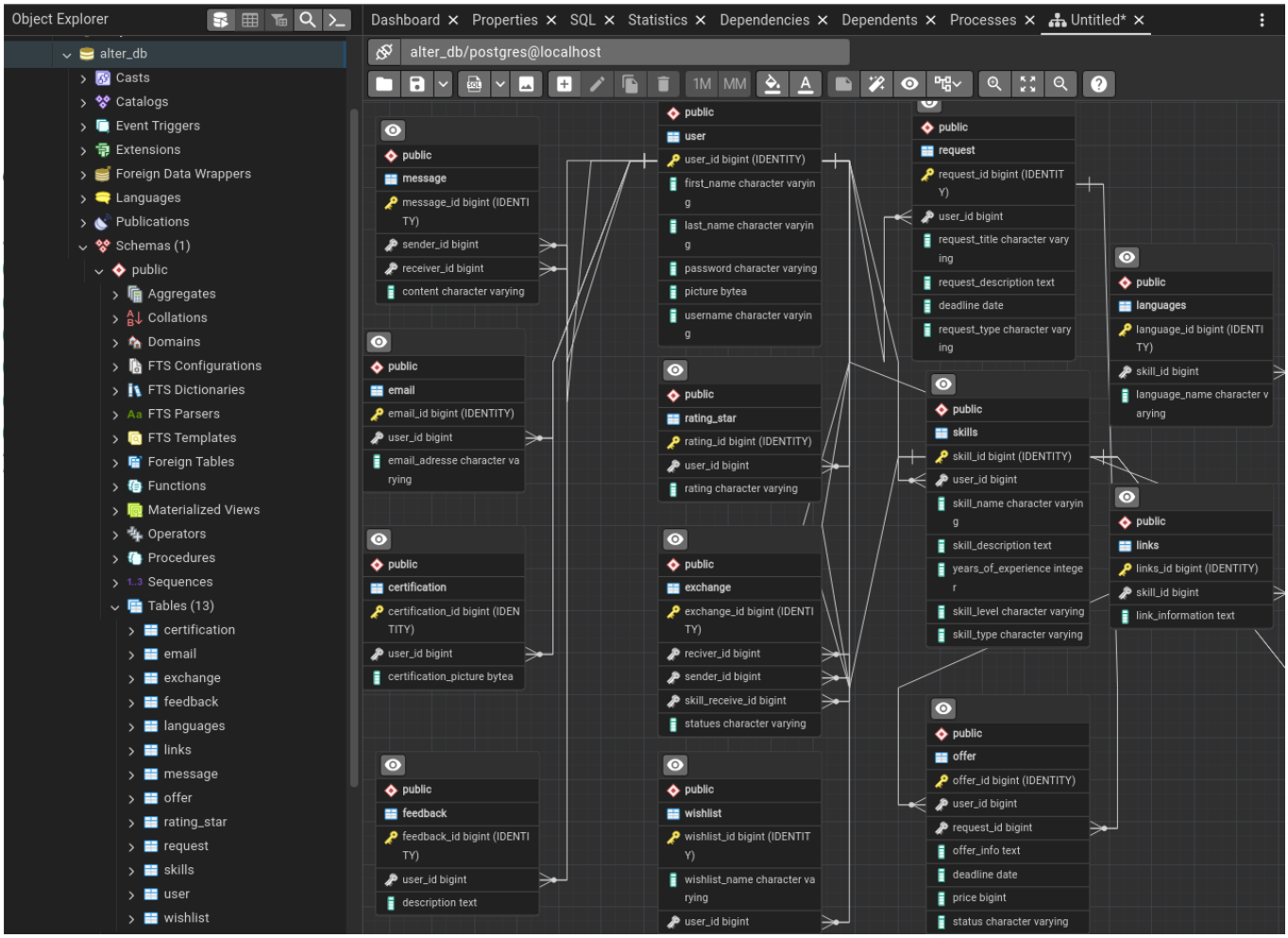

Figure 15. Database, pgAdmin GUI

2.3. Frameworks and Libraries

2.3.1. ASP.NET CORE

ASP.NET Core extends the .NET developer platform with tools and libraries specifically for building web apps [23].

Figure 16. ASP.NET CORE logo

2.3.2. Entity Framework

Entity Framework is a modern object-relation mapper that lets you build a clean, portable, and high-level data access layer with .NET (C#) across a variety of databases, including SQL Database (on-premises and Azure), SQLite, MySQL, PostgreSQL, and Azure Cosmos DB. It supports LINQ queries, change tracking, updates, and schema migrations [24].

Figure 17. Entity Framework logo

2.3.3. ReactJS

React is a JavaScript library for rendering user interfaces (UI). UI is built from small units like buttons, text, and images. React lets you combine them into reusable, nestable components. From web sites to phone apps, everything on the screen can be broken down into components. In this chapter, you'll learn to create, customize, and conditionally display React components [25].

Figure 18. ReactJS logo

2.3.4. Axios

Axios is a promise-based HTTP Client for node.js and the browser. It is isomorphic (= it can run in the browser and nodejs with the same codebase). On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequests [26].

Here, we send an HTTP request to an endpoint that is responsible for checking whether the email already exists or not. If there is no email like the one we wrote in our input, the HTTP response will be 200, which means OK, that will allow us to go to the next step in our multi-form registration, but if a user types an already existing email, the response will be 400 which means Bad Request, and user won't be able to continue the process of registration see Figure (19).

```
const nextStep = async () => {
  if (!handlePasswordValidation()) {
    return;
  }
  if (step === 1) {
    const emailData = { email: formData.Email };
    try {
      console.log('Sending email verification request with data:', emailData);
      const emailValidation = await axios.post('http://localhost:5105/api/User/checkEmail', emailData);
      console.log('Email validation response:', emailValidation);

      if (emailValidation.status === 200 || emailValidation.status === 201) {
        setStep(prevStep => prevStep + 1);
      } else if (emailValidation.status === 400) {
        setEmailVerificationError("Email is already taken");
      }
    } catch (error) {
      console.error('Error checking email:', error);
      setEmailVerificationError("Error checking email");
    }
  }
};
```

Figure 19. Handle HTTP Request/Response with Axios

2.3.5. SignalR

ASP.NET SignalR is a library for ASP.NET developers that simplifies the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data [27].

Figure 20. SignalR logo

2.3.6. Bootstrap

Bootstrap is a free and open-source web development framework. It's designed to ease the web development process of responsive, mobile-first websites by providing a collection of syntax for template designs [28].

Figure 21. Bootstrap logo

2.4. Development Tools

2.4.1. Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET) [29].

Figure 22. Visual Studio Code logo

2.4.2. Postman

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster [30].

Figure 23. Postman logo

Here is a presentation of the method for testing this endpoint with Postman see Figure (24), which allows displaying all users with whom the process of successfully exchanging skills also with those whose offers were received and approved.

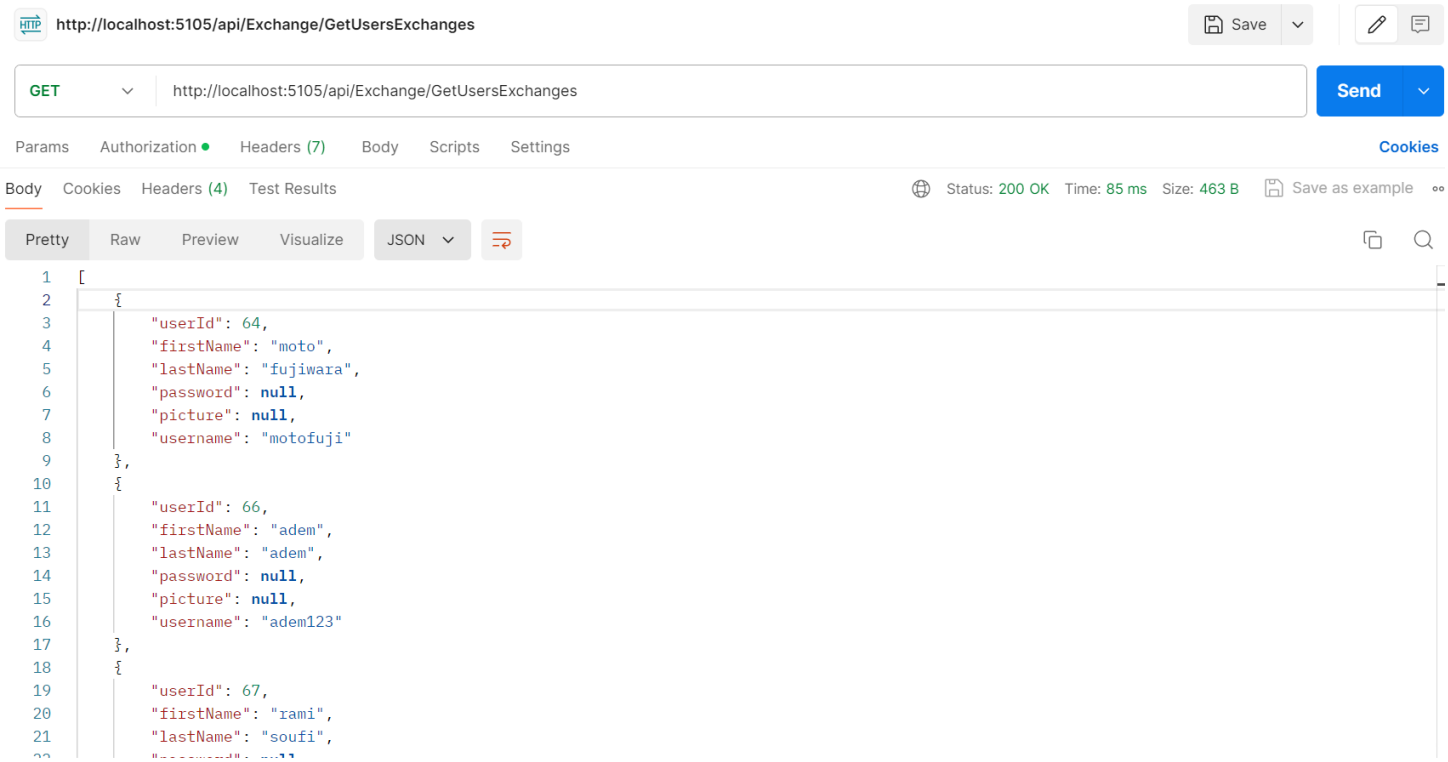

Figure 24. Postman interface

2.4.3. Swagger UI

Swagger UI allows anyone — be it your development team or your end consumers — to visualize and interact with the API's resources without having any of the implementation logic in place [31].

Figure 25. Swagger UI logo

Here is a screenshot of how to document endpoints programmed in the backend to automatically generated in SwaggerUI for use directly in the frontend.

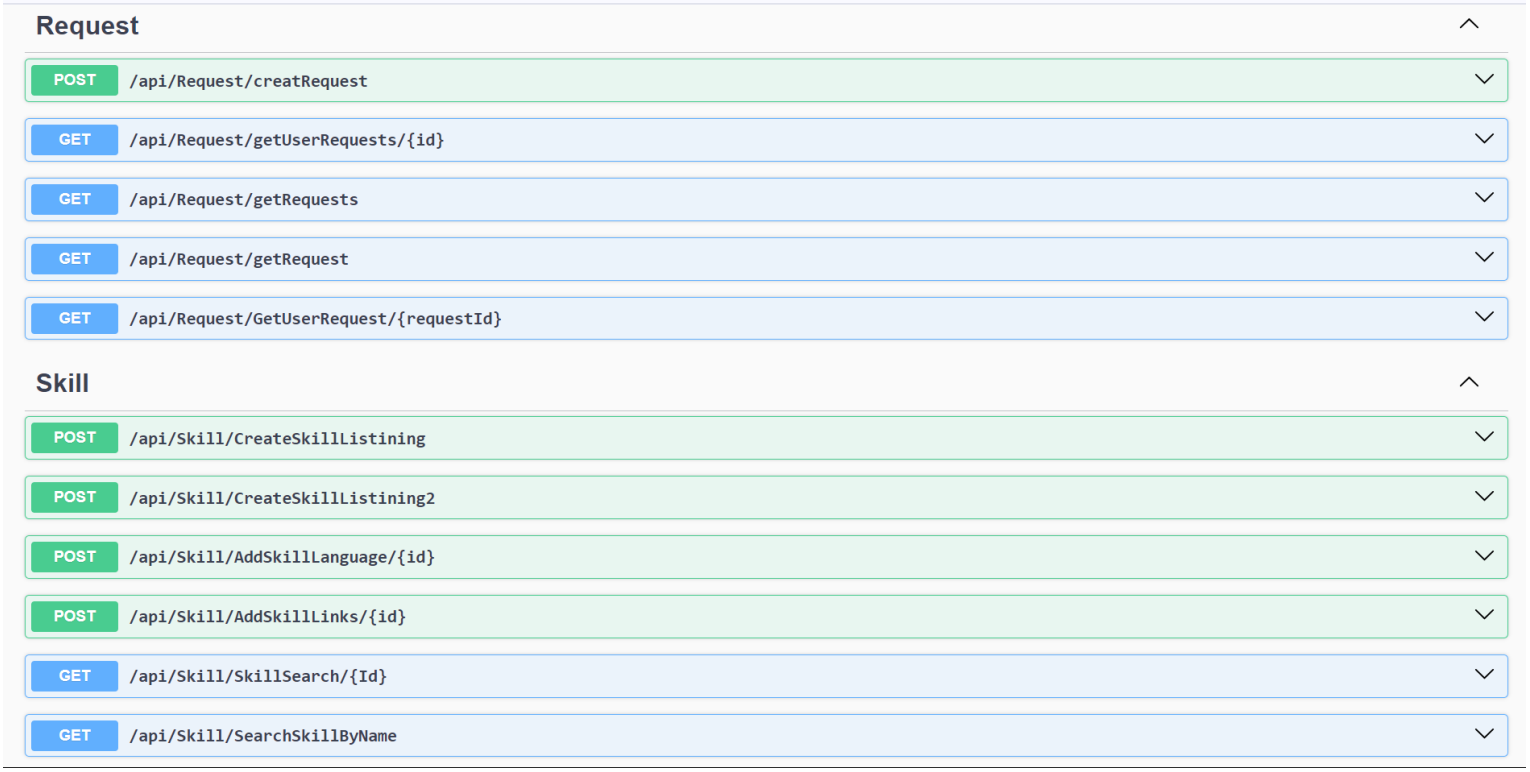

Figure 26. SwaggerUI

2.4.4. Figma

Figma is a free, online UI tool to create, collaborate, prototype, and handoff [32].

Figure 27. Figma logo

2.4.5. DrawIO

Draw.io is free online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams [33].

Figure 28. DrawIO logo

3. Security

3.1. JWT

3.1.1. What is JWT

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA [34].

3.1.2. When should you use JSON Web Tokens?

Here are some scenarios where JSON Web Tokens are useful:

- **Authorization**: This is the most common scenario for using JWT. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used across different domains [34].
- **Information Exchange**: JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed—for example, using public/private key pairs—you can be sure the senders are who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with [34].

3.1.3. JSON Web Token Structure?

In its compact form, JSON Web Tokens consist of three parts separated by dots (.), which are: [34]

- **Header**: The header *typically* consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA [34].
- **Payload**: The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims: *registered*, *public*, and *private* claims [34].
- **Signature**: To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that [34].

3.1.4. How JWT work?

In authentication, when the user successfully logs in using their credentials, a JSON Web Token will be returned. Since tokens are credentials, great care must be taken to prevent security issues. In general, you should not keep tokens longer than required [34].

Whenever the user wants to access a protected route or resource, the user agent should send the JWT, typically in the Authorization header using the Bearer schema. The content of the header should look like the following [34], see Figure (29):

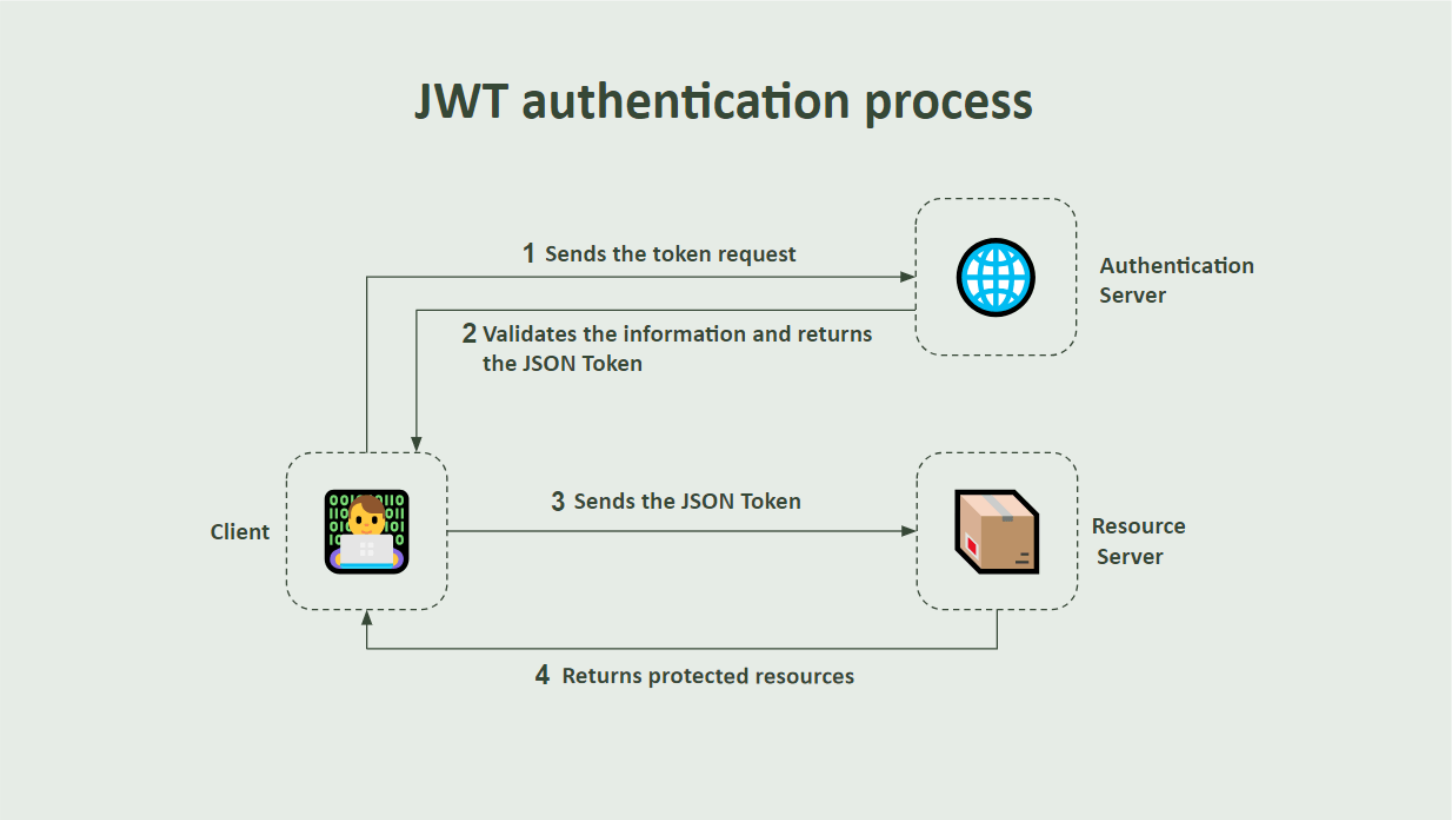

Figure 29. How JWT work [35]

3.1.5. Implement JWT in this project

Implementation steps in ASP.NET Core.

1. JWT Services Configuration.

```
// Add Authentication services
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = builder.Configuration["Jwt:Issuer"],
            ValidAudience = builder.Configuration["Jwt:Audience"],
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"] ?? "")),
            ClockSkew = TimeSpan.Zero
        };
    });

var app = builder.Build();
```

Figure 30. JWT Services Configuration

2. JWT Application Configuration, this property holds the secret signing key used to sign and verify JWT tokens.

```
{} appsettings.json > ...
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "Jwt": {
    "Key": "BAKACHWIKLERFISTERGOOKKKJJJJKKKKKJJJJJKKKKKKJJJJJKKKKJJJJKJKJKKJJKKJKKJKJKJ",
    "Issuer": "http://localhost:5105",
    "Audience": "http://localhost:5105"
  }
}
```

Figure 31. Store Secret Key

3. Generating the token method.

```
private string Generate(User user)
{
    var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"] ?? string.Empty));
    var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);
    var claims = new[] {
        new Claim(ClaimTypes.NameIdentifier, user.UserId.ToString()),
        new Claim(ClaimTypes.Name, user.Username)
    };
    var token = new JwtSecurityToken(_configuration["Jwt:Issuer"],
     _configuration["Jwt:Audience"],
      claims,
       expires: DateTime.Now.AddHours(30),
        signingCredentials: credentials);
    return new JwtSecurityTokenHandler().WriteToken(token);
}
```

Figure 32. Generate Token method

4. User authentication method.

```
private User Auth(LoginDto dto)
{
    var user = _db.Users.FirstOrDefault(u => u.Username == dto.Username);
    var passwordHasher = new PasswordHasher<User>();
    if (user != null)
    {
        if (passwordHasher.VerifyHashedPassword(user, user.Password, dto.Password) == PasswordVerificationResult.Success)
        {
            return user;
        }
    }
    return null!;
}
```

Figure 33. Authentication Method

5. Function GetCurrentUser that retrieves information about the currently logged-in user.

```
private User GetCurrentUser()
{
    var Identity = HttpContext.User.Identity as ClaimsIdentity;
    if (Identity != null)
    {
        var userClaim = Identity.Claims;
        return new User
        {
            UserId = Convert.ToInt64(userClaim.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier)?.Value),
            Username = userClaim.FirstOrDefault(c => c.Type == ClaimTypes.Name)?.Value ?? string.Empty
        };
    }
    return null!; // Add a return statement for the case when Identity is null
}
```

Figure 34. How JWT work [35]

6. Example of how to implement generate token method in login.

```
[AllowAnonymous]
[HttpPost("login")]
0 references
public IActionResult Login2([FromBody] LoginDto dto)
{
    var tokenDto = new TokenDto();
    var user = Auth(dto);
    if (user != null)
    {
        var token = Generate(user);
        tokenDto.Token = token;
        return Ok(tokenDto);
    }
    return NotFound("Invalid username or password");
}
```

Figure 35. Login Endpoint

7. Test Login endpoint API that return token see Figure (36).

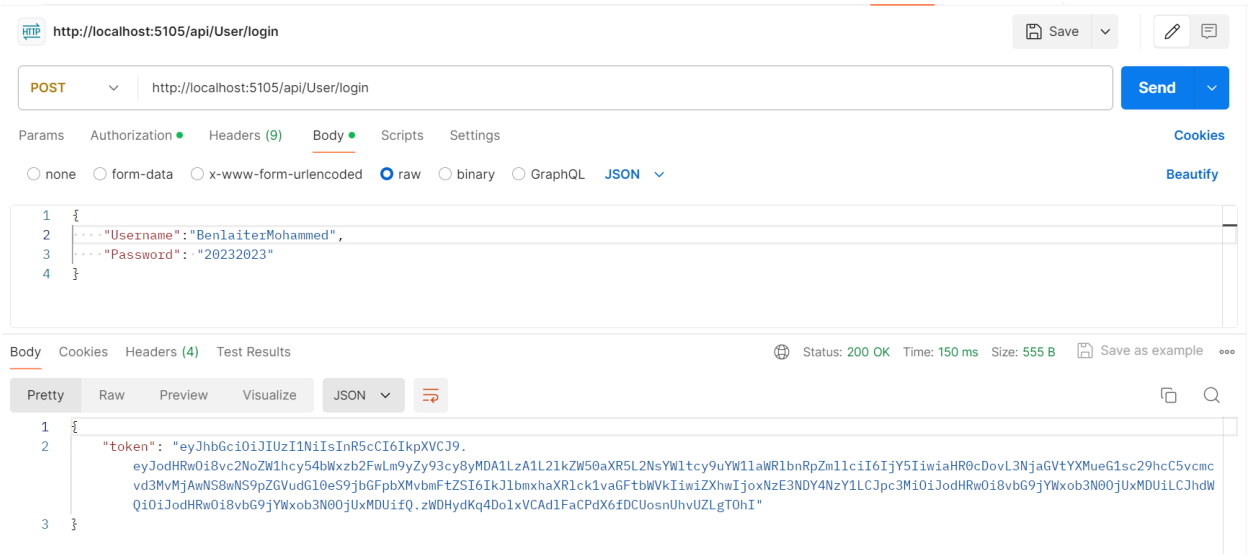

Figure 36. Login Endpoint Test

4. Presentation of Project Interfaces

4.1. Home Page

It is the initial page that appears to users for the first time, see Figure (37).

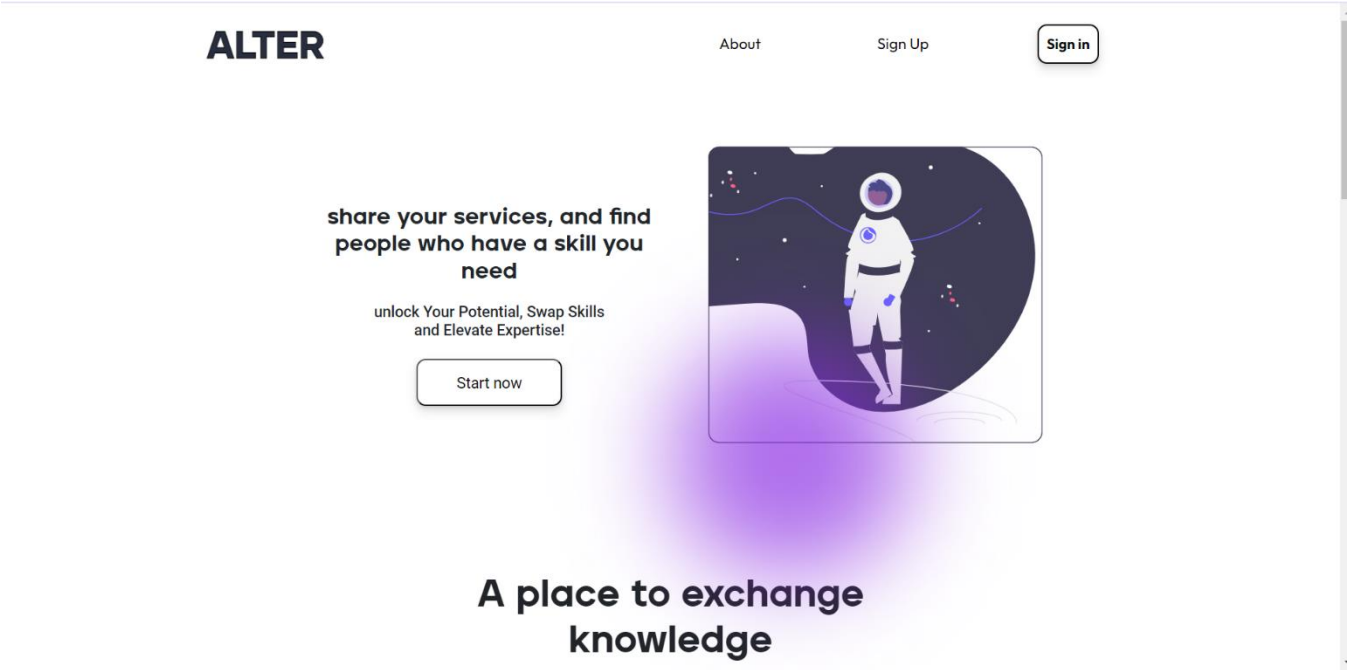

Figure 37. Home Page

4.2. Sign Up

The interface presents the feature to create a new account, where the user fills in all their information, see Figure (38).

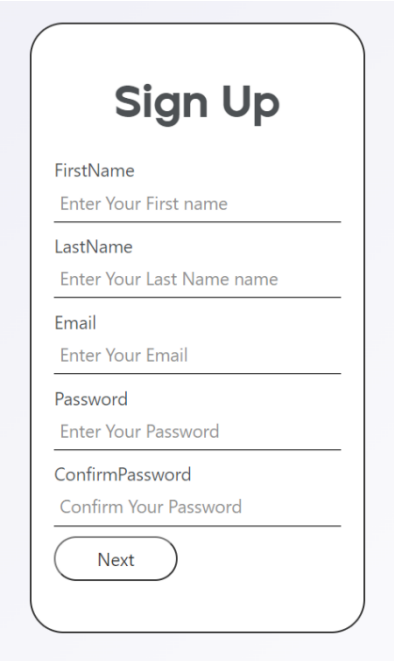

Figure 38. Sign Up

4.3. Create Wishlist

After the user fills out and creates an account here on this page Figure (39), he can fill out a list of the skills he would like to learn in the future.

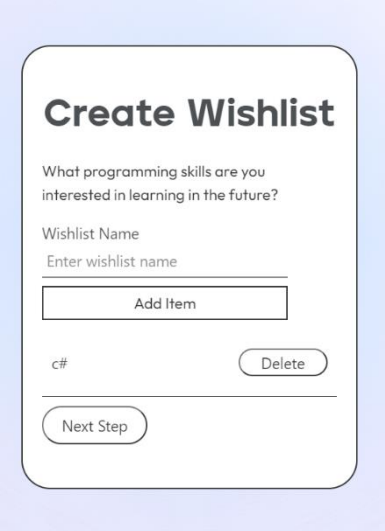

Figure 39. Wishlist

4.4. Sign in

This page displays the login process. see Figure (40).

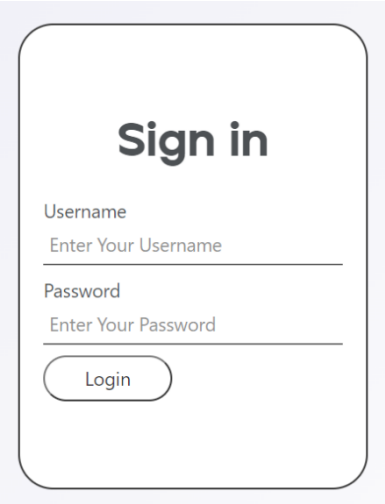

Figure 40. Sign in

4.5. Main Page

This is the main page, which can display two sections, view all skills by filtering based on skill domain, or view all the requests. Users can also directly search for skills. see Figure (41).

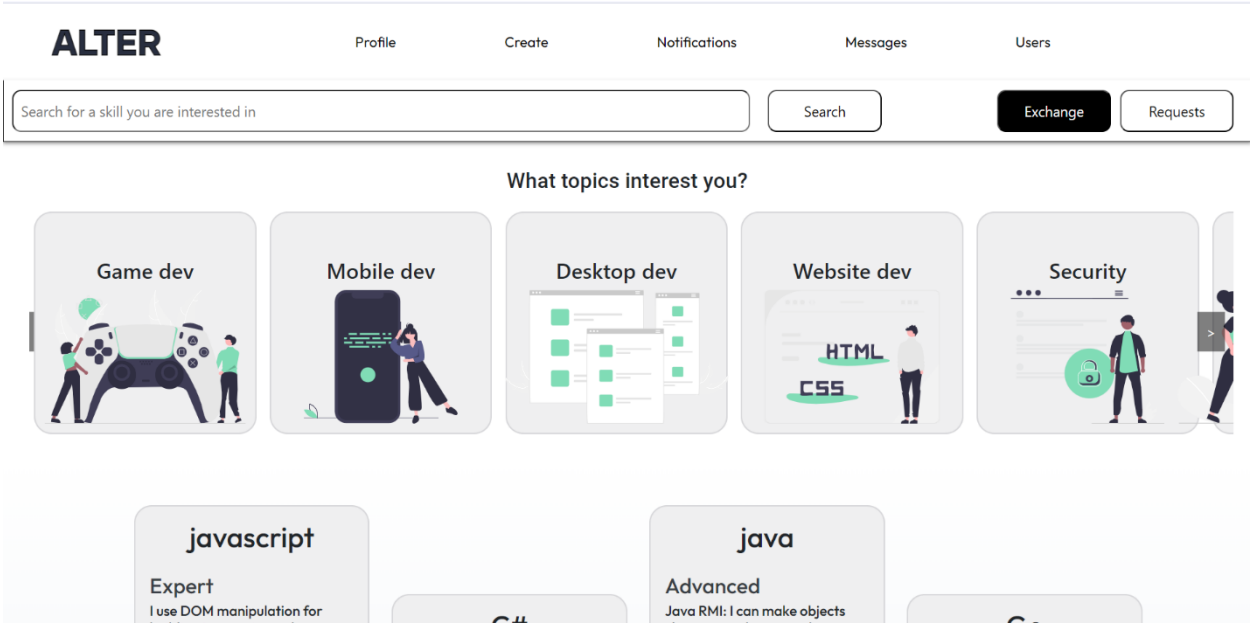

Figure 41. Main Page

4.5.1. Skills

After choosing the domain of the skills, the skills will be displayed based on their domain of knowledge follows see Figure (42).

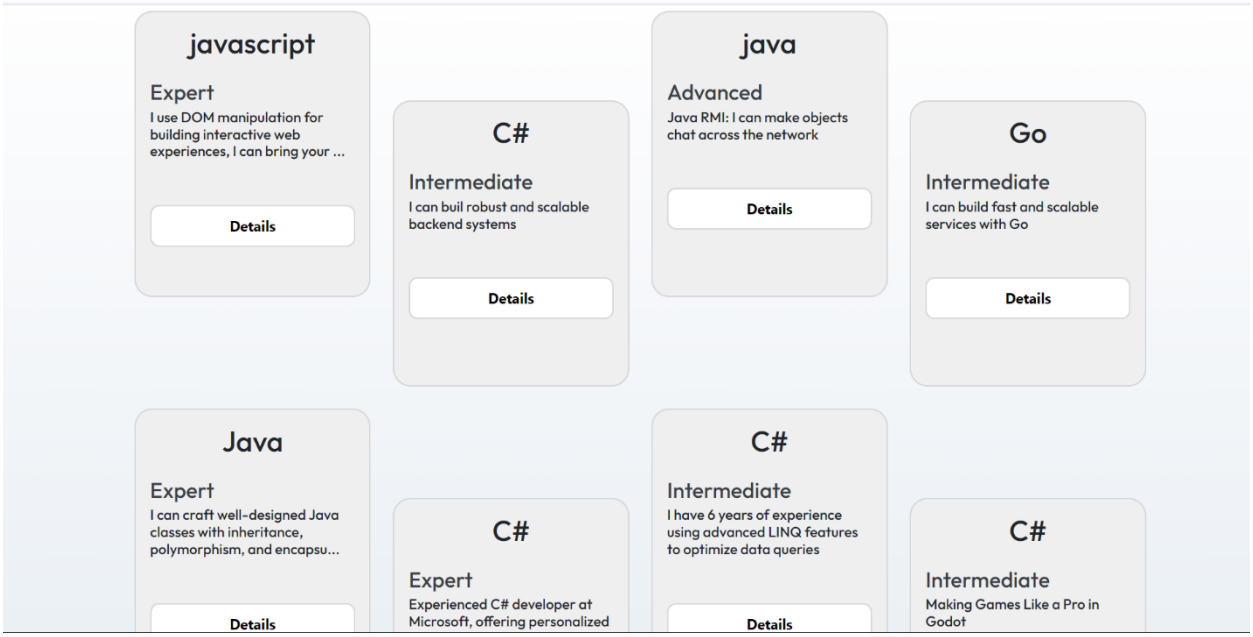

Figure 42. Skills Section

4.5.2. Apply skill exchange

After clicking on the details button in the previous Figure (42), this page appears when the user wants to submit an exchange request to that particular user, Figure (43) and Figure (44).

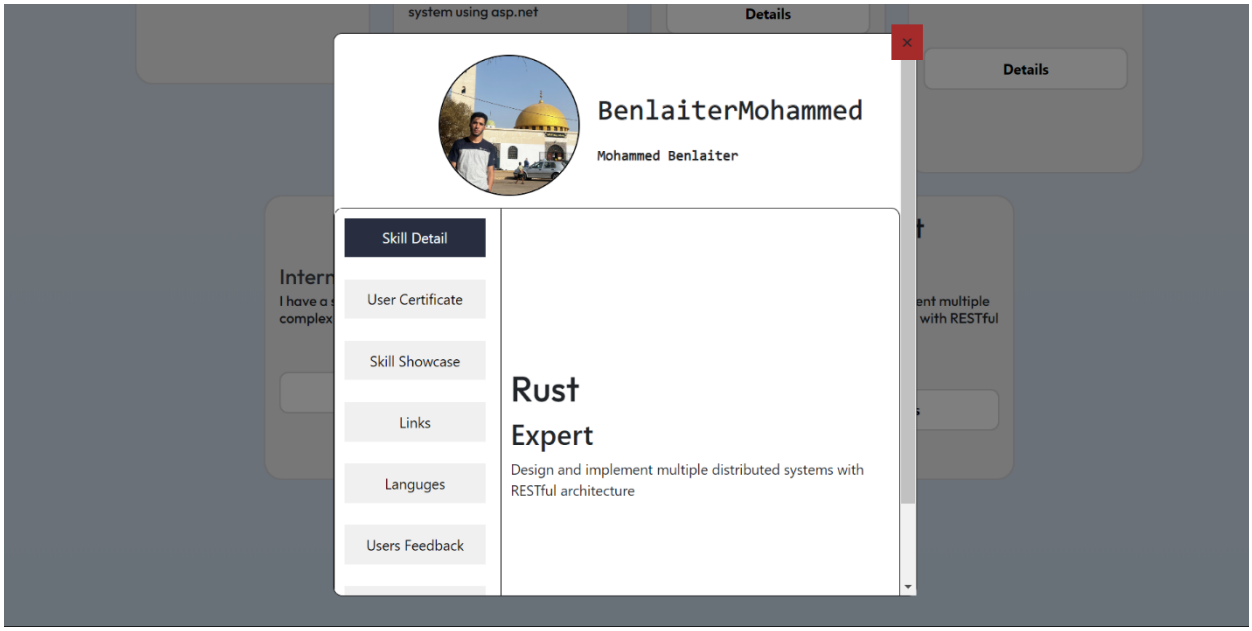

Figure 43. Skill Detail 1

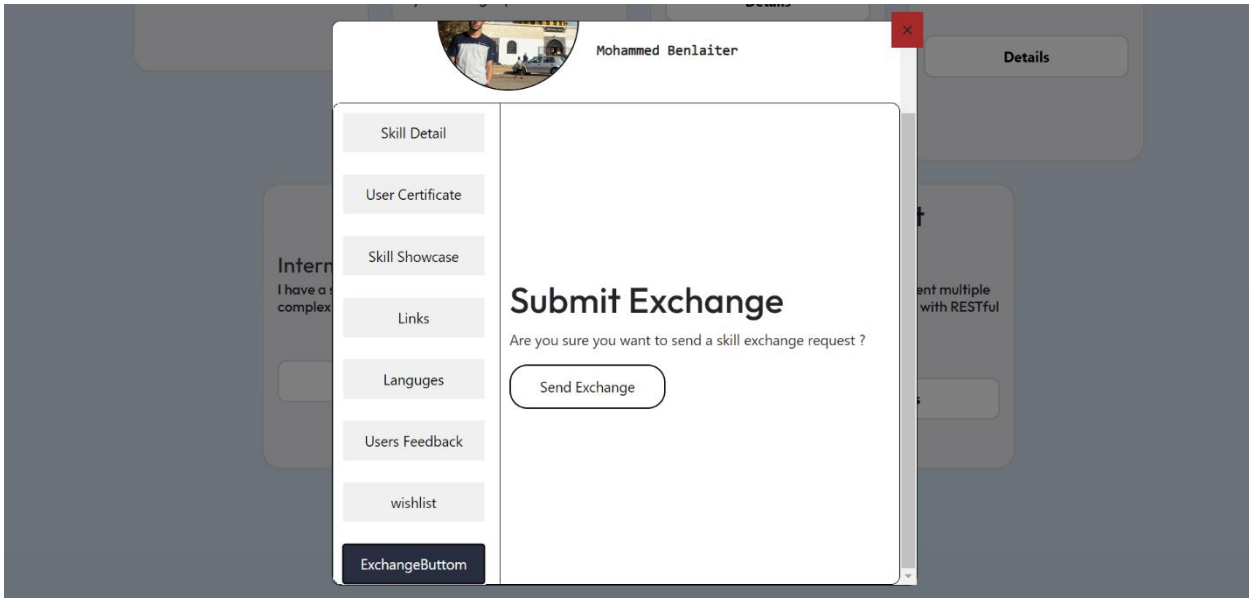

Figure 44. Skill Detail 2

4.5.3. Requests

After choosing the requests section, the available requests will be displayed for the users follows see Figure (45).

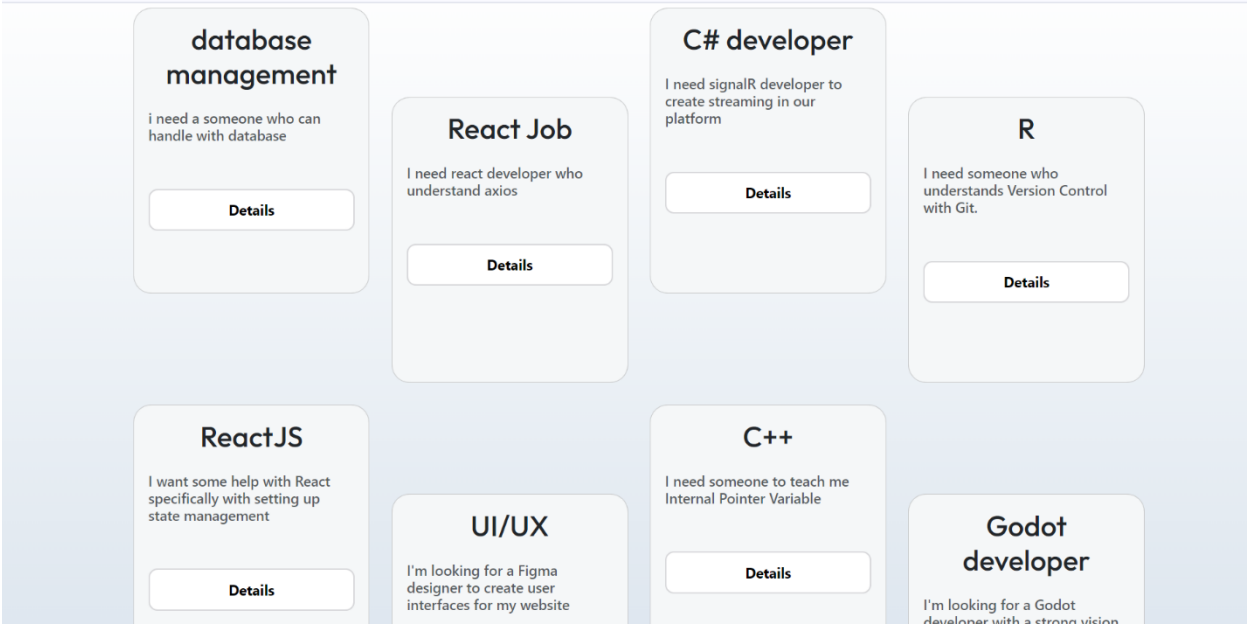

Figure 45. Requests Section

4.5.4. Apply an offer to the request

After selecting any request, this interface appears, users are able to send an offer to that request by filling out the offer information see Figure (46).

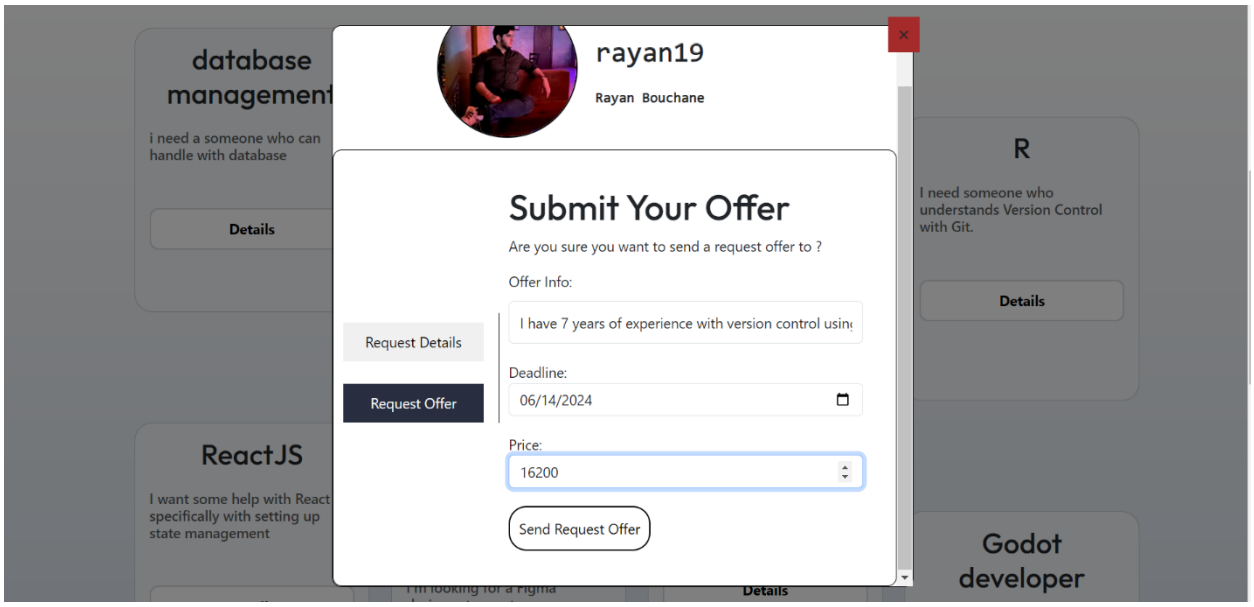

Figure 46. Apply offer

4.6. Create Section

If the user wants to create a request, an Exchange skill, or a Wishlist, he can do it from this section, see Figure (47).

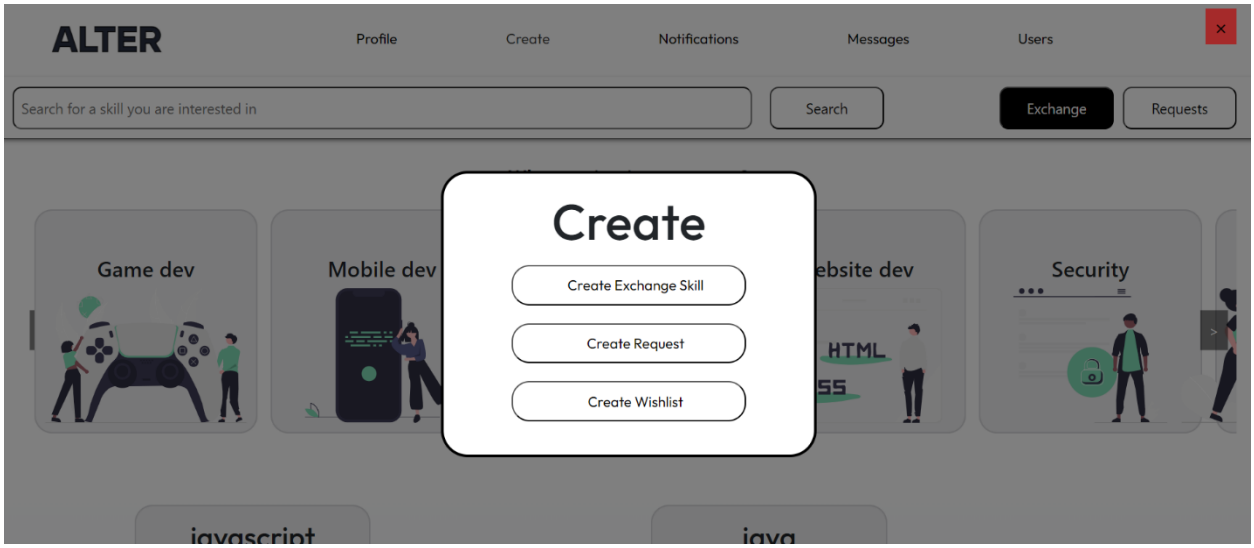

Figure 47. Create Section

4.6.1. Create skill listing

This interface represents the stages of creating skill Exchange listening see Figure (48,49,50,51).

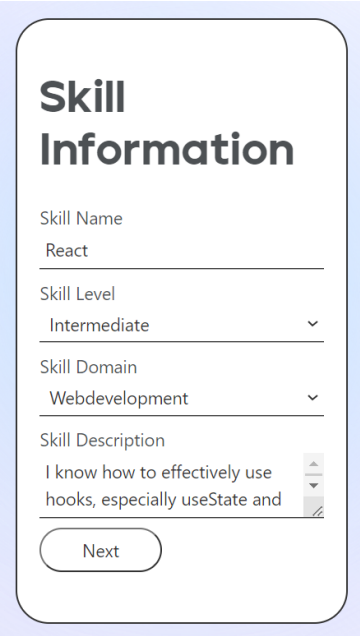

Figure 48. Create Skill

Figure 49. Add links to skill

Figure 50. Add Languages to skill

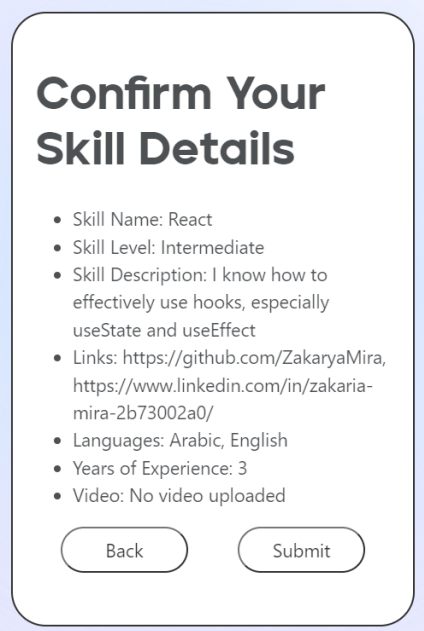

Figure 51. Confirm Skill

4.6.2. Create request

The following interface Represents the process of creating a request see Figure (52).

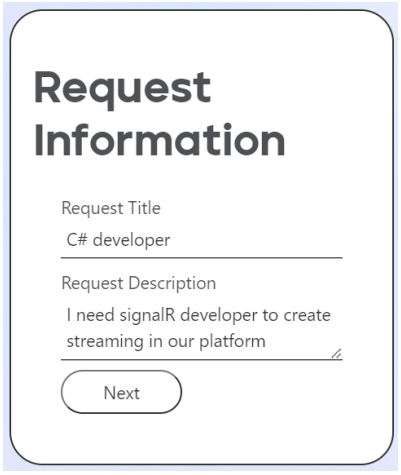

Figure 52. Create Request

4.7. Profile Page

This is the profile page, where users can see their personal information's, their uploaded skills. And their requests, (53).

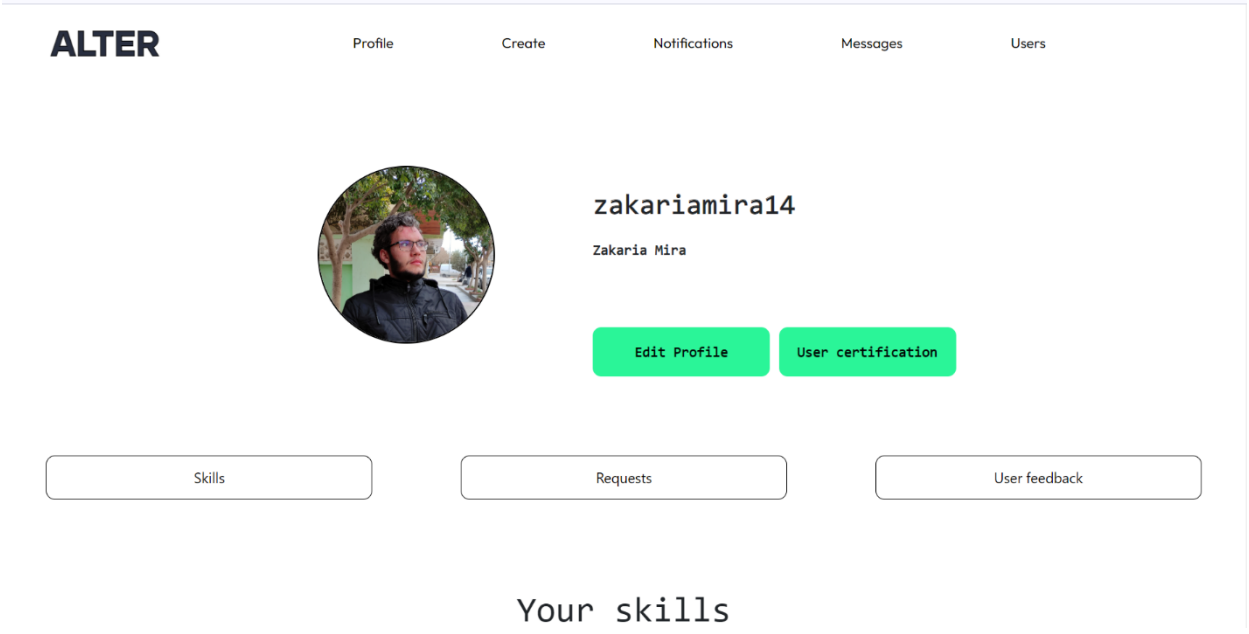

Figure 53. Profile Page

4.7.1. Skills

Here users can see all of their skills and Edit them, see Figure (54).

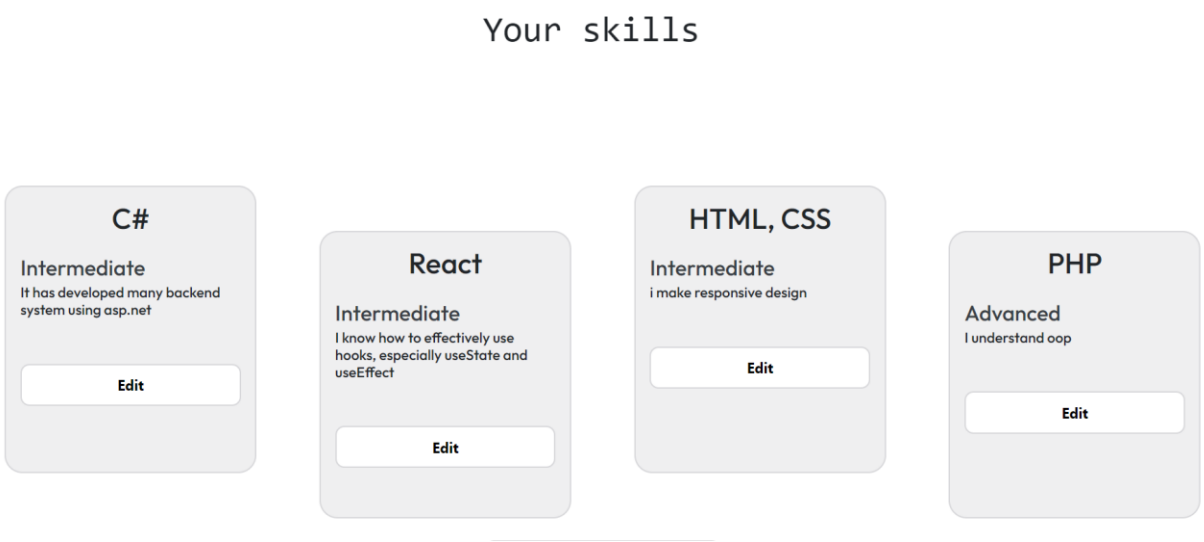

Figure 54. Skills of profile

4.7.2. Request

Here users can see their requests and the offers they got from each request, see Figure (55).

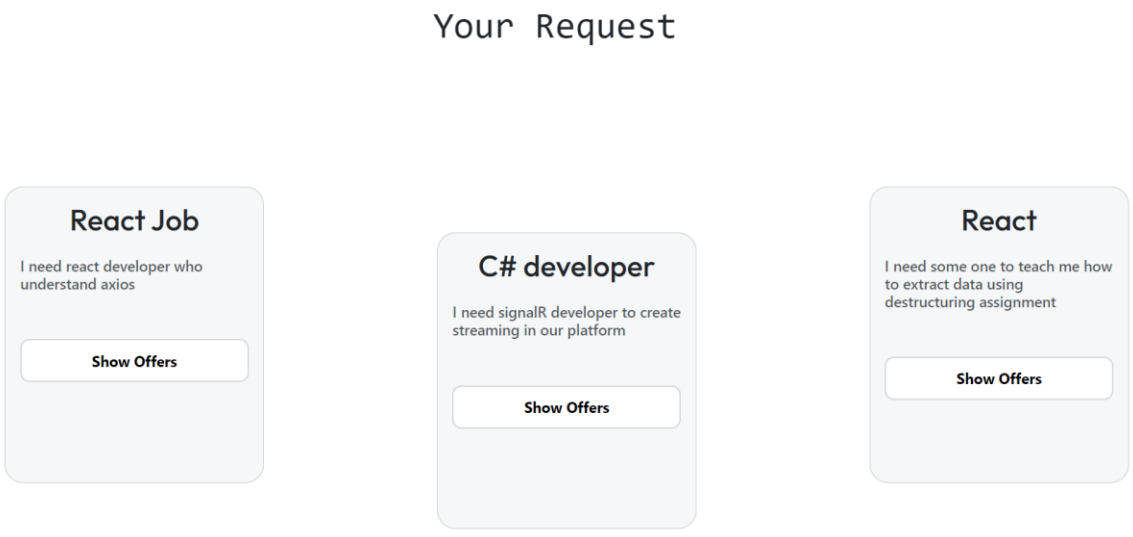

Figure 55. Requests of profile

Users are able to view the offers they got, their information's, and choosing the best offer for their need, see Figure (56).

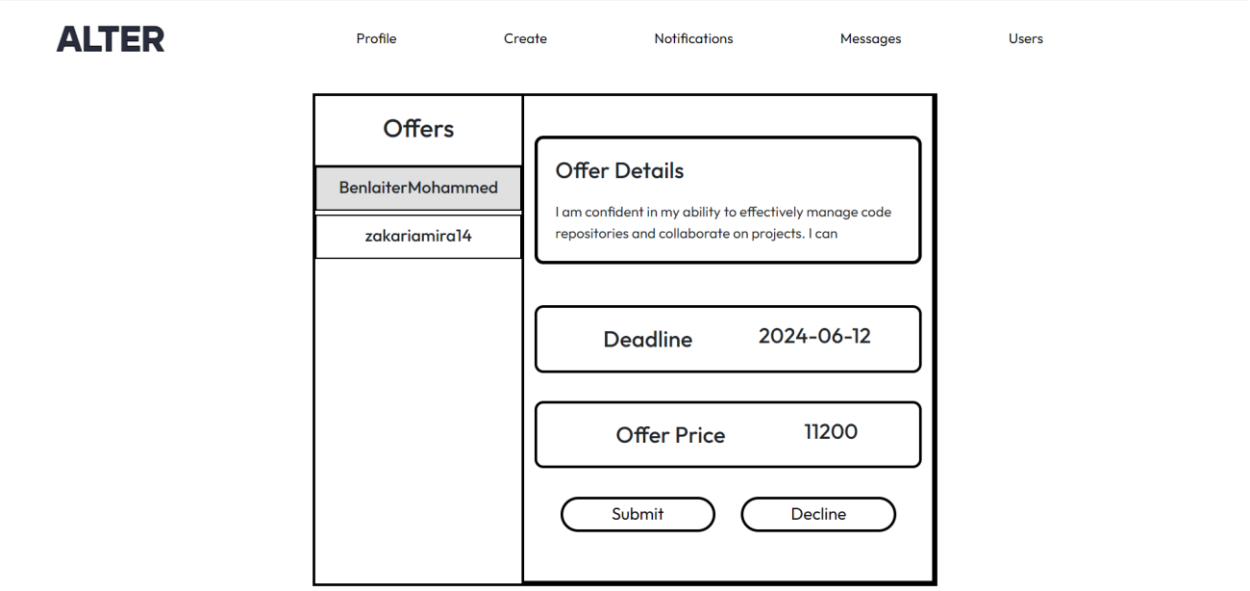

Figure 56. Offers Page

4.8. Notification Page

On this page, users can see the received notifications from all the people who send a skill exchange, the user who recived the skill exchange notification have to option to choose between accepting or rejecting that skill exchange see Figure (57).

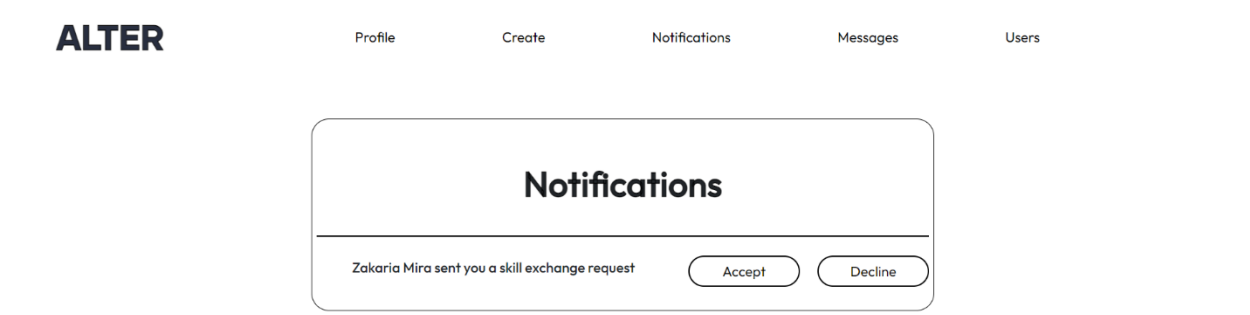

Figure 57. Notification Page

4.9. Messages Page

The message page is the meeting point for users who have successfully exchanged skills or a successful request, see Figure (58).

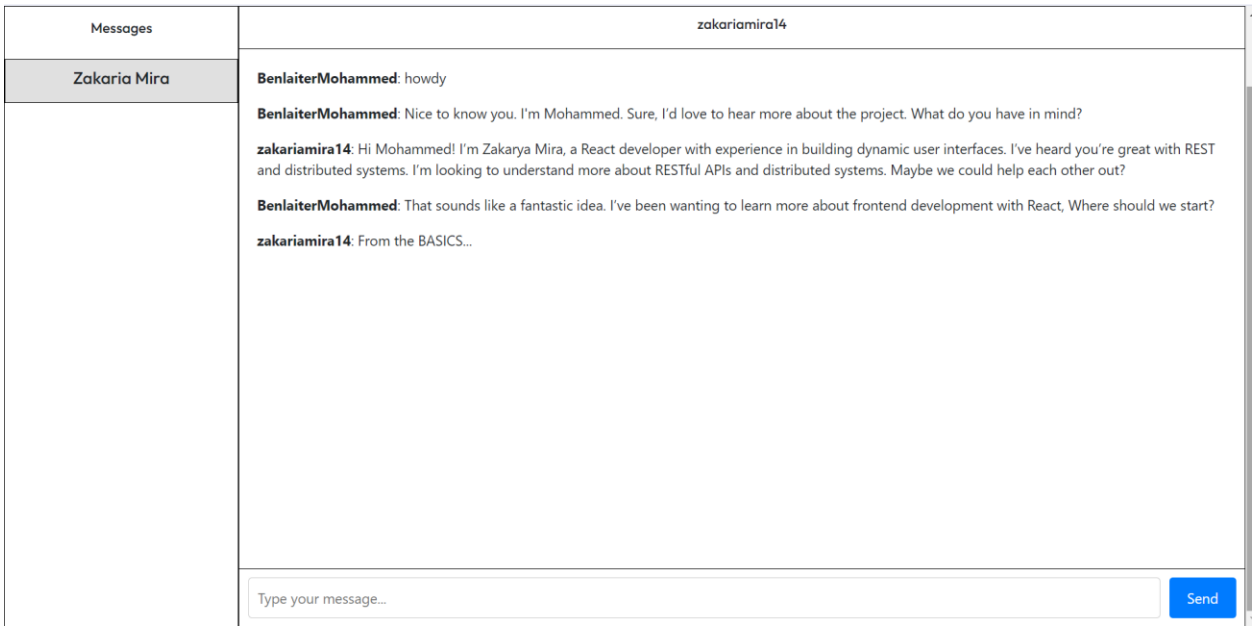

Figure 58. Messages Page

4.10. User Page

This page displays all users of this platform and provides you with the ability to search for a specific user see Figure (59).

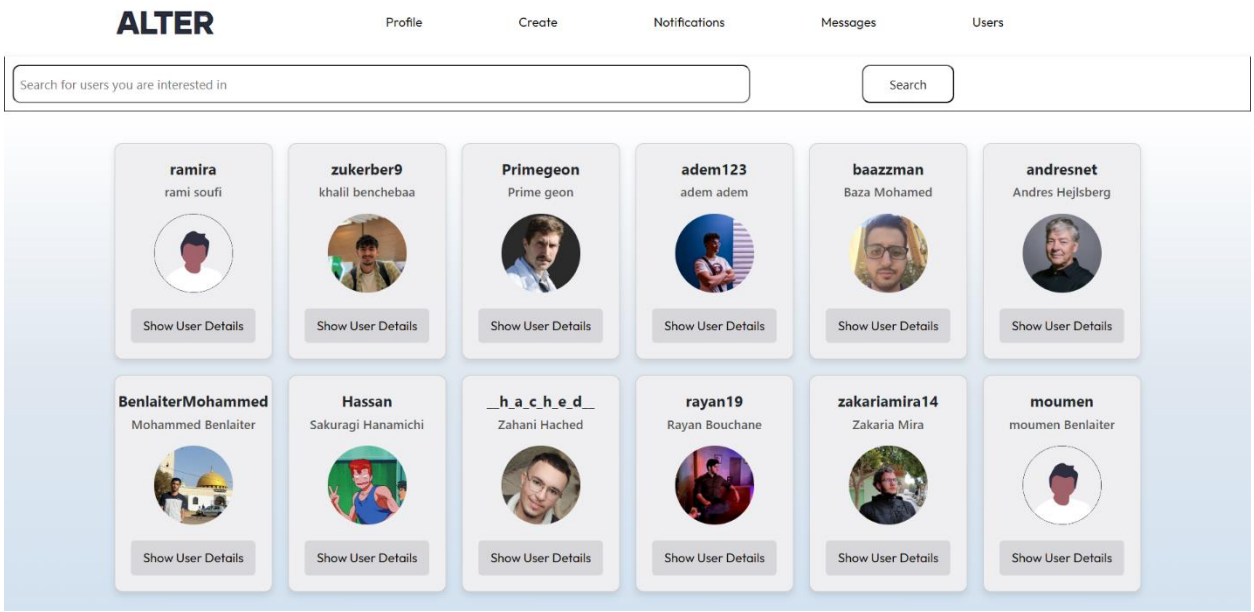

Figure 59. Users Page

4.10.1.Profile of user

This page represents an entry to other user's page, and from here you can request, or skill exchange them, see Figure (60).

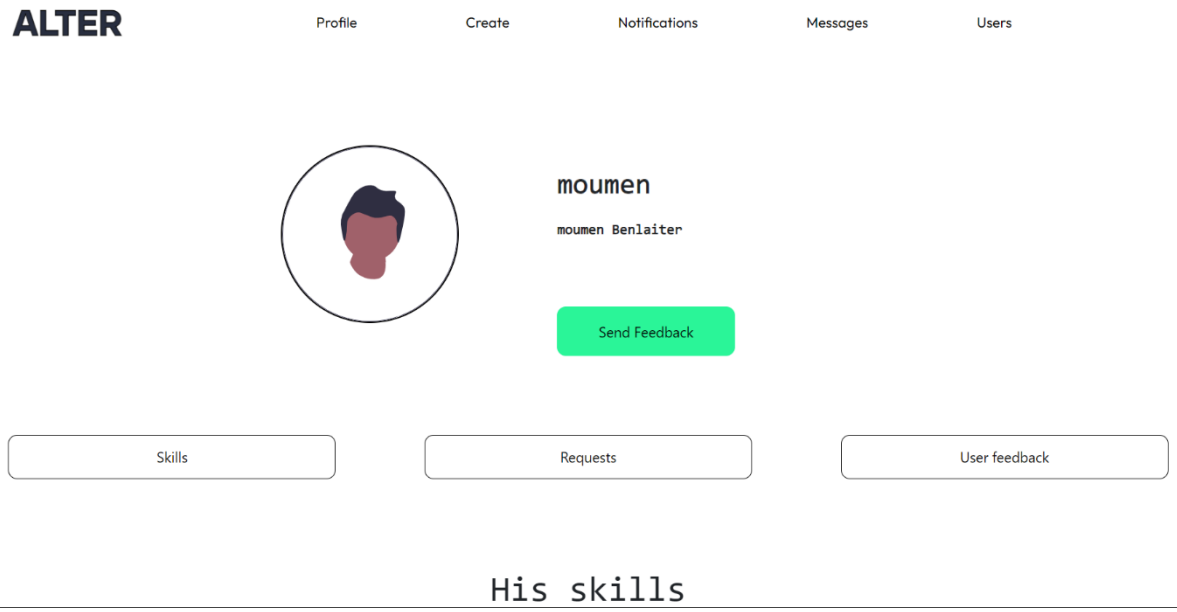

Figure 60. User Page

4.10.2.Sending Feedback

You can also send Feedbacks to other users see Figure (61).

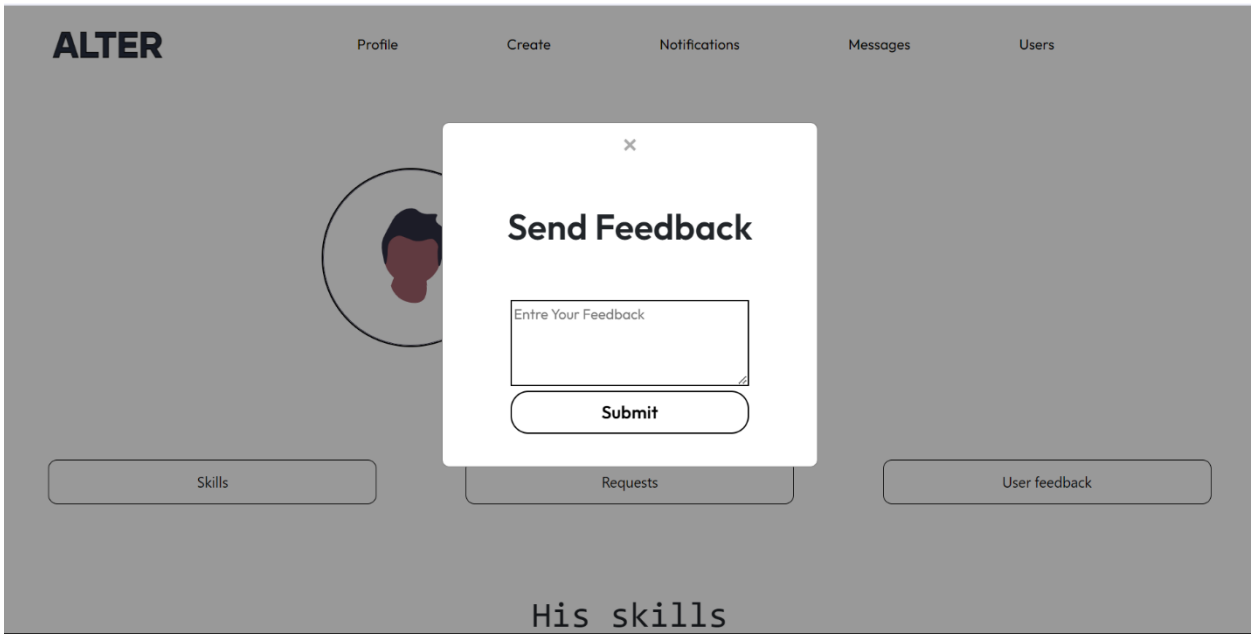

Figure 61. Feedback

5. Conclusion

In this chapter, we discussed the applied part of this project, knowing all the technologies that were used and their purpose, and presenting the most important interfaces of this application to understand its functions more.

Conclusion

This work aims to advance the field of computer science by creating a platform that connects the developer community, enabling them to exchange skills, assist one another, and offer paid services to those seeking project help.

First, we discuss the system analysis, its requirements, and the specific characteristics it must meet. We present real-world examples and compare them to our application, highlighting the added value our platform provides.

Next, we focus on designing the system and propose a structure that explains how the platform operates through various scenarios, developing a comprehensive framework to guide the platform's development.

Finally, we outline the key technologies used in this project and detail how the platform was developed, accompanied by screenshots demonstrating the application's functionality.

To further enhance this work, we suggest potential future improvements and expansions, including new features and broader community engagement strategies. Defining metrics or criteria for evaluating the platform's success, such as user engagement statistics and satisfaction ratings, will provide a clear measure of its impact.

References

[1] Charband, Y., & Jafari Navimipour, N. (2016). Online knowledge sharing mechanisms: a systematic review of the state of the art literature and recommendations for future research. *Information Systems Frontiers*, *18*, 1131-1151.

[2] Pang, S., Bao, P., Hao, W., Kim, J., & Gu, W. (2020). Knowledge sharing platforms: An empirical study of the factors affecting continued use intention. Sustainability, 12(6), 2341.

[3] Kwahk, K. Y., & Park, D. H. (2016). The effects of network sharing on knowledge-sharing activities and job performance in enterprise social media environments. *Computers in Human Behavior*, *55*, 826-839.

[4] Zhang, X., Jiang, S., Xiao, Y., & Cheng, Y. (2018). Global challenges and developmental lessons in the knowledge sharing economy. *Journal of Global Information Technology Management*, *21*(3), 167-171.

[5] *What is a Web App? - Web Application Explained - AWS. (n.d.). Amazon Web Services, Inc. Retrieved March 20, 2024, from https://aws.amazon.com/what-is/web-application/?nc1=h_ls*

[6] Codecademy. (n.d.). *What is a Web App?* Codecademy. Retrieved March 20, 2024, from https://www.codecademy.com/article/what-is-a-web-app

[7] Petrenko, V., & Litslink. (2024, April 29). *Understanding contemporary web application architecture: key components, best practices, and beyond*. Litslink. Retrieved May 4, 2024, from https://litslink.com/blog/web-application-architecture

[8] *MVC - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. (2023, December 20). MDN Web Docs. Retrieved May 4, 2024, from https://developer.mozilla.org/en-US/docs/Glossary/MVC*

[9] *Model-View-Controller: Does the user interact with the View or with the Controller? (n.d.). Software Engineering Stack Exchange. Retrieved May 6, 2024, from https://softwareengineering.stackexchange.com/questions/234116/model-view-controller-does-the-user-interact-with-the-view-or-with-the-controll*

[10] Codecademy. (n.d.-b). *What is REST?* Codecademy. Retrieved May 6, 2024, from https://www.codecademy.com/article/what-is-rest

[11] *10 types of web application development & which to choose | BairesDev. (2022, August 30). BairesDev. Retrieved May 6, 2024, from https://www.bairesdev.com/blog/types-web-application-development/*

[12] Das, S. (2024, January 29). *Web Application Development: Process, Tools, & Examples | BrowserStack*. BrowserStack. Retrieved May 7, 2024, from https://www.browserstack.com/guide/web-application-development-guide

[13] *What is Unified Modeling Language (UML)? (n.d.). Retrieved May 9, 2024, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/*

[14] GeeksforGeeks. (2024, February 9). *Use case diagrams Unified Modeling Language (UML)*. GeeksforGeeks. Retrieved May 9, 2024, from https://www.geeksforgeeks.org/use-case-diagram/

[15] *UML Class Diagram Tutorial. (n.d.-b). Lucidchart. Retrieved May 9, 2024, from https://www.lucidchart.com/pages/uml-class-diagram*

[16] *UML Sequence Diagram Tutorial*. (n.d.). Lucidchart. Retrieved May 10, 2024, from https://www.lucidchart.com/pages/uml-sequence-diagram

[17] *What is .NET? An open-source developer platform. (n.d.-b). Microsoft. Retrieved May 19, 2024, from https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet*

[18] *JavaScript | MDN. (2024b, March 5). MDN Web Docs. Retrieved May 19, 2024, from https://developer.mozilla.org/en-US/docs/Web/JavaScript*

[19] *HTML: HyperText Markup Language | MDN. (2024b, April 25). MDN Web Docs. Retrieved May 19, 2024, from https://developer.mozilla.org/en-US/docs/Web/HTML*

[20] *CSS: Cascading Style Sheets | MDN. (2024b, March 5). MDN Web Docs. Retrieved May 19, 2024, from https://developer.mozilla.org/en-US/docs/Web/CSS*

[21] *PostgreSQL. (2024b, June 1). PostgreSQL. Retrieved June 1, 2024, from https://www.postgresql.org/*

[22] Page, D. (n.d.-b). *pgAdmin 4 — pgAdmin 4 8.7 documentation*. Retrieved June 1, 2024, from https://www.pgadmin.org/docs/pgadmin4/8.7/index.html

[23] *ASP.NET Core | Open-source web framework for .NET. (n.d.-b). Microsoft. Retrieved June 1, 2024, from https://dotnet.microsoft.com/en-us/apps/aspnet*

[24] Jcjiang. (n.d.-b). *Entity Framework documentation hub*. Microsoft Learn. Retrieved June 1, 2024, from https://learn.microsoft.com/en-us/ef/

[25] *Describing the UI – react*. (n.d.-b). Retrieved June 1, 2024, from https://react.dev/learn/describing-the-ui

[26] *Getting started | Axios Docs*. (n.d.-b). Retrieved June 2, 2024, from https://axios-http.com/docs/intro

[27] Bradygaster. (2020b, September 9). *Introduction to SignalR*. Microsoft Learn. Retrieved June 2, 2024, from https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr

[28] Alexandrea, J. (2023b, mai 19). What is Bootstrap ? Hostinger Tutorials. Retrieved June 2, 2024, from https://www.hostinger.com/tutorials/what-is-bootstrap/

[29] *Documentation for Visual Studio code*. (2021b, November 3). Retrieved June 2, 2024, from https://code.visualstudio.com/docs

[30] *Postman API Platform*. (n.d.-b). Retrieved June 2, 2024, from https://www.postman.com/

[31] *REST API Documentation Tool | Swagger UI. (n.d.-b). Retrieved June 2, 2024, from https://swagger.io/tools/swagger-ui/*

[32] *Free online UI design tool & software for teams | FigMa. (n.d.-b). Figma. Retrieved June 2, 2024, from https://www.figma.com/ui-design-tool/*

[33] *draw.io - free flowchart maker and diagrams online*. (n.d.-b). Retrieved June 2, 2024, from https://app.diagrams.net/

[34] auth0.com. (n.d.-b). *JWT.IO - JSON Web Tokens Introduction*. JSON Web Tokens - jwt.io. Retrieved June 2, 2024, from https://jwt.io/introduction

[35] Zang, A. (2024b, May 16). ASP.NET Core Basics: Authentication and Authorization with JWT. *Telerik Blogs*. Retrieved June 2, 2024, from https://www.telerik.com/blogs/asp-net-core-basics-authentication-authorization-jwt

ملخص :

مع ثورة التطور التكنولوجي أصبح بالإمكان للأفراد تجاوز التحديات المعقدة، خاصة في مجال الحوسبة. يعتبر الإلمام بمجموعة متنوعة من المجالات والتقنيات أمرًا أساسيًا عند مطوري البرمجيات مما يجعل التعاون أمرًا حيويًا. يقترح مشروعنا منصة لتبادل المهارات، تسهل مشاركة المهارات عبر الإنترنت بين المطورين. يمكن للمستخدمين تبادل الخبرات في مجالات معينة وطلب المساعدة مقابل دفع مبلغ مالي، مما يعزز تطور البرمجيات. تهدف هذه المنصة إلى توفير بيئة تعاونية لتبادل المعرفة. من خلال هذه المبادرة، يمكن للأفراد الوصول وتعلم مهارات جديدة كما يساهمون في الرقي بالخبرات الجماعية لمجتمع المطورين.

كلمات مفتاحية: منصة تبادل المهارات، مجتمع المطورين، التعاون، مشاركة المهارات عبر الإنترنت.

Abstract

Over time, technological advancements have revolutionized society, enabling individuals to overcome complex challenges, particularly in computer science. In the developer community, where proficiency in diverse fields and technologies is essential, collaboration has become vital. Our project proposes a Skill Exchange Platform, facilitating online skill sharing among developers. Users can exchange expertise and request assistance in exchange for payment, fostering community growth. This platform aims to provide a collaborative environment for knowledge exchange. Through this initiative, individuals can access and learn new skills while contributing to the collective expertise of the developer community.

Key words: Skill Exchange Platform, Developer community, Collaboration, Online skill sharing.

Résumé

Au fil du temps, les avancées technologiques ont révolutionné la société, permettant aux individus de surmonter des défis complexes, notamment en informatique. Dans la communauté des développeurs, où la maîtrise de divers domaines et technologies est essentielle, la collaboration est devenue vitale. Notre projet propose une plateforme d'échange de compétences, facilitant le partage de compétences en ligne entre les développeurs. Les utilisateurs peuvent échanger leur expertise dans des domaines spécifiques et demander de l'aide en échange d'un paiement, favorisant ainsi la croissance de la communauté. Cette plateforme vise à améliorer les avancées en informatique en fournissant un environnement collaboratif pour l'échange de connaissances. Grâce à cette initiative, les individus peuvent accéder et apprendre de nouvelles compétences tout en contribuant à l'expertise collective de la communauté des développeurs.

Mots clés : Plateforme d'échange de compétences, Communauté des développeurs, Collaboration, Partage de compétences en ligne.